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.

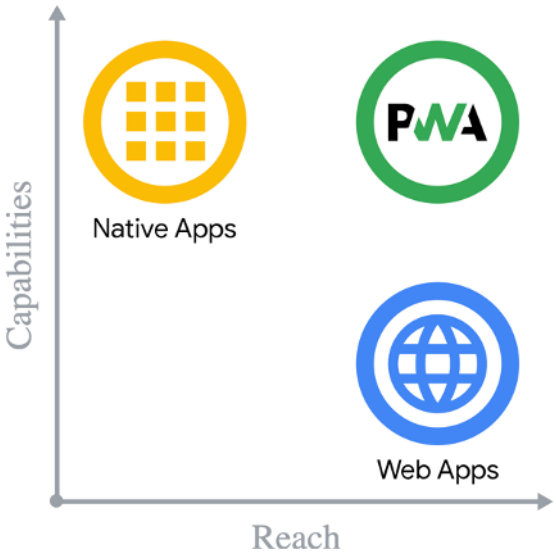
Bij het beslissen of een PWA gebruikt zal worden of niet voor een project is het belangrijk om te weten wat de voor- en nadelen zijn van deze technologie. Ook deze worden verwerkt in de literatuurstudie.

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.



Figuur 2.1: voorstelling van wat is een PWA (Richard & LePage, 2020)

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)

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, 2020a)

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:

(Developers, 2019) (Mozilla, 2019d)

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: beschrijving notifications API

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.

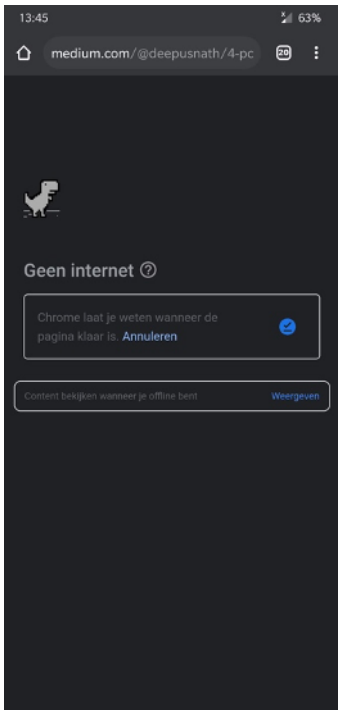
Google Chrome op Android maakt hier gebruik van. Als er een website bezocht wordt als er geen internetverbinding is, krijgt de gebruiker de melding: Chrome laat je weten wanneer de pagina klaar is. Vanaf het toestel de pagina heeft kunnen downloaden, krijgt de gebruiker een melding dat de pagina nu bekeken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

Service worker lifecycle

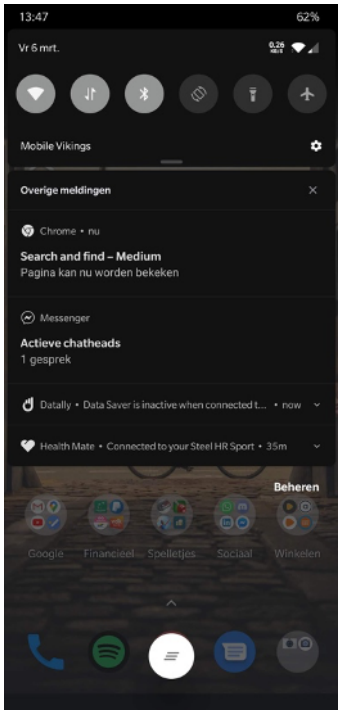
De levenscyclus van de service worker is onafhankelijk van de levenscyclus van de webapplicatie. Als de webapplicatie gesloten wordt, blijft de service worker gewoon werken.

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.

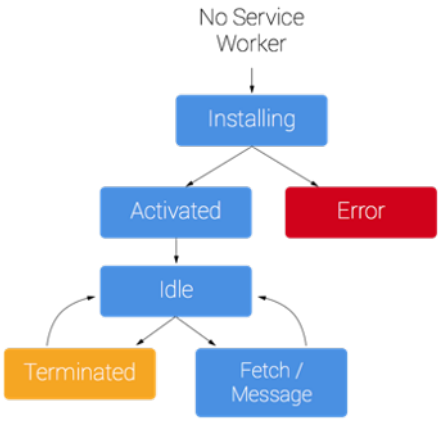
```
1 function displayNotification() {
2 if ('serviceWorker' in navigator) {
3   navigator.serviceWorker.register('/service-worker.js');
4 }
```



Figuur 2.2: demonstratie van achtergrond synchrondisatie bij Google Chrome op Android - pagina offline



Figuur 2.3: demonstratie van achtergrond synchrondisatie bij Google Chrome op Android - melding als gebruiker terug online is



Figuur 2.4: schema levenscyclus van een service worker (Gaunt, 2019)

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.

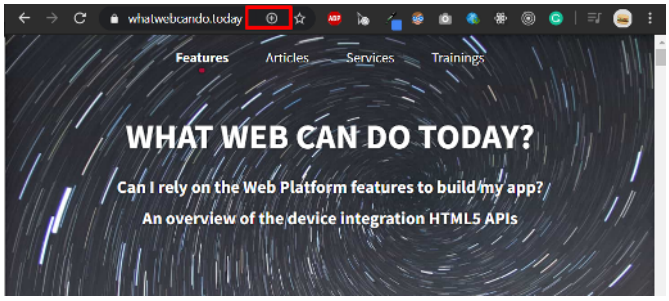
(Gaunt, 2019)

2.1.2 A2HS

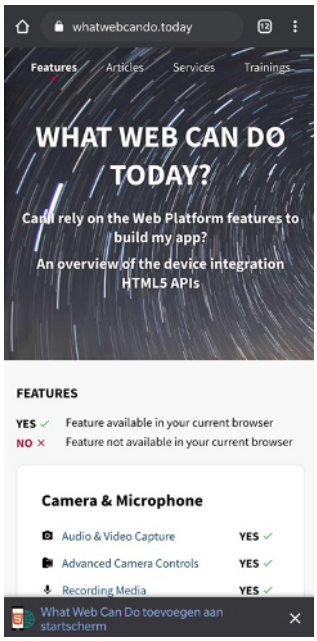
Als een applicatie voldoet aan bepaalde criteria, kan deze geïnstalleerd worden op het toestel van de gebruiker. Deze functie is beschikbaar voor verschillende besturingssystemen: Windows, Mac OS, Android, IOS.

