HO GENT

Faculteit Bedrijf en Organisatie

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

Tijs Martens

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

Promotor:
Karine Samyn
Co-promotor:
Simon Floré

Instelling: Bothrs

Academiejaar: 2019 - 20202

Tweede examenperiode

Faculteit Bedrijf en Organisatie

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

Tijs Martens

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

Promotor:
Karine Samyn
Co-promotor:
Simon Floré

Instelling: Bothrs

Academiejaar: 2019 - 20202

Tweede examenperiode

Woord vooraf

Samenvatting

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (**Pollefliet2011**):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1 Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgelijnd zijn. Doelgroepen als "bedrijven," "KMO's," systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2 Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specifiëren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3 Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is 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. Literatuurstudie

In de literatuurstudie van dit onderzoek wordt in eerste instantie onderzocht wat een progressive web application (PWA) is en hoe het werkt.

Vervolgens wordt er een lijst gemaakt met functionaliteiten die native applicaties ter beschikking hebben. Per functionaliteit zal er bekeken worden of deze geïmplementeerd kan worden in een PWA. Als een functie beschikbaar is, zal ook kort toegelicht worden hoe deze kan geïmplementeerd worden.

Bij het beslissen of een PWA gebruikt zal worden of niet voor een project is het belangrijk om te weten wat de voor- en nadelen zijn van deze technologie. Ook deze worden verwerkt in de literatuurstudie.

Uiteindelijk zal er ook bekeken worden met welke andere technologieën ook applicaties gemaakt kunnen worden waarbij er maar 1 codebase is. De voor- en nadelen van deze technologieën zullen ook besproken worden.

In een laatste fase zal geconcludeerd worden, op basis van al de vergaarde informatie, voor welk type applicaties er wel gebruik gemaakt kan worden van een PWA.

2.1 Wat is een PWA

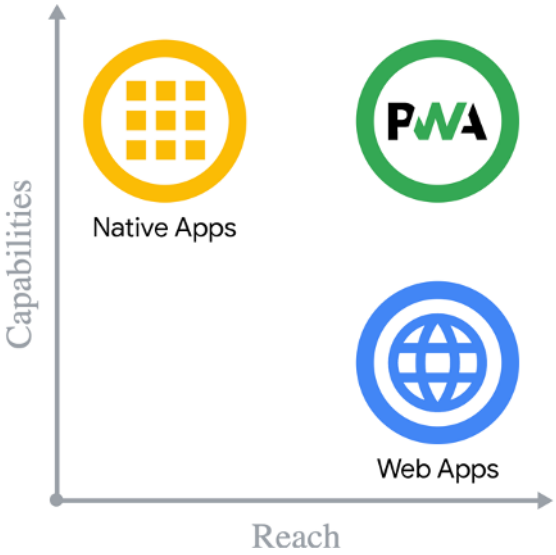
Het web is een platform waar applicaties kunnen gepubliceerd worden zonder afhankelijk te zijn van een overkoepelend bedrijf of organisatie. Voor een website is er slechts één codebase en de laatste versie is steeds beschikbaar voor de gebruiker. Dit allemaal zorgt ervoor dat een webapplicatie iedereen overal kan bereiken en dit op elk mogelijk toestel.

Native applicaties zijn betrouwbaar en bieden een heel goede gebruikerservaring. Ze starten op als een alleenstaande toepassing en ze kunnen uitgebreid gebruik maken van het besturingssysteem: ze kunnen bestanden lezen en schrijven, gebruik maken van usb-connecties en bluetooth, ze hebben toegang tot de contacten en de kalender en nog veel meer. Native applicaties voelen aan alsof ze deel uitmaken van het toestel waarop ze werken.

We kunnen dus stellen dat webapplicaties de bovenhand hebben in bereik maar dat native applicaties de bovenhand hebben als het op functies aankomt.

