



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 - 2020

Tweede examenperiode

Woord vooraf

Samenvatting

Inhoudsopgave

1	Inleiding	17
1.1	Probleemstelling	17
1.2	Onderzoeksvraag	18
1.3	Onderzoeksdoelstelling	18
1.4	Opzet van deze bachelorproef	18
2	Literatuurstudie	19
2.1	Wat is een PWA	19
2.1.1	Service workers	20
2.1.2	A2HS	25
2.1.3	Application shell	26
2.1.4	Progressive enhancement	27
2.1.5	Application manifest	27

2.1.6	Geschiedenis van PWA's	28
2.1.7	Voorbeelden van PWA's	29
2.2	Besturingssystemen en PWAs	32
2.2.1	Onderzoek	33
2.2.2	concluderende tabel	50
2.2.3	Conclusie	51
2.3	Waarom een PWA	55
2.3.1	Bereik	55
2.3.2	Platformonafhankelijkheid	55
2.3.3	Omzet	56
2.3.4	Bundle size	56
2.3.5	Offline gebruik	58
2.3.6	Betrokkenheid	58
2.3.7	Kost	59
2.3.8	Deployment	60
2.3.9	Updates	61
2.3.10	Conclusie	61
2.4	Beperkingen van een PWA	61
2.4.1	Cameragebruik	62
2.4.2	Controle over platformen	62
2.4.3	Functies van een besturingssysteem	62
2.4.4	Browserondersteuning	63
2.4.5	Aanwezigheid in app-stores	63
2.4.6	Conclusie	63

2.5	Tools voor het ontwikkelen van en een PWA	63
2.5.1	Lighthouse	63
2.5.2	Workbox	65
2.5.3	PWAbuilder	66
2.5.4	Chrome developer tools	66
2.6	PWA-alternatieven	66
2.6.1	Cross platform hybrid development	67
2.6.2	Cross platform native development	69
2.6.3	Conclusie	72
3	Methodologie	75
4	Conclusie	77
A	Onderzoeksvoorstel	79
A.1	Introductie	79
A.2	State-of-the-art	80
A.2.1	Wat is een PWA	80
A.2.2	Welke functies van een besturingssysteem kan een PWA gebruiken . . .	80
A.2.3	Waarom PWA's	81
A.2.4	Beveiliging	81
A.2.5	Native containers	82
A.2.6	Appllication shell architecture	82
A.3	Methodologie	82
A.3.1	Technologieën	82
A.4	Verwachte resultaten	83

Lijst van figuren

2.1	voorstelling van wat is een PWA (Richard & LePage, 2020)	20
2.2	demonstratie van achtergrond synchronisatie bij Google Chrome op Android - pagina offline	24
2.3	demonstratie van achtergrond synchronisatie bij Google Chrome op Android - melding als gebruiker terug online is	24
2.4	schema levenscyclus van een service worker (Gaunt, 2019)	25
2.5	gedrag van Google Chrome op Windows 10 op het beforeinstallprompt	26
2.6	gedrag van Google Chrome op Android op het beforeinstallprompt	26
2.7	voorbeeld van een application shell architectuur (Osmani & Gaunt, 2015)	27
2.8	Outlook op Mac	30
2.9	Outlook op Mac als PWA	30
2.10	Screenshot van het installatieproces op een iPhone met iOS 13	52
2.11	Screenshot van het installatieproces op Android 10	53
2.12	Screenshot van de zoekresultaten van Outlook op Windows 10	54
2.13	Screenshot van de zoekresultaten van Outlook op Mac Os	54
2.14	Android native applicatie versie 11.8.1	57