Een website moet voldoen aan volgende criteria:

- Nog niet geïnstalleerd zijn



Figuur 2.5: gedrag van Google Chrome op Windows 10 op het beforeinstallprompt



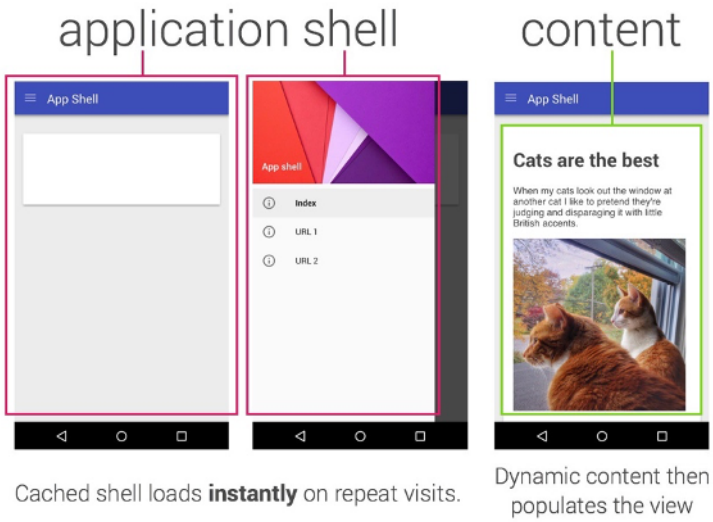
Figuur 2.6: gedrag van Google Chrome op Android op het beforeinstallprompt

- Een HTTPS-connectie hebben
- Een manifest.json bestand hebben
- Een service worker registreren.

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

Op Apple toestellen (IPhone, Mac) heeft het beforeinstallprompt geen effect. De gebruiker moet zelf op zoek gaan in het menu om de applicatie te installeren, maar dit is wel mogelijk. Het beforeinstallprompt kan wel in de code opgevangen worden. De gebruiker kan dan geïnformeerd wordendat deze webapplicatie geïnstalleerd kan worden. Er kan ook getoond worden hoe dit moet gebeuren.

(PWAbuilder, 2020)



Figuur 2.7: voorbeeld van een application shell architectuur (Osmani & Gaunt, 2015)

2.1.3 Application shell

De application shell of app shell is de minimale HTML, CSS en JavaScript die nodig is om een userinterface te tonen op een toestel. Dit is vaak de header, footer en de navigatie.

Door de bestanden die steeds terugkomen offline op te slaan, worden deze veelgebruikte elementen onmiddellijk geladen. Dit zorgt ervoor dat een webapplicatie meer zal aanvoelen als een native applicatie.

Het toepassen van deze architectuur biedt een aantal voordelen:

- Consistent snel
- Voelt native aan
- De gebruiker moet minder data downloaden

(Osmani, 2019b)

2.1.4 Progressive enhancement

Progressive enhancement is een strategie bij webontwikkeling waarbij er gezorgd wordt dat alle hoofdfunctionaliteiten beschikbaar zijn op alle toestellen. Meer geavanceerde functies worden aan deze basisversie toegevoegd.

Het doel van progressive enhancement is dat een applicatie gebruikt kan worden op elk toestel en dat meer modernere toestellen of browsers van een rijkere ervaring kunnen genieten.

(Vanhala, 2017)

2.1.5 Application manifest

Het web app manifest is een JSON-bestand dat informatie bevat over de applicatie. Deze informatie is nodig voor het installeren van een PWA op een toestel. Aan de hand van dit bestand weet het besturingssysteem bijvoorbeeld welk icoon er gebruikt moet worden en hoe het startscherm er moet uitzien.

Voorbeeld van een minimum app manifest voor de Google Maps PWA.

```
{
  "short_name": "Maps",
  "name": "Google Maps",
  "icons": [
    {
      "src": "/images/icons-192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/images/icons-512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/maps/?source=pwa",
  "background_color": "#3367D6",
  "display": "standalone",
  "scope": "/maps/",
  "theme_color": "#3367D6"
}
```

********** TABEL FIXEN **********

(LePage & Beaufort, 2020)

2.1.6 Geschiedenis van PWA's

"One last thing"

One last thing is de zin waarmee Steve jobs, co-founder van Apple, zijn jaarlijkse toespraak steeds afsloot. Er volgde meestal een revolutionair idee of product dat Apple zal uitbrengen.

In juni 2007 sloot Steve Jobs, nadat hij net de eerst IPhone had voorgesteld, zijn toespraak af met de visie die Apple toen had over hoe het web er moet uitzien. De term progressive web apps bestond nog niet, maar de concepten die hij uitlegde zijn wel de basis van PWAs. Steve Jobs citeerde volgende stellingen

- *We have got an inovative way to create applications for mobile devices, and its all based on the fact that IPhone has the full Safari engine on board*

- *You can write apps that look like IPhone apps and that integrate with IPhone services*
- *Instant distribution, just put them on the internet*
- *They are really easy to update, just put the update on your server*

(Jobs, 2007)

Apple had verwacht dat dit een succes zou zijn. Maar de ontwikkelaars waren teleurgesteld en ze hadden verwacht dat ze meer toegang zouden krijgen tot de IPhone. In de eerste versie van de IPhone kon enkel gebruik gemaakt worden van de apps die geïnstalleerd waren door Apple, er konden geen nieuwe apps gedownload worden. (Strieb, 2016)

De visie van Apple veranderde echter toen ze het volgend jaar de app-store lanceerden. (Silver, 2018)

Crome dev summit 2015

Chrome dev summit is een conferentie voor webontwikkelaars georganiseerd door Google.

Alex Russel en Andreas Bovens gebruikten voor het eerst de term progressive web apps. Op dit moment ondersteunde enkel Android service workers en A2HS.