Een Progressive web application (PWA) is een webapplicatie die gebruik maakt van moderne web APIs om functies aan te bieden die voordien enkel beschikbaar waren voor native applicaties. De bedoeling van PWAs is om de sterktes van webapplicaties (het bereik) en native applicaties (de functionaliteit) te combineren.

(Richard & LePage, 2020) (Google, Microsoft & Awwwards, 2020)



voorstelling van wat is een PWA (Richard & LePage, 2020)

2.1.1 Service workers

De service worker is een script dat veel functionaliteiten beschikbaar maakt die voordien enkel beschikbaar waren voor native applicaties. In dit hoofdstuk wordt er bekeken welke functionaliteiten de service worker juist beschikbaar maakt en hoe dit gebeurt.

Wat is een service worker

Een service worker is een web worker die tussen het netwerk en de applicatie wordt geplaatst. Dit zorgt ervoor dat de service worker inkomende en uitgaande netwerkverzoeken kan controleren en eventueel manipuleren. (D. Mozilla, 2020)

Een web worker is een script dat in de achtergrond van een applicatie werkt en die onafhankelijk is van de andere scripts. Web workers hebben dus geen impact op de prestaties van op de webapplicatie die er gebruik van maakt. Web workers hebben geen toegang tot de DOM van een webapplicatie, ze kunnen de inhoud van een website dus niet rechtstreeks manipuleren.

(Verdú & Pajuelo, 2015) (Hiltunen, 2018)

Services workers werken dus constant op de achtergrond, maar de manier waarop service workers opgebouwd zijn (zie hoofdstuk service worker lifecycle) heeft geen significante invloed op de batterijduur van een mobiel toestel.

(Ivano, 2016)

Functionaliteiten die een serivce worker mogelijk maakt

De service worker werkt onafhankelijk van de applicatie. Dit houdt in dat een service worker wel nog kan werken terwijl de applicatie afgesloten is. Hierdoor zijn volgende functies mogelijk binnen een webapplicatie:

Offline gebruik

en er is geen internetverbinding, dan zal de service worker de client antwoorden met een boodschap dat er geen internetverbinding is. Zonder een service worker zou de applicatie crashen.

Met service workers kunnen netwerkverzoeken en paginas ook gecached worden. Als een pagina geladen wordt, kunnen alle elementen opgeslagen worden op het toestel. Als deze pagina later opnieuw bezocht wordt, hoeft deze niet meer aan de server gevraagd te worden. Hierdoor wordt de applicatie sneller en minder afhankelijk van de netwerkverbinding. Volgens onderzoek, dat uitgevoerd werd door Google, verlaten 53% procent van de gebruikers een website als deze niet geladen is binnen 3 seconden. Service workers kunnen dus helpen om het aantal gebruikers op jouw website te verhogen.

(Google & awwwards, 2017)

De twee mechanismes die gebruikt worden om data offline beschikbaar te maken zijn indexedDB en de cache API. (Osmani & Cohen, 2019) (Mozilla, 2020)

De cache API wordt gebruikt om data die verkregen werd van netwerkverzoeken op te slaan. Zowel de request als de response van een netwerkverzoek kunnen in de cache API opgeslagen worden. (Scales, 2019)

IndexedDB is een mechanisme dat gebruikt wordt om lokaal gestructureerde data op te slaan. Het kan vergeleken worden met traditionele relationele databasemanagementsystemen. Er wordt echter geen gebruik gemaakt van kolommen maar van javascript-objecten. Een IndexedDb maakt gebruik van indexen. Dit heeft als voordeel dat het uitlezen van

data snel kan gebeuren. (Mozilla, 2019b)

Notificaties

Er zijn twee soorten notificaties: lokale notificaties en push notificaties. Lokale notificaties worden geactiveerd vanop de applicatie van de gebruiker, er zijn geen externe invloeden die deze notificatie activeren.