2.15	PWA-versie 118	58
2.16	Screenshot van het PWA-onderdeel van een lighthouse audit op de site dart501.netlify.com	65

Lijst van tabellen

2.1	beschrijving notifications API	23
2.2	ondersteuning van de Media Capture API	34
2.3	ondersteuning van de Image Capture API	34
2.4	ondersteuning van de Image Capture API	35
2.5	ondersteuning van WebRTC	36
2.6	ondersteuning Apple AirPlay	36
2.7	ondersteuning van Chrome Sender API	36
2.8	ondersteuning van Media Session API	37
2.9	ondersteuning van Web Bluetooth API	37
2.10	ondersteuning van Web USB API	38
2.11	ondersteuning van Web NFC API	38
2.12	ondersteuning van Network information API	38
2.13	ondersteuning online status	38
2.14	ondersteuning vibratiemotor	39
2.15	ondersteuning batterijstatus	39
2.16	ondersteuning toestelgeheugen	39
2.17	ondersteuning lokale notificaties	40

2.18	ondersteuning push notificaties	40
2.19	ondersteuning A2HS	41
2.20	ondersteuning Badging API	41
2.21	ondersteuning voorgrond detectie	41
2.22	ondersteuning Permissions API	42
2.23	ondersteuning web storage	42
2.24	ondersteuning IndexedDB	42
2.25	ondersteuning Cache API	43
2.26	ondersteuning Storage API	43
2.27	ondersteuning File API	43
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.	44
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.	44
2.30	ondersteuning Task Sheduler API	45
2.31	ondersteuning touch gebaren	45
2.32	ondersteuning Clipboard API	45
2.33	ondersteuning offline gebruik	46
2.34	ondersteuning achtergrondsynchronisatie	46
2.35	ondersteuning Web Share API en Web Share Target API	46
2.36	ondersteuning Payment Request API	47
2.37	ondersteuning Credential Management API	47
2.38	ondersteuning Geolocation API	47
2.39	ondersteuning Geofencing API	48
2.40	ondersteuning Device Orientation API	48
2.41	ondersteuning DeviceMotionEvent	48
2.42	ondersteuning Accelerometer	49
2.43	ondersteuning Gyroscope	49
2.44	ondersteuning Magnetometer	49
2.45	ondersteuning LinearAccelerationSensor	49
2.46	ondersteuning Proximity API	49

LIJST VAN TABELLEN	15
2.47 ondersteuning WebVR en WebXR	50
2.48 ondersteuning Fullscreen API	50
2.49 ondersteuning Wake Lock API	50
2.50 concluderende tabel 'waarom een PWA'	61
2.51 concluderende tabel 'beperkingen van een PWA'	63

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 (**Polleffiet2011**):

- 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 afgeleid 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 specificeren 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

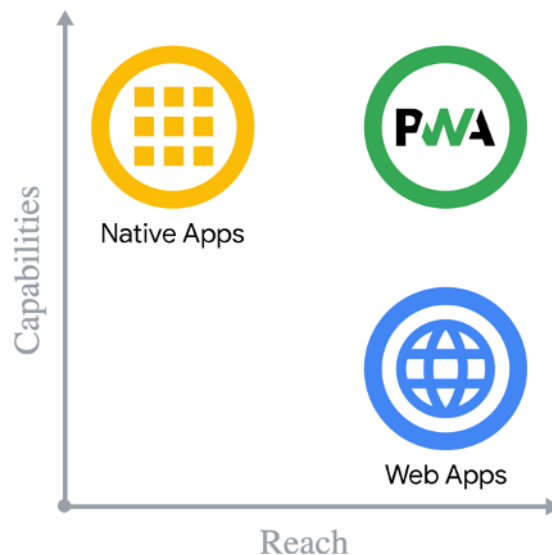
Het web is een platform waar applicaties kunnen gepubliceerd worden zonder afhankelijk te zijn van een overkoepelend bedrijf of organisatie. Voor een website is er slechts één codebase en de laatste versie is steeds beschikbaar voor de gebruiker. Dit allemaal zorgt ervoor dat een webapplicatie iedereen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



Figuur 2.1: 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 service 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 pagina's 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 & awwards, 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, 2019a)

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

```
1 function displayNotification() {  
2   if (Notification.permission == 'granted') {  
3     navigator.serviceWorker.getRegistration().then(function(reg) {  
4       var options = {  
5         body: 'Here is a notification body!',  
6         icon: 'images/example.png',  
7         vibrate: [100, 50, 100],  
8         data: {  
9           userId: 383209489398274  
10        },  
11        actions: [  
12          {action: 'explore', title: 'Explore this new world',  
13            icon: 'images/checkmark.png'},  
14          {action: 'close', title: 'Close notification',  
15            icon: 'images/xmark.png'},  
16        ]  
17      };  
18      reg.showNotification('Hello world!', options);  
19    });  
20  }  
21 }
```