Ook werd het concept van de application shell voorgesteld. (Russel & Bovens, 2015)

IOS 13

Op 19 september 2019 stelde Apple IOS 13 voor. Deze software-update zorgde ervoor dat de IPhone gebruik kon maken van een service worker. De applicaties kunnen nu ook geïnstalleerd worden op het toestel, de gebruiker moet dit wel nog zelf doen aan de hand van het menu. (Apple, 2020b)

2.1.7 Voorbeelden van PWA's

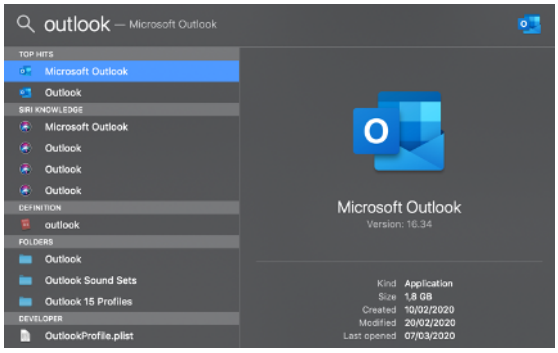
Outlook

Google is niet de enige grote speler die PWAs wil promoten en implementeren. (Microsoft, 2020)

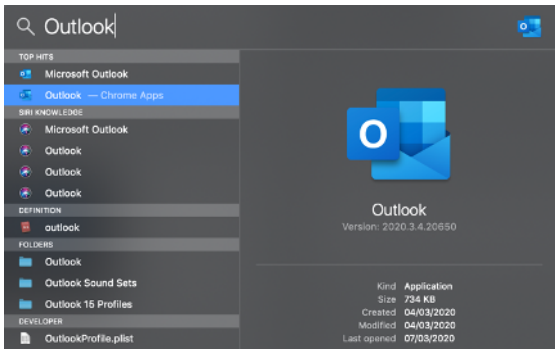
Microsoft geeft het goede voorbeeld door de veelgebruikte email client Outlook ook beschikbaar te maken als een PWA. De PWA-versie van de applicatie is slechts 800KB groot terwijl de traditionele versie op een desktop 1,8GB groot is.

AliExpress

AliExpress, één van de grootste e-commerce platformen, implementeerde hun website als een PWA. De resultaten waren heel positief, bij nieuwe gebruikers werd er 104% meer



Figuur 2.8: Outlook op Mac



Figuur 2.9: Outlook op Mac als PWA

verkocht. De gemiddelde gebruiker bleef ook 74% langer op de website.

(Developers, 2020a)

Twitter

Twitter, één van de grootste sociale media platformen, implementeerde hun website ook als PWA. De resultaten waren ook hier positief. Het datagebruik van de gebruiker werd met 70% verminderd. De gemiddelde gebruiker bekeek ook 65% meer paginas dan op de vorige website van Twitter. Het afdelingshoofd van ontwikkeling van Twitter deed volgende uitspraak over de PWA:

Twitter Lite is now the fastest, least expensive, and most reliable way to use Twitter. The web app rivals the performance of our native apps but requires less than 3% of the device storage space compared to Twitter for Android.

(Developers, 2020c) (Love, 2018)

Starbucks

Starbucks heeft een native applicatie waarmee er koffie of andere dranken besteld kunnen worden als ze in een Starbucks zaak zijn. Deze applicatie is populair bij klanten die

vaak terugkomen naar Starbucks. Klanten die slechts eenmalig komen, willen vaak geen applicatie downloaden om deze maar éénmaal te gebruiken.

Dit werd opgelost door de functionaliteit die de native applicatie biedt ook te implementeren als een PWA zodat deze kan gebruikt worden via het web zonder geïnstalleerd te worden.

De klanten kunnen nu vanuit de website een bestelling personaliseren en bestellen. Hierdoor wordt er tijd gewonnen bij het bestellingproces.

(Formidable, 2020) (Kawatka & Wala, 2020)

Uber

Uber is aan het uitbreiden naar nieuwe markten. Ze willen dat hun service ook beschikbaar wordt ongeacht de locatie, netwerksnelheid of toestel van de gebruiker.

Met deze drie parameters in gedachten hebben ze een alternatief gebouwd voor hun native mobiele applicatie. Deze website kan bezocht worden op m.uber.com.

Het grootste doel was om de website snel te laten werken op low-end toestellen met een 2g connectie.

(Croll, 2017)

Pinterest

Pinterest zag aan de hand van hun analytics dat slechts 1% van de niet geregistreerde bezoekers op hun vorige site een account aanmaakte. Dit probleem hebben ze opgelost aan de hand van een PWA. Na het implementeren van de PWA steeg het aantal registraties met 60% en het aantal sessies van langer dan 5 minuten steeg met 40%.

(Osmani, 2019a)

2.2 Besturingssystemen en PWAs

Om te weten te komen voor welke toepassingen een PWA gemaakt kan worden en voor welke toepassingen nog steeds een native applicatie nodig is, is het belangrijk om te weten wat de technische mogelijkheden zijn van een PWA. In deze sectie van de literatuurstudie wordt er bekeken welke functies, die beschikbaar zijn voor native applicaties, al dan niet gebruikt kunnen worden door PWAs.

Dit onderzoek werd gevoerd met behulp van de website whatwebcando.today en caniuse.com.

whatwebcando.today is een website die kleine voorbeelden van verschillende technologieën demonstreert. Door deze voorbeelden te testen op verschillende platformen kan er uitgemaakt worden welke technologieën er beschikbaar zijn voor het web, en op welke platformen deze beschikbaar zijn.

Caniuse.com is een website die voor verschillende web-technologieën een overzicht biedt op welke browsers deze technologie gebruikt kan worden en op welke niet. Deze website werd gebruikt om de ondervindingen van de testen die werden uitgevoerd te valideren.

Een web-API is een API die wordt aangeboden door de browser. Het verschil met web-APIs en traditionele APIs is dat web-APIs lokaal worden aangeboden door de browser en er dus geen internetverbinding nodig is om van deze functionaliteiten te genieten.