Binnen lokale notificaties kunnen we nog het onderscheid maken tussen persistente en niet-persistente notificaties. Niet-persistente notificaties zijn notificaties die enkel getoond kunnen worden als de applicatie geopend is. Dit type notificaties heeft geen service worker nodig. Persistente notificaties zijn notificaties die nog steeds geactiveerd worden vanuit de code op het toestel, maar de applicatie moet niet meer actief zijn. Hier is wel een service worker nodig.

Push notificaties worden niet geactiveerd binnen de applicatie, maar worden geactiveerd door een server. Om push notificaties te gebruiken, moet er gebruik gemaakt worden van twee webAPIs: de notifications API en de Push API.

Notifications API Dit is een API die het uiterlijk en het gedrag van een notificatie zal bepalen. Deze API wordt zowel gebruikt voor lokale als voor push notifications. Om gebruik te maken van deze API moet de gebruiker expliciet toegang geven aan de applicatie.

Een voorbeeld van de code van een notificatie kan er als volgend uitzien

```
function displayNotification() {
  if (Notification.permission == 'granted') {
    navigator.serviceWorker.getRegistration().then(function(reg) {
      var options = {
        body: 'Here is a notification body!',
        icon: 'images/example.png',
        vibrate: [100, 50, 100],
        data: {
          userId: 383209489398274
        },
        actions: [
          {action: 'explore', title: 'Explore this new world',
            icon: 'images/checkmark.png'},
          {action: 'close', title: 'Close notification',
            icon: 'images/xmark.png'},
        ]
      };
      reg.showNotification('Hello world!', options);
    });
  }
}
```

Om een notificatie weer te geven, wordt er een object verwacht waar de inhoud van de melding wordt vastgelegd. Volgende keys kunnen meegegeven worden:

Body	De boodschap die in de melding staat.
Icon	Het icoontje dat in de notificatie wordt getoond.
Vibrate	Het vibratiepatroon dat de melding zal maken in milliseconden.
Data	Data is een object dat gebruikt kan worden als de gebruiker op de notificatie klikt. Dit object za
Actions	Er kunnen ook acties toegevoegd worden aan de melding. Elk object in deze array zal een knop

Tabel 2.1: tabel: beschrijving notifications API

(developers, 2019) (Mozilla, 2019c)

Push API De push API wordt gebruikt door de service worker. Als de server een notificatie verstuurt, wordt deze opgevangen door de push API. Deze push API zal dan gebruik maken van de notifications API om een melding op het toestel van de eindgebruiker te tonen. (Mozilla, 2019a) (Gaunt, 2020)

Achtergrondsynchronisatie

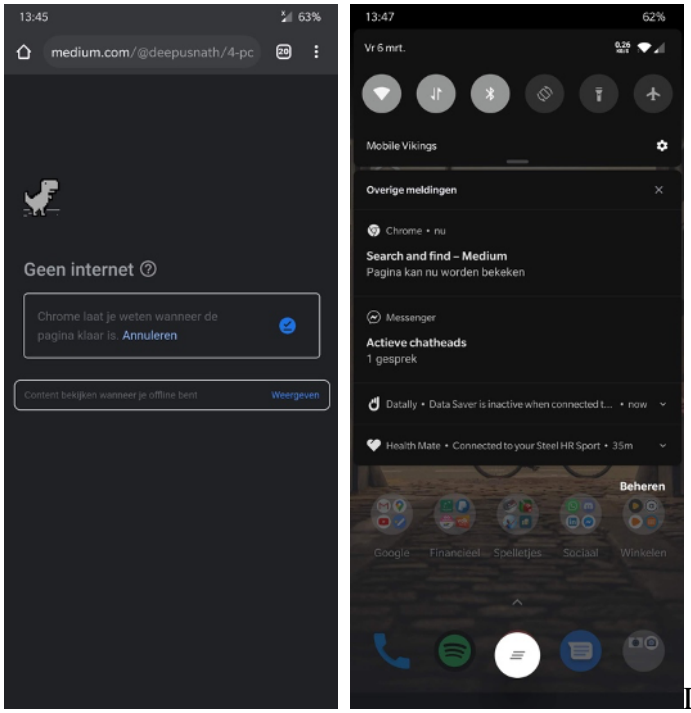
Een PWA kan gebruik maken van de background sync API om achtergrondsynchronisatie toe te passen.