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, 2019a) (Mozilla, 2019c)

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 zal
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, 2019d) (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

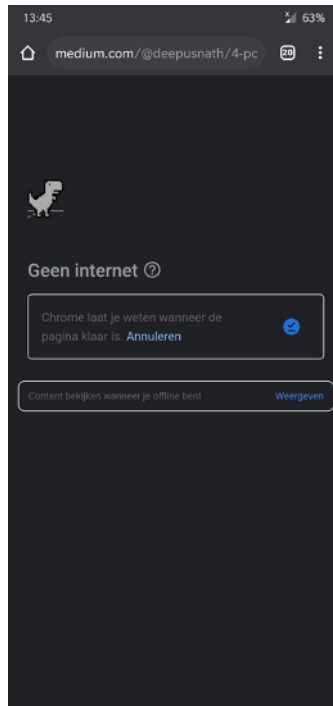
De levenscyclus van de service worker is onafhankelijk van de levenscyclus van de webapplicatie. Als de webapplicatie gesloten wordt, blijft de service worker gewoon werken.

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.

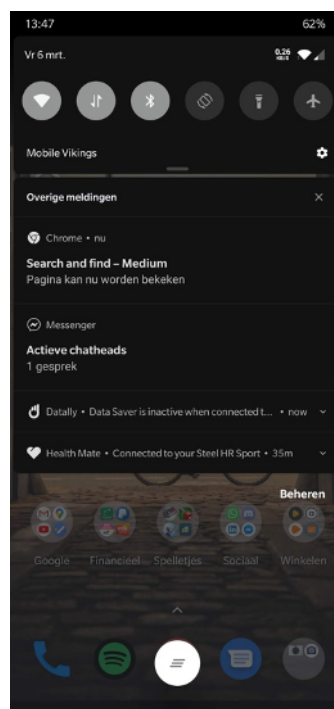
```

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

```



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



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

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.

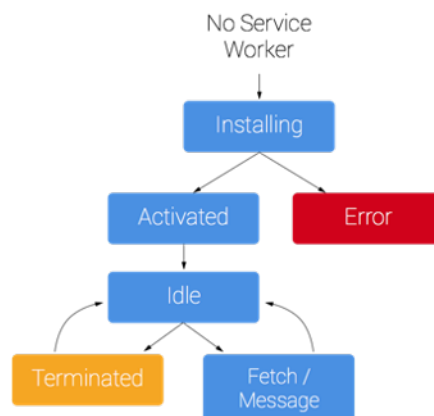
Tijdens de activatiefase wordt er bekeken welke gecachte gegevens geüpdatet moeten worden en welke niet. De service worker zal de bestanden die het ontvangen heeft van het eerste netwerkverzoek vergelijken met zijn huidige cachegeheugen. Als er verschillen zijn zal dit cachegeheugen aangepast worden.

Is 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)



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

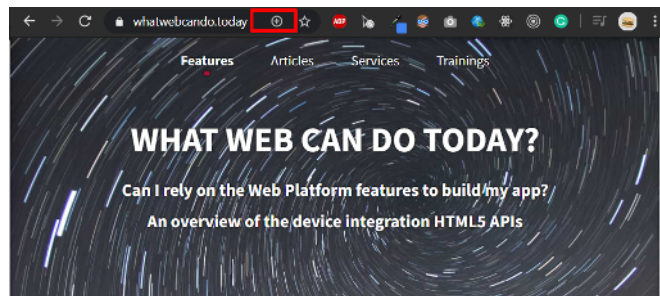
2.1.2 A2HS

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.