(Mozilla, 2019c)

Als er meer informatie nodig was over de web APIs werd deze gevonden op developers.Google.com of op developer.mozilla.org.

Er werd gekeken op welke platformen bepaalde functies wel en niet werkten. De volgende platformen werden onderzocht:

- Desktop:
  - Microsoft edge versie 80 op Windows 10
  - Mozilla firefox versie 73.0 op Windows 10
  - Google Chrome versie 79.0 op Windows 10
  - Safari (desktop) versie 13.0.5 op een Macbook Pro met macOS Mojave (10.14.6)

- Mobiel:
  - Google Chrome versie 80 op Android 10 op een OnePlus 6
  - Safari (mobiel) versie 13 op IOS 13 op een IPhone SE

De testen werden uitgevoerd op 7 maart 2020.

De website whatwebcando.today geeft een overzicht van de functionaliteiten aan de hand van een bepaalde structuur. Deze structuur werd overgenomen en ziet er als volgt uit:

- Media
- Verbinding
- Toestel kenmerken
- Native gedrag
- Besturingssysteem
- Input
- User experience
- Locatie en positionering
- Scherm en output

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Ja

Tabel 2.2: ondersteuning van de Media Capture API

2.2.1 Onderzoek

Media

Video met audio

Sommige van de meest populaire mobiele applicaties zijn sterk afhankelijk van camera-functionaliteit. Voorbeelden hiervan zijn Snapchat, Instagram, Messenger, WhatsApp,

Bij deze applicaties is het belangrijk dat de camera aan volgende vereisten voldoet:

- Snel en eenvoudig te gebruiken
- Er moet van camera gewisseld kunnen worden
- Er moet ingezoomd kunnen worden
- De flashlight moet gebruikt kunnen worden

De media capture API (Dzung D Tran, Ilkka Oksanen & Ingmar Kliche, 2020) maakt het mogelijk om een video die opgenomen wordt met de camera van het toestel te tonen op de webpagina. Deze video kan dan opgeslagen worden in de code en verzonden worden naar een server.

(Fransson & Driaguine, 2017)

De media capture API is ook in staat om aan het toestel te vragen welke cameras er beschikbaar zijn en dan te verwisselen van camera.

(Scales, 2020)

Ook meer geavanceerde functionaliteiten zijn beschikbaar. Het gedrag van de zoom en de flashlight kan ook programmatisch bepaald worden.

(Oberhofer, 2017) (Ogundipe, 2018)

Al de belangrijkste functionaliteiten die een gebruiker verwacht, zijn allemaal aanwezig. Applicaties die afhankelijk zijn van video-opnames kunnen dus geïmplementeerd worden als PWA.

Foto's

Het vastleggen van fotos is ook voor veel populaire applicaties belangrijk. Ook dit is een belangrijke functionaliteit voor sociale media applicaties.

Fotos die gedeeld worden op sociale media moeten vaak van een zo hoog mogelijke kwali-

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Ja	Nee	Ja	Nee

Tabel 2.3: ondersteuning van de Image Capture API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Nee

Tabel 2.4: ondersteuning van de Image Capture API

teit zijn. Op native applicaties wordt deze kwaliteit bereikt door volgende eigenschappen:

- Manuele en automatische focus
- Aanpassen van sluitersnelheid
- Aanpassen van witbalans
- Aanpassen ISO-waarde
- Gebruik maken van HDR

Fotos kunnen ook, net zoals een video, genomen worden aan de hand van de Media capture API. Deze API is echter niet in staat om deze instellingen van de camera aan te passen.

De Image Capture API (Mandyam & Casas-Sanchez, 2020) is ontwikkeld om meer controle te hebben over de camera. Deze API zorgt ervoor dat instellingen zoals witbalans, temperatuur, exposure, ISO, helderheid, contrast, saturatie, zoom, programmatisch aangepast kunnen worden.

Deze API heeft standaard geen ondersteuning voor HDR, maar dit kan zelf geïmplementeerd worden aan de hand van third-party-packages.

(Bhaumik, 2019)

Geluidopname

De mediarecorder API, (Casas-Sanchez, 2020) door meerdere browsers aangeboden, is een manier om eenvoudig geluidsfragmenten op te nemen en te importeren in een webapplicatie.

Helaas is er voor Apple-toestellen geen ondersteuning. In de toekomst zal deze functie waarschijnlijk ook beschikbaar worden voor deze toestellen. Dit wordt in de volgende versie van Safari (Safari 14) voor desktop verwacht. Voor Safari voor IOS bestaat deze functie al maar is het nog een experimentele functie die de gebruiker zelf moet activeren.

********** checken voor nieuwe update IOS 14 **********

Gelukkig is er een alternatief voorzien met HTML5-tags. Dit is een methode die voor alle platformen zal werken maar niet op dezelfde manier.

```
1 <input type="file" accept="audio/*" capture>
```

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	ja

Tabel 2.5: ondersteuning van WebRTC

Er wordt gebruik gemaakt van een inputveld waar de gebruiker een bestand kan uploaden. Door het accept attribuut wordt duidelijk gemaakt dat enkel audiofragmenten geüpload mogen worden. Het capture attribuut zorgt ervoor dat waar mogelijk de gebruiker een audiofragment kan opnemen in de default geluidsopname app. Dit fragment wordt dan automatisch geïmporteerd in de webapplicatie. Dit is enkel mogelijk op mobiele toestellen en dus niet in desktopbrowsers.

(Kinlan, 2019)

Dit is een goed voorbeeld van progressive enhancement.

Real-time communicatie

Bij de meeste populaire communicatieapplicaties zoals WhatsApp, Messenger, Skype, is videobellen mogelijk. Om dit mogelijk te maken moet er live video en audio gestreamd kunnen worden tussen twee of meer personen.

Real-time communication in the web of WebRTC (Jennings, Boström & Bruaroey, 2020) is een verzameling van APIs die het verzenden en ontvangen van real-time video en audio mogelijk maakt, zonder afhankelijk te zijn van een gecentraliseerde server. Deze server is echter wel nodig om een connectie tot stand te brengen. Eens deze connectie er is, is er een peer-to-peer verbinding.