Achtergrondsynchronisatie kan toegepast worden als er een trage of geen netwerkverbinding is.

Achtergrondsynchronisatie is het proces waarbij een netwerkverzoek, dat uitgevoerd werd als er geen of een te zwakke internetverbinding was, wordt opgeslagen in de service worker en wordt uitgevoerd als er wel een stabiele internetconnectie is.

Een voorbeeld hiervan is het verzenden van een bericht via een sociaal media platform. Als het bericht verzonden wordt terwijl de gebruiker offline is, zal er geen fout getoond worden maar zal dit bericht verzonden worden vanaf er internet is.

Google Chrome op Android maakt hier gebruik van. Als er een website bezocht wordt als er geen internetverbinding is, krijgt de gebruiker de melding: Chrome laat je weten wanneer de pagina klaar is. Vanaf het toestel de pagina heeft kunnen downloaden, krijgt de gebruiker een melding dat de pagina nu bekeken kan worden.



Demonstratie van achtergrond synchronisatie bij Google Chrome op Android

Service worker lifecycle

De levenscyclus van de service worker is onafhankelijk van de levenscyclus van de webapplicatie. Als de webapplicatie gesloten wordt, blijft de service worker gewoon werken.

Om een service worker te installeren moet deze geregistreerd worden in de javascript van de webapplicatie. Dit gebeurt normaal bij het eerst bezoek aan de website van de gebruiker.

```
1 function displayNotification() {
2 if ('serviceWorker' in navigator) {
3   navigator.serviceWorker.register('/service-worker.js');
4 }
```

Als een service worker wordt geïnstalleerd, worden de opgegeven statische bestanden (fotos, css-bestanden, javascript-bestanden) gedownload. Als dit slaagt, wordt er naar de activatiefase gegaan, als dit niet slaagt zal dit proces zich herhalen tot het slaagt.

Tijdens de activatiefase wordt er bekeken welke gecachete gegevens geüpdatet moeten worden en welke niet. De service worker zal de bestanden die het ontvangen heeft van het eerste netwerkverzoek vergelijken met zijn huidig cachegeheugen. Als er verschillen zijn zal dit cachegeheugen aangepast worden.

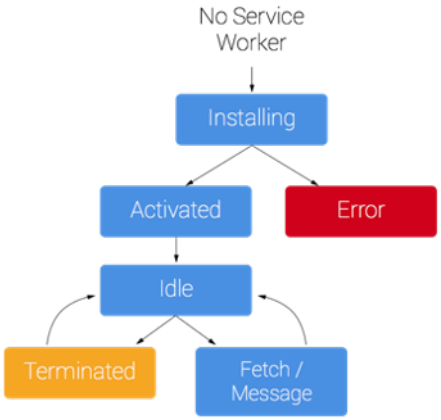
ls de activatiefase geslaagd is, heeft de service worker controle over de paginas die binnen zijn scope vallen. Deze scope moet gedefinieerd worden binnen de service worker.

Nu het oude cachegeheugen up-to-date is, zal de service worker overgaan naar een rust-

toestand, hierbij wacht de service worker op netwerkverzoeken van bestanden die binnen zijn scope vallen.

Als er een netwerkverzoek wordt verstuurd, zal de service worker deze verzoeken afhandelen. Na een bepaalde tijd zal de service worker terug naar de rust-modus gaan tot er een nieuw netwerkverzoek is. De service worker gaat naar deze rust-toestand om zowel cpu-kracht als geheugen te sparen.