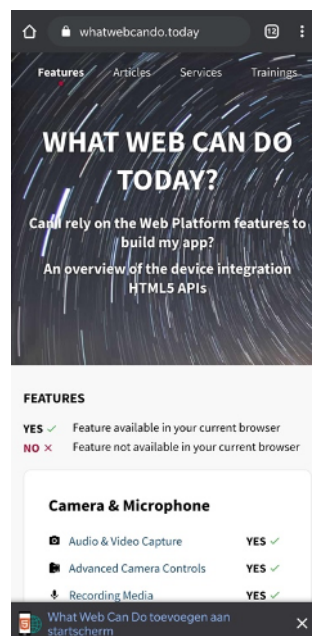
Een website moet voldoen aan volgende criteria:

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

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



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

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 worden dat deze webapplicatie geïnstalleerd kan worden. Er kan ook getoond worden hoe dit moet gebeuren.

(PWAbuilder, 2020)

2.1.3 Application shell

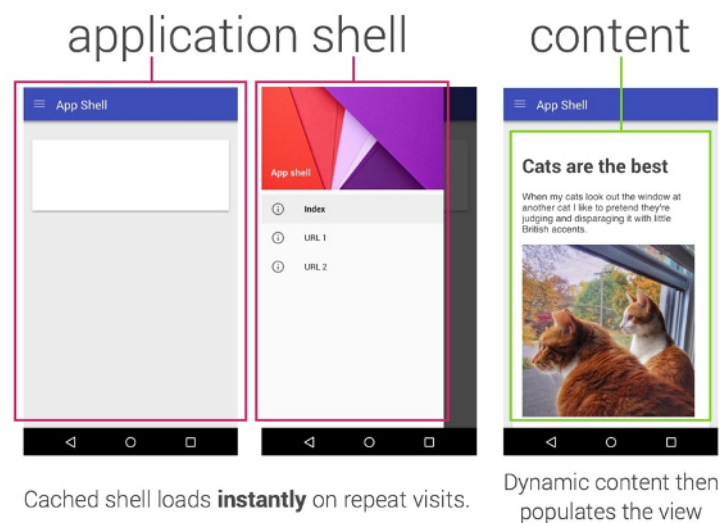
De application shell of app shell is de minimale HTML, CSS en JavaScript die nodig is om een userinterface te tonen op een toestel. Dit is vaak de header, footer en de navigatie.

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)



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

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.

```

1 {
2   "short_name": "Maps",
3   "name": "Google Maps",
4   "icons": [
5     {
6       "src": "/images/icons-192.png",
7       "type": "image/png",
8       "sizes": "192x192"
9     },
10    {
11      "src": "/images/icons-512.png",
12      "type": "image/png",
13      "sizes": "512x512"
14    }
15  ],
16  "start_url": "/maps/?source=pwa",
17  "background_color": "#3367D6",
18  "display": "standalone",
19  "scope": "/maps/",
20  "theme_color": "#3367D6"
21 }
```

***** 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 eerste 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 volgende jaar de app-store lanceerden. (Silver, 2018)

Chrome 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

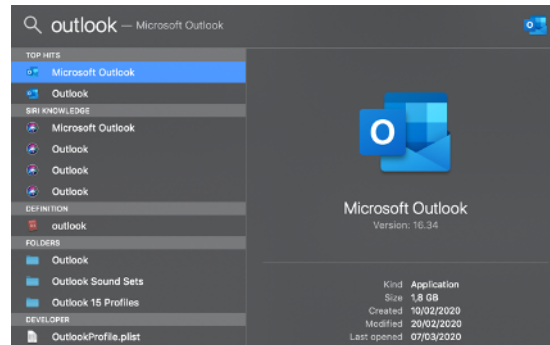
Op 19 september 2019 stelde Apple IOS 13 voor. Deze software-update zorgde ervoor dat de iPhone gebruik kon maken van een service worker. De applicaties kunnen nu ook geïnstalleerd worden op het toestel, de gebruiker moet dit wel nog zelf doen aan de hand van het menu. (Apple, 2020e)

2.1.7 Voorbeelden van PWA's

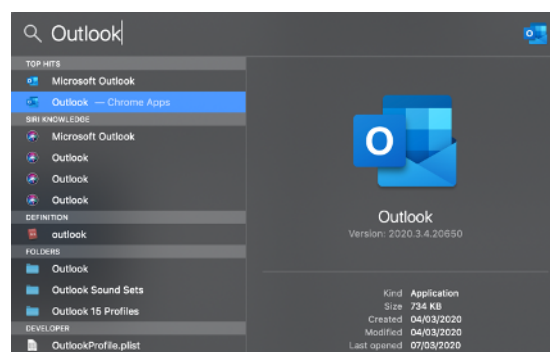
Outlook

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

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.