Casting

Applicaties die media tonen aan de gebruiker kunnen deze casten naar tv-toestel. Dit gebeurt bij Apple toestellen aan de hand van Airplay en bij Android toestellen aan de hand van google cast.

YouTube is een applicatie die hier gebruik van maakt. Als er een video bekeken wordt, zal de gebruiker een optie krijgen om deze te tonen om een tv.

Op Apple toestellen kan een PWA nu ook AirPlay implementeren.

(Apple, 2020a)

De Chrome Sender API (Developers, 2020b) zorgt ervoor dat alle modern toestellen media kunnen delen op een tv of ander scherm die dit ondersteund.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Ja	Nee	Ja

Tabel 2.6: ondersteuning Apple AirPlay

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	ja

Tabel 2.7: ondersteuning van Chrome Sender API

Media-controle in de notificatie

Als een native applicatie media afspeelt op een mobiel toestel, kan deze applicatie bestuurd worden vanuit de notificatie. De afgespeelde media zal ook niet stoppen als een gebruiker de applicatie verlaat

Een voorbeeld hiervan is Spotify, in de notificatie van Spotify kan de gebruiker volgende acties uitvoeren.

- Informatie bekijken over het nummer
- Naar het volgende nummer gaan
- Het nummer pauzeren
- Het nummer toevoegen aan mijn favorieten
- De vooruitgang van het nummer zien en aanpassen

De Media Session API (François Beaufort, 2019b) zorgt ervoor dat als er media afgespeeld wordt op een website, en de browser wordt gesloten, de media niet zal stoppen met afspelen.

Deze API zorgt er ook voor dat er een notificatie komt waar de gebruiker controle heeft over de afgespeelde media.

Het voorbeeld van Spotify kan dus volledig geïmplementeerd worden als PWA.

Connectie met andere apparaten

Bluetooth

Native applicaties kunnen een verbinding maken met bluetooth-toestellen. Eens er een verbinding is, kan er informatie uitgewisseld worden tussen de toestellen. Een voorbeeld van een applicatie die hier gebruik van maakt is de Sony Headphones app. Aan de hand van deze app kan er verbinding gemaakt worden met een koptelefoon en kunnen de instellingen van de koptelefoon aangepast worden.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	ja

Tabel 2.8: ondersteuning van Media Session API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.9: ondersteuning van Web Bluetooth API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.10: ondersteuning van Web USB API

Met de Web Bluetooth API (Grant & Ruiz-Henrìquez, 2020) kan er vanuit de browser verbinding gemaakt worden met bluetooth-toestellen. De web API heeft zowel schrijf- als leesrechten bij externe toestellen.

Er kan dus geconcludeerd worden dat de Web Bluetooth API kan gebruikt worden voor applicaties die gebruik moeten maken van bluetooth-toestellen.

(François Beaufort, 2019c)

USB

Verkopers van toestellen met USB kunnen nu gebruik maken van de Web USB API, (Rockot, Grant & Ruiz-Henríquez, 2020). Bij het verbinden van een USB-toestel kan er automatisch een website geopend worden waarmee het toestel kan interageren.

Dit kan interessant zijn voor toestellen die een eenmalige set-up nodig hebben. Met deze technologie kan vermeden worden dat er overbodige software moet geïnstalleerd worden op het toestel van de gebruiker.

Dit is echter enkel mogelijk met een beperkt aantal browsers en er moet een HTTPS-verbinding zijn.

(François Beaufort, 2019a)

NFC

Near field communication of NFC is een technologie om een kleine hoeveelheid informatie uit te wisselen over een kleine afstand (Maximum 20cm). NFC wordt gebruikt om draadloze betalingen uit te voeren met een betaalkaart of met een smartphone. (Paus, 2007)

Dit is een functie met veel mogelijkheden die helaas niet beschikbaar is voor webappli-

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Nee	Nee	Nee

Tabel 2.11: ondersteuning van Web NFC API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Ja	Nee	Ja	Nee

Tabel 2.12: ondersteuning van Network information API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Ja

Tabel 2.13: ondersteuning online status

caties. Er bestaat echter wel een API om gebruik te kunnen maken van NFC (Rohde-Christiansen, Kis & Beaufort, 2020), maar de Web NFC API is een experimentele API. Dit betekent dat de eindgebruiker dit nog niet kan gebruiken.

Toestelkenmerken

Netwerkinformatie

De Network information API (Lamouri, 2020) voorziet informatie over het type netwerkverbinding die de gebruiker momenteel bezit. Deze informatie bevat het connectietype (2g, 3g, 4g) en wat de maximale downloadsnelheid is van deze verbinding.

Online status

dit is een eenvoudige eigenschap die kan opgeroepen worden op het navigator object. Deze eigenschap bevat een booleaanse waarde die waar zal zijn als de gebruiker een connectie heeft met het internet. Deze informatie kan interessant zijn bij het ontwikkelen van een PWA met offline functionaliteiten.

Vibratiemotor

De Vibration API (Kostionen, 2020) zorgt ervoor dat de vibratiemotor kan aangesproken worden vanuit de webapplicatie.

Batterijstatus

Aan de hand van de Battery Status API (Kostiainen & Lamouri, 2020) kan er informatie over de batterij van het toestel verkregen worden.

Volgende informatie kan verkregen worden:

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Nee

Tabel 2.14: ondersteuning vibratiemotor

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.15: ondersteuning batterijstatus

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.16: ondersteuning toestelgeheugen

- Aan het opladen
- Batterijpercentage
- Bij opladen, tijd tot volladen
- Bij niet opladen, tijd tot batterij leeg

Aan de hand van deze API kunnen er ook acties uitgevoerd worden op basis van het veranderen van de toestand van de batterij. Er kan bijvoorbeeld een functie uitgevoerd worden als de gebruiker zijn toestel met een energiebron verbindt.

Toestelgeheugen

de Device Memory API (Panicker, 2020) geeft informatie over het RAM-geheugen van het toestel van de gebruiker. Dit kan interessant zijn voor het laden van een eventuele lichtere versie van een website voor minder capabele toestellen.