(Gaunt, 2019)



Schema levenscyclus van een service worker (Gaunt, 2019)

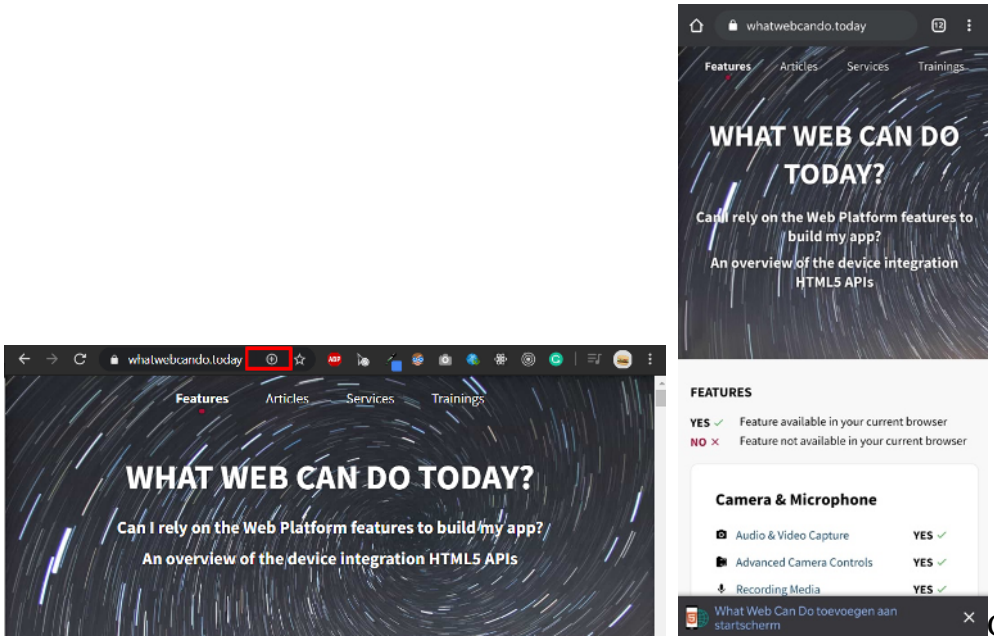
2.1.2 A2HS

Als een applicatie voldoet aan bepaalde criteria, kan deze geïnstalleerd worden op het toestel van de gebruiker. Deze functie is beschikbaar voor verschillende besturingssystemen: Windows, Mac OS, Android, IOS.

Een website moet voldoen aan volgende criteria:

- Nog niet geïnstalleerd zijn
- Een HTTPS-connectie hebben
- Een manifest.json bestand hebben
- Een service worker registreren. S

Als een website aan alle criteria voldoet zal er een beforeinstallprompt event gestart worden. Elke browser gaat hier anders mee om.



Gedrag van Google Chrome op desktop en Android.

3. Methodologie

4. Conclusie

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

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?

A.2 State-of-the-art

A.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. (**Sayali2018**)

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)

A.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 (videos)
- Maximum opslag van 50Mb op IOS
- Geen widgets

(Ivano, 2016) (Destrebecq, 2019)

A.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 & awwwards, 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 code-bases 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, Microsoft & Awwwards, 2019)

Volgens Google zijn er drie grote redenen om over te schakelen naar PWAs:

- 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)

A.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, Kasten & Halderman, 2013) Dit is een stap in de juiste richting maar het zorgt er dus niet voor dat elke PWA een veilige toepassing is. PWAs 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, Kim & Park, 2018)

A.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)

A.2.6 Apllication 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)

A.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.

A.3.1 Technologieën

PWA's kunnen gemaakt worden met de tools en technologiën 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 PWAs. 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.

A.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 PWAs. 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, Majchrzak & Gronli, 2017)

A.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 PWAs 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.