Figuur 2.8: Outlook op Mac



Figuur 2.9: Outlook op Mac als PWA

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 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](https://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, 2019b)

Als er meer informatie nodig was over de web APIs werd deze gevonden op [developers.Google.com](https://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

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.

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

Tabel 2.2: ondersteuning van de Media Capture API

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 kwaliteit 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)

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

Tabel 2.3: ondersteuning van de Image Capture API

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 *****

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

Tabel 2.4: ondersteuning van de Image Capture API

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>
```

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, Boströ & 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.

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

Tabel 2.5: ondersteuning van WebRTC

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.

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

Tabel 2.8: ondersteuning van Media Session API

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.

Met de Web Bluetooth API (Grant & Ruiz-Henríquez, 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)

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

Tabel 2.9: ondersteuning van Web Bluetooth API

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)

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

Tabel 2.10: ondersteuning van Web USB API

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)

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

Tabel 2.11: ondersteuning van Web NFC API

Dit is een functie met veel mogelijkheden die helaas niet beschikbaar is voor webapplicaties. 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.

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

Tabel 2.12: ondersteuning van Network information API

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.

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

Tabel 2.13: ondersteuning online status

Vibratiemotor

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

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

Tabel 2.14: ondersteuning vibratiemotor

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:

- 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.

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

Tabel 2.15: ondersteuning batterijstatus

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.

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

Tabel 2.16: ondersteuning toestelgeheugen

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 service 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

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, Fulla & 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 service worker mogelijk maakt.

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

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

Tabel 2.18: ondersteuning push notificaties

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.

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

Tabel 2.19: ondersteuning A2HS

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 Badging 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.

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

Tabel 2.21: ondersteuning voorgrond detectie

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
- Gyroscop
- Magneetsensor
- Betalingen

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

Tabel 2.22: ondersteuning Permissions API

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)

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

Tabel 2.23: ondersteuning web storage

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)

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

Tabel 2.24: ondersteuning IndexedDB

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)

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

Tabel 2.25: ondersteuning Cache API