Native gedrag

Lokale notificaties

Bij native applicaties kan een bepaalde actie binnen de app resulteren in een notificatie. Veel gezondheids-tracking applicaties maken hier gebruik van. Een gebruiker zal bijvoorbeeld een melding krijgen als een vooropgesteld aantal stappen op een dag is bereikt.

Lokale notificaties zijn beschikbaar via de Notifications API (Gregg, 2020). Lokale notificaties zijn notificaties die geen internet of server nodig hebben. Deze kunnen gepland worden bij het laden van de website. Ze worden dus lokaal geactiveerd.

Meer informatie over notificaties kan gevonden worden in het hoofdstuk functionaliteiten die een servcie worker mogelijk maakt.

Dankzij persistent local notifications en zijn service worker kan het voorbeeld van de fitness-tracking applicatie ook geïmplementeerd worden als PWA.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.17: ondersteuning lokale notificaties

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Nee

Tabel 2.18: ondersteuning push notificaties

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Nee

Tabel 2.19: ondersteuning A2HS

Push notificaties

Native applicaties kunnen genieten van notificaties die niet geactiveerd worden vanop het toestel zelf. Een voorbeeld hiervan is een sport-applicatie die een melding geeft als de gebruiker zijn favoriete voetbalploeg een doelpunt heeft gemaakt.

Push notificaties zijn notificaties die verstuurd worden vanop een server. Door gebruik te maken van de Push API (Sullivan, Fullea & van Ouwerkerken, 2020) om notificaties te ontvangen en de Notification API om notificaties op het scherm te tonen, kan een PWA push notificaties implementeren.

Meer informatie over notificaties kan gevonden worden in het hoofdstuk functionaliteiten die een servcie worker mogelijk maakt.

Door deze functionaliteiten kan het voorbeeld van een sportapplicatie ook geïmplementeerd worden als PWA.

A2HS

Door het toevoegen van een web app manifest kan je de browser duidelijk maken hoe een applicatie er moet uitzien als het toegevoegd wordt aan het startscherm. De PWA zal er dan op het startscherm gelijk uitzien als een native applicatie. Meer informatie over notificaties kan gevonden worden in het hoofdstuk Wat is een PWA.

Badges

Als een native applicatie een melding heeft ontvangen zal er een indicatie staan naast het icoontje op het startscherm. Dit kan nu ook geïmplementeerd worden voor geïnstalleerde PWAs aan de hand van de Badging API (LePage, 2020)

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Onbekend	Ja	Nee	Ja	Nee

Tabel 2.20: ondersteuning badges

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Ja

Tabel 2.21: ondersteuning voorgrond detectie

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	nee	Ja	nee

Tabel 2.22: ondersteuning Permissions API

Voorgrond-detectie

Native applicaties kunnen detecteren als een applicatie op de voorgrond wordt gebruikt. YouTube maakt hier gebruik van om zeker te zijn dat de gebruiker de applicatie actief gebruikt op het moment dat een advertentie getoond wordt.

Met de Page Visibility Detection API (Grigorik, Jain & Mann, 2020) kan gedetecteerd worden of een applicatie in de voorgrond gebruikt wordt of niet.

Aan de hand van deze applicatie kan het gedrag van de applicatie aangepast worden als de gebruiker de applicatie niet meer in de voorgrond gebruikt.

Toestemmingen

Om gebruik te maken van hardware functies van een toestel is vaak, om privacy redenen, de toestemming van de gebruiker nodig. Hiervoor is de Permissions API (Cáceres, Lamouri & Yasskin, 2020) ontwikkeld. Er kan toestemming gevraagd worden op een gelijkaardige manier voor verschillende functies.

Functies waarvoor toestemming gevraagd kan worden:

- Locatie
- Notificaties
- Push-notificaties
- Midi (musical instrument digital interface)
- Klembord
- Camera
- Microfoon
- Achtergrondsynchronisatie
- Lichtsensor
- Versnellingsmeter
- Gyroscoop
- Magneetsensor
- Betalingen

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Ja

Tabel 2.23: ondersteuning web storrage

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Ja

Tabel 2.24: ondersteuning IndexedDB

Besturingssysteem

offline opslage

Native applicaties kunnen nog steeds gebruikt worden als er geen internetverbinding is. Toepassingen die geen netwerkverzoeken doen zijn dus nog volledig operationeel zonder internetverbinding.

Er zijn verschillende technologieën om data offline op te slaan.

- Web storage
- IndexedDB
- Cache API
- Storage API

Web storage De meest eenvoudige manier om data op te slaan. Er kunnen key-value paren opgeslagen worden in het localStorage of in het sessionStorage. (Hickson, 2020)

IndexedDB Een API voor het opslaan van grote hoeveelheden gestructureerde data op het toestel van de eindgebruiker. De data kan snel gelezen worden omdat er indexen gebruikt worden. (Alabbas & Bell, 2020)

Cache API Deze API is gespecialiseerd in het opslaan van netwerkverzoeken. Dit is heel handig in samenwerking met een serviceworker. API-calls kunnen opgeslagen worden voor offline gebruik. (van Kesteren & Hickson, 2020)

Storage API Data die is opgeslagen in een van vorige technologieën kan eenvoudig verwijderd worden door de browser. Met de storage API kan data opgeslagen worden op het systeem voor een langere periode . (Mozilla, 2020b)

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Nee

Tabel 2.25: ondersteuning Cache API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Nee

Tabel 2.26: ondersteuning Storage API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Nee

Tabel 2.27: ondersteuning File API

Door gebruik te maken van deze verschillende APIs kan er ook een offline ervaring aangeboden worden aan de gebruiker.

Bestandentoegang

Native applicaties hebben toegang tot het volledige bestandssysteem van het toestel. Er kunnen bestaande bestanden gelezen en aangepast worden. Er kunnen ook nieuwe bestanden aangemaakt en opgeslagen worden.

Door gebruik te maken van de File API (Kruisselbrink, 2020a) heeft een webapplicatie ook toegang tot het bestandssysteem. Bestanden kunnen gelezen worden en metadata over deze bestanden kan verkregen worden.