Bibliografie

Biorn-Hansen, A., Majchrzak, T. A. & Gronli, T.-M. (2017). Progressive Web Apps: The Possible Web-native Unier for Mobile Development. *SCITEPRESS*. doi:10.5220/0006353703440351

Destrebecq, O. (2019, februari 12). For your MVP: PWA or mobile application? Verkregen van https://medium.com/swlh/pwa-or-mobile-application-d26e904e814e

developers, G. (2019, mei 1). Introduction to Push Notifications. Verkregen van https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications

Gaunt, M. (2019, augustus 9). Service Workers: an Introduction. Verkregen van https://developers.google.com/web/fundamentals/primers/service-workers

Gaunt, M. (2020, maart 25). *Web Push Book*. Verkregen van https://web-push-book.gauntface.com/downloads/web-push-book.pdf

Google. (2019). Progressive Web Apps. Verkregen van https://developers.google.com/web/progressive-web-apps

Google & awwwards. (2017). *speed matters*. Verkregen van https://www.awwwards.com/brainfood-mobile-performance-vol3.pdf

Google, Microsoft & Awwwards. (2019). *Progressive Web Apps: The future of the mobile web.* Verkregen van https://www.awwwards.com/PWA-ebook/en

Google, Microsoft & Awwwards. (2020, maart 20). Progressive Web AppsThe Future of theMobile Web. *Awwwards*. Verkregen van https://www.awwwards.com/PWA-ebook/

Harris, M. (2017, oktober 10). PWAs vs Native (aka There's A Progressive Web App For That). Verkregen van https://www.youtube.com/watch?v=vhg01Ml-8pI

Hiltunen, M. (2018, december 13). Creating multiplatform experiences with progressive web apps. *Metropolia*. Verkregen van https://www.theseus.fi/bitstream/handle/10024/157245/Hiltunen_Mira.pdf?sequence=1

Ivano, M. (2016, oktober 1). Beyond Native Apps: Web Technologies To The Rescue. *ResearchGate*. Verkregen van https://research.vu.nl/en/publications/beyond-native-apps-web-technologies-to-the-rescue-keynote

Lee, J., Kim, H. & Park, J. (2018, oktober 15). 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*. Verkregen van https://cps.kaist.ac.kr/papers/18CCS-PPP.pdf

Mozilla. (2019a, december 13). https://developer.mozilla.org/en-US/docs/Web/API/Push$_A$*PI*. *Push API*.

Mozilla. (2019b, november 18). IndexedDB API. Verkregen van https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

Mozilla. (2019c, december 11). Notifications API. Verkregen van https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API

Mozilla. (2020, februari 6). Using Service Workers. Verkregen van https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers

Mozilla, D. (2020, maart 6). Making PWAs work offline with Service workers. Verkregen van https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

Osmani, A. (2016, oktober 5). Progressive Web Apps with React.js: Part 3 Offline support and network resilience. Verkregen van https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-3-offline-support-and-network-resilience-c84db889162c

Osmani, A. & Cohen, M. (2019, februari 12). Offline Storage for Progressive Web Apps. Verkregen van https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/offline-for-pwa

Richard, S. (2019, november 16). Bridging the native app gap (Chrome Dev Summit 2019). Verkregen van https://www.youtube.com/watch?v=JKVZMqpiY7w&feature=youtu.be

Richard, S. & LePage, P. (2020, maart 6). What are Progressive Web Apps? Verkregen van https://web.dev/what-are-pwas/

Scales, M. (2019, februari 12). Using the Cache API. Verkregen van https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/cache-api

Verdú, J. & Pajuelo, A. (2015, oktober 26). Performance Scalability Analysis of JavaScript Applications with Web Workers. *IEEE*, *15*, 105–108. doi:10.1109/LCA.2015.2494585

Zakir, D., Kasten, J. & Halderman, A. j. (2013, oktober 23). Analysis of the HTTPS Certicate Ecosystem. *IMC '13 Proceedings of the 2013 conference on Internet measurement conference*. doi:10.1145/2504730.2504755