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 2.26: ondersteuning Storage 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.

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

Tabel 2.27: ondersteuning File API

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.

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.

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 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.

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.

Taakplanning

De Task Sheduler API (Kulkarni, 2020) kan ervoor zorgen dat taken zoals alarmeren, 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

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.

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

Tabel 2.31: ondersteuning touch gebaren

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.

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

Tabel 2.32: ondersteuning Clipboard API

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

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 service worker mogelijk maakt.

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

Tabel 2.34: ondersteuning achtergrondsynchronisatie

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.

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

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)
Ja	Nee	Ja	Ja	ja	Ja

Tabel 2.36: ondersteuning Payment Request 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.

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

Tabel 2.37: ondersteuning Credential Management API

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.

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

Tabel 2.38: ondersteuning Geolocation API

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 een functie die enkel beschikbaar is voor native IOS en Android-applicaties.

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

Tabel 2.39: ondersteuning Geofencing API

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.

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.

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

Tabel 2.40: ondersteuning Device Orientation API

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.

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

Tabel 2.41: ondersteuning DeviceMotionEvent

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

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

Tabel 2.42: ondersteuning Accelerometer

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

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.

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

Tabel 2.44: ondersteuning Magnetometer

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

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

Tabel 2.45: ondersteuning LinearAccelerationSensor

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.

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

Tabel 2.46: ondersteuning Proximity API

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.

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

Fullscreen

De Fullscreen API (van Kesteren & Çelik, 2020) 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)
Ja	Ja	Ja	Ja	Ja	Nee

Tabel 2.48: ondersteuning Fullscreen API

Wake lock

De Wake Lock API (Bogdanovich, 2020) zorgt ervoor dat het toestel het scherm dimmen na een bepaalde tijd van inactiviteit. Dit kan onhandig zijn voor bepaalde applicaties zoals een gps. Met de Wake Lock API (Bogdanovich,

Logvinov & Caseres, 2020) kan het automatisch dimmen van het scherm tegengegaan worden.

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

Tabel 2.49: ondersteuning Wake Lock API

2.2.2 concluderende tabel

***** tabel invoegen *****

2.2.3 Conclusie

Het is opvallend hoeveel functies beschikbaar zijn voor het web. Slechts een beperkt aantal native applicaties zouden niet met web-technologieën gemaakt kunnen worden.

Het probleem is echter consistentie. Het voordeel van het web is dat je één codebase hebt en dat deze applicatie op verschillende soorten toestellen werkt. Dit is voor een basisapplicatie het geval, maar als er specifieke functies gebruikt moeten worden, wordt het moeilijker. Verschillende browsers verwachten verschillende web-APIs. Veel van de APIs bestaan en kunnen gebruikt worden, maar zullen niet op alle browsers werken.