Een webapplicatie heeft echter enkel leesrechten op deze bestanden. Er kunnen dus geen bestanden geschreven of aangepast worden.

Dit betekent dus dat bepaalde toepassingen die hier gebruik van maken nog steeds een native applicatie nodig hebben.

Contacten

Bepaalde native applicaties hebben toegang nodig tot de contacten van de gebruiker. Een voorbeeld hiervan is WhatsApp. Deze importeert de contacten van het toestel in de applicatie.

De contacten die opgeslagen staan op het systeem van de gebruiker kunnen geïmporteerd worden in een webapplicatie met de Contacts API (Richard Tibbett, Berjon & Song, 2020).

In theorie zouden PWAs hier dus gebruik kunnen van maken maar de ondersteuning is heel beperkt. Het is dus niet aangeraden om een webapplicatie te ontwikkelen die afhankelijk is van de contacten van een gebruiker.

Sms

Native Applicaties kunnen binnenkomende sms-berichten lezen. Dit wordt vaak gebruikt om de authenticatie van een gebruiker sneller te laten verlopen. Een voorbeeld van deze

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Nee	Beperkt	Nee

Tabel 2.28: ondersteuning Contacts API - * Op het moment van schrijven is dit een nieuwe en experimentele functie die enkel werkt op Android 10. Verdere ondersteuning is nog onbekend.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Nee	Beperkt	Nee

Tabel 2.29: ondersteuning Messaging API - * Op het moment van schrijven is dit een nieuwe en experimentele functie die enkel werkt op Android 10. Verdere ondersteuning is nog onbekend.

use-case kan bij de applicatie van het betalingsplatform PayPal gevonden worden. Als een gebruiker zich registreert, zal zijn telefoonnummer gecontroleerd worden door er een sms naar dit nummer te sturen met een code. PayPal zal zien dat er een sms binnenkomt en zal automatisch de code uit dit bericht halen. Op deze manier hoeft de gebruiker de app niet te verlaten.

Native applicaties kunnen niet enkel smsen lezen, ze kunnen er ook schrijven. Dit betekent dus dat elke ontwikkelaar een sms-client applicatie kan maken.

Met de SMS-receiver API (Fullea, Cantera & Kis, 2020) kan er gekeken worden naar inkomende smsen. Het voorbeeld van de PayPal applicatie kan dus ook geïmplementeerd worden als PWA. PWAs hebben echter enkel toegang tot binnenkomende smsen.

Het voorbeeld van PayPal kan ook geïmplementeerd worden aan de hand van een PWA. Applicaties die ook sms-berichten moeten versturen kunnen niet ontwikkeld worden als PWA.

Taakplanning

De Task Sheduler API (Kulkarni, 2020) kan ervoor zorgen dat taken zoals alarmen, herinneringen en gelijkaardige taken kunnen ingepland worden in het systeem. Deze API is slechts een voorstel en heeft dus nog geen ondersteuning.

Een applicatie schrijven die het alarm van een smartphone in de ochtend laat afgaan, of een activiteit in jouw agenda plaatst, is dus niet mogelijk met een PWA. Dit zijn toepassingen die wel mogelijk zijn met native applicaties.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Nee	Nee	Nee

Tabel 2.30: ondersteuning Task Sheduler API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	ja	Ja

Tabel 2.31: ondersteuning touch gebaren

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	ja	Ja

Tabel 2.32: ondersteuning Clipboard API

Input

Touch gebaren

Native applicaties hebben een verwachtingspatroon ontwikkeld bij de gebruiker. Voorbeelden hiervan zijn:

- Swipe van links opent het menu
- Knijpen om in te zoomen

HTML5 voegt aan de reeds bestaande input methodes nu ook touch-controls toe. Dit is belangrijk om een applicatie intuïtief te laten werken. Het is logisch dat Safari op desktop dit niet ondersteunt aangezien Safari enkel kan gedownload worden op Mac-toestellen en geen enkel Mac-toestel een touchscreen heeft.

Deze gebaren kunnen nu ook gebruikt worden in een PWA. Dit zorgt ervoor dat een geïnstalleerde PWA meer zal aanvoelen als een native applicatie.

Klembord toegang

De Clipboard API (Kacmarcik & Lyukshin, 2020) geeft een ontwikkelaar de mogelijkheid om te interageren met het klembord. Er kunnen zowel items van het klembord gelezen worden als dat er items kunnen geschreven worden naar het klembord.

Er worden ook methodes voorzien voor het reageren op de actie waarbij een gebruiker zelf iets kopieert of plakt.

User experience

Offline gebruik

Door het gebruik van serviceworkers kan een website offline gebruikt worden. Deze website moet eerst bezocht worden als de gebruiker online is. De geladen paginas en andere items zoals fotos kunnen opgeslagen worden voor offline gebruik.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	ja	Ja

Tabel 2.33: ondersteuning offline gebruik

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	ja	Ja

Tabel 2.34: ondersteuning achtergrondsynchronisatie

Achtergrondsynchronisatie

Een actie kan van start gaan als de gebruiker een trage verbinding heeft of als hij offline is. Achtergrondsynchronisatie zal ervoor zorgen dat deze actie uitgevoerd wordt vanaf dat er een stabiele internetconnectie is, zelfs al is de applicatie reeds gesloten.

Meer info over achtergrondsynchronisatie kan gevonden worden in het hoofdstuk Functionaliteiten die een serivce worker mogelijk maakt.

Inter-app communicatie

Native applicaties kunnen gebruik maken van deep-linking, dit is een concept waarbij er in een app een link kan staan naar een specifieke pagina op een andere app.

De Web Share API (Giuca, 2020) en de Web Share Target API (Williger & Giuca, 2020) zorgen ervoor dat links van websites kunnen geopend worden in native applicaties.

De web Share API moet gebruikt worden in de applicatie die een link heeft naar een andere applicatie.

De web Share Target API moet gebruikt worden in de applicatie waarnaar gerefereerd wordt. Deze zal ervoor zorgen dat de gebruiker uiteindelijk op de juiste pagina terecht komt.

Betalingen

Aan de hand van de Payment Request API (Denicola, Cáceres, Koch, Jacobs & Solomakhin, 2020) kan heel snel, zonder de website te verlaten, een betaling uitgevoerd worden. Bij Apple-toestellen zal dit gebeuren via Apple-pay.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Nee	ja	Ja

Tabel 2.35: ondersteuning Web Share API en Web Share Target API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Ja	ja	Ja

Tabel 2.36: ondersteuning Payment Request API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	ja	Nee

Tabel 2.37: ondersteuning Credential Management API

Credentials

De Credential Management API (West, 2020) levert methodes voor het ophalen en opslaan van de credentials van een gebruiker. Op deze manier kan de gebruiker eenvoudiger en veiliger aanmelden op een platform.

Locatie en positionering

Geolocatie

De Geolocation API (Popescu, 2020) zorgt ervoor dat een applicatie toegang heeft tot de locatie van een toestel. Dat wordt gedaan op basis van de gps-sensor of op basis van het netwerk. De API voorziet niet enkel de locatie maar ook methodes die de applicatie informeren als de locatie verandert. Navigatie-applicaties zoals Google Maps of Waze kunnen dus ook geïmplementeerd worden als PWA.

Geofencing

Geofencing is een technologie waarbij er een geografische zone wordt ingesteld. Als de gebruiker in deze zone komt, wordt er automatisch een actie uitgevoerd.

Er was een voorstel om deze API (Kruisselbrink, 2020b) uit te werken voor het web, maar op het moment van schrijven heeft geen enkele browser dit geïmplementeerd. Dit is eenfunctie die enkel beschikbaar is voor native IOS en Android-applicaties.

Toesteloriëntatie

Native applicaties en meer specifiek, mobiele games maken vaak gebruik van de toesteloriëntatie als input methode. Dit wordt bijvoorbeeld gebruikt bij racegames om een stuur te simuleren.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	ja	Ja

Tabel 2.38: ondersteuning Geolocation API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Nee	Nee	Nee	Nee

Tabel 2.39: ondersteuning Geofencing API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Ja

Tabel 2.40: ondersteuning Device Orientation API

De Device Orientation API (Rich Tibbett, 2020) levert methodes voor het detecteren van de oriëntatie van het toestel. Er zijn drie eigenschappen die de oriëntatie bepalen:

- Alpha: Dit is de richting naar waar het toestel gericht is.
- Beta: Dit is het aantal graden dat het toestel voorwaarts of achterwaarts gekanteld is.
- Gamma: Dit is het aantal graden dat het toestel naar links of naar rechts gekanteld is.

De toesteloriëntatie kan, net zoals bij native applicaties, gebruikt worden als een inputmethode voor applicaties. Dit wordt vaak gebruikt bij games.

Deze API levert ook methodes die website in een bepaalde oriëntatie forceren.

Niet elk toestel heeft deze sensoren. Als ze aanwezig zijn, worden ze ondersteund in volgende browsers. Deze API is vooral gericht naar mobiele toestellen.

Toestelbeweging

De Generic Sensor API (Waldroon, 2020) is een verzameling van APIs voor het gebruiken van verschillende sensoren van het toestel.

DeviceMotionEvent Levert informatie over de snelheid waarmee een toestel zijn oriëntatie en positie verandert.

Accelerometer Levert informatie over de snelheid waarbij het toestel zich beweegt in een ruimte. De API levert zowel X-, Y- als Z-coördinaten

Gyroscope Levert informatie over de snelheid waarmee een toestel aan het roteren is.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Nee	Ja	Ja

Tabel 2.41: ondersteuning DeviceMotionEvent

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.42: ondersteuning Accelerometer

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.43: ondersteuning Gyroscope

Magnetometer Meet het magnetische veld rond het toestel.

LinearAccelerationSensor Levert informatie over de snelheid waarmee een toestel beweegt, maar dit zonder de impact van de zwaartekracht.

Nabijheid-sensoren

De Proximity API (Kostiainen & Bhaumik, 2020) geeft informatie over de afstand tussen het toestel en een object. Deze sensor wordt gebruikt om het scherm uit te zetten als een persoon aan het bellen is.

Scherm en output

Virtual en augmented reality

De webXR Device API (Jones & Waliczek, 2020) zorgt voor een interface voor het verbinden van een virtual reality toestel met de browser. De sensoren van het toestel kunnen gebruikt worden om het canvas-element te laden in het VR-toestel.

De oudere webVR API levert gelijkaardige methodes maar is beter ondersteund.

Fullscreen

De Fullscreen API (**Kesteren2020**) geeft een aantal methodes die ervoor zorgen dat de browser-elementen verwijderd worden. Dit laat een website meer aanvoelen als een native applicatie.

Er worden twee methodes voorzien: een voor in fullscreen mode te gaan en een om deze te verlaten.

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Nee	Ja	Nee	Nee	Nee

Tabel 2.44: ondersteuning Magnetometer

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.45: ondersteuning LinearAccelerationSensor

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Nee	Ja	Nee	Nee	Nee	Nee

Tabel 2.46: ondersteuning Proximity API

Wake lock

eel toestellen gaan het scherm dimmen na een bepaalde tijd van inactiviteit. Dit kan onhandig zijn voor bepaalde applicaties zoals een gps. Met de Wake Lock API (**Bogdanovich2020**) kan het automatisch dimmen van het scherm tegengegaan worden.

2.2.2 Conclusie

Browsers

Mobiele besturingssystemen

Desktop besturingssystemen

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja (webXR)	Ja (webVR)	Ja (webVR + webXR	Nee	Ja (webVR + webXR)	Nee Nee

Tabel 2.47: ondersteuning WebVR en WebXR

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Ja	Ja	Ja	Ja	Nee

Tabel 2.48: ondersteuning Fullscreen API

Edge	Firefox	Chrome	Safari	Android (Chrome)	IOS (Safari)
Ja	Nee	Ja	Onbekend	Ja	Nee

Tabel 2.49: ondersteuning Wake Lock API

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.