Deze technologieën kunnen dus gebruikt worden om de functionaliteit van een applicatie te ondersteunen. Maar een applicatie zou niet mogen afhangen van de functionaliteiten aangezien niet alle gebruikers de toepassing kunnen gebruiken.

Browsers

We kunnen concluderen dat de browsers die Google maakt (Google Chrome, Google Chrome for Android) een betere ondersteuning geeft dan de browsers die Apple maakt (Safari, Safari IOS). Deze trend is zowel te zien bij de mobiele browsers als de browsers voor desktop.

De nieuwste versie van Microsoft Edge ondersteunt veel functies. Dit komt omdat deze laatste versie (v80.0) gebaseerd is op de open source browser Chromium die ontwikkeld is door Google. Ook Google Chrome is op Chromium gebaseerd.

Het is opvallend dat bij de desktops van Apple de browser de limiterende factor is en niet het besturingssysteem. Als Google Chrome gedownload wordt op een Apple-toestel zijn deze functionaliteiten wel beschikbaar voor dit toestel.



Figuur 2.10: Screenshot van het installatieproces op een iPhone met IOS 13

Mobiele besturingssystemen

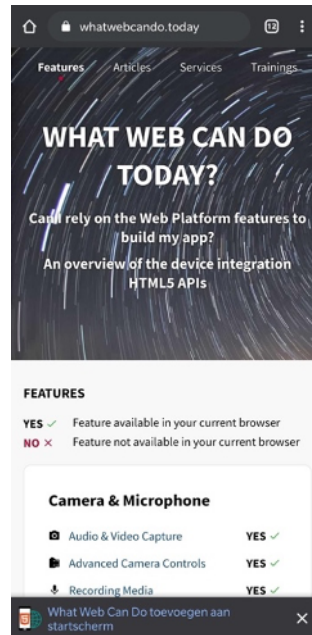
Net zoals bij browsers, is de ondersteuning voor PWAs op toestellen en systemen die ontwikkeld zijn door Google, beter dan de toestellen die Apple ontwikkelde.

Toestellen die Android gebruiken als besturingssysteem kunnen meer genieten van de functionaliteiten die het web te bieden heeft. De grootste limitatie die IOS heeft in vergelijking met Android is het niet ondersteunen van push-notificaties. Dit is een functie die belangrijk is om gebruikers betrokken te houden op een platform. Push notificaties zijn voor e-commerce platformen vaak een heel belangrijke tool binnen hun marketingplan. (Anastasia, 2017)

Achtergrondsynchronisatie is een functie die de gebruikerservaring kan verbeteren. Deze functie die mogelijk gemaakt wordt door service workers is ook niet beschikbaar op een IOS-toestel.

Apple laat PWAs toe om slechts 50MB aan data offline op te slaan. Als deze applicatie voor een bepaalde tijd niet gebruikt wordt, zal al deze data verwijderd worden om ruimte op het toestel te besparen. Bij Android is dit 6% van de beschikbare opslagruimte en zullen de opgeslagen gegevens niet automatisch verwijderd worden.

Ook A2HS-ervaring is op een Android toestel beter. Als een web-applicatie voldoet aan de normen om geïnstalleerd te worden zal Google Chrome op Android de gebruiker voorstellen om deze applicatie toe te voegen aan het startscherm. Deze functionaliteit is ook beschikbaar voor Safari op een iPhone maar hier moet de gebruiker zelf actie ondernemen en naar de instellingen van de website gaan om de knop 'zet op beginscherm' te vinden.



Figuur 2.11: Screenshot van het installatieproces op Android 10

De integratie van de ingebouwde slimme assistent is op IOS beperkter dan op Android. Als een applicatie geïnstalleerd is op een Android toestel, kan Google assistant deze openen aan de hand van een stemcommando. Siri heeft deze functionaliteit niet. (Lathiya, 2020)

Desktop besturingssystemen

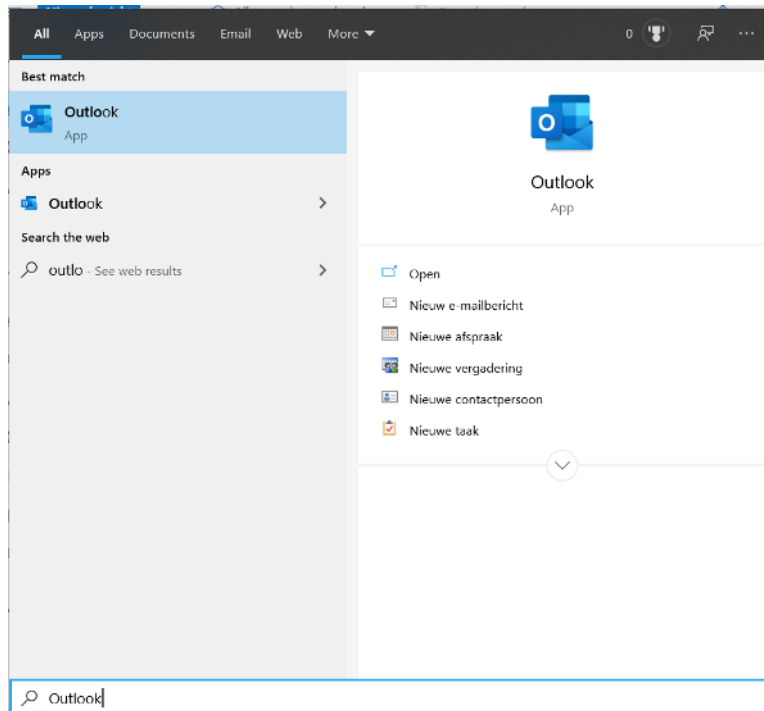
Bij deze vergelijking zullen enkel Windows en Mac OS in beschouwing genomen worden. Andere besturingssysteem hebben relatief gezien niet veel gebruikers. 88,14% van de computers werkt op Windows en 9,42% werkt op Mac OS. Slechts 2,44% van de computers maakt gebruik van andere besturingssystemen. (netMarketShare, 2020)

Microsoft wil PWAs zo goed mogelijk ondersteunen. Ze zijn zelf ook PWAs aan het ontwikkelen. Een voorbeeld hiervan is Outlook, de populaire online email-client kan nu ook geïnstalleerd worden als PWA. (Microsoft, 2020b)

PWAs kunnen ook zonder aanpassingen in de Windows store geplaatst worden. Deze applicatie kan dan door alle Windows-toestellen gedownload worden. Er zijn verschillende toestellen die gebruik maken van deze store:

- Windows 10 toestellen
- Windows S toestellen
- XBox
- Microsoft Hololens

Windows gaat nog een stap verder dan dit. Het gaat zelf op zoek naar PWAs op het internet en zal deze automatisch toevoegen aan de Windows store. Dit kan wel tegengegaan



Figuur 2.12: Screenshot van de zoekresultaten van Outlook op Windows 10

worden als de uitgever van de PWA dit niet wil. (Gustafson & Burtoft, 2017a) (Gustafson & Burtoft, 2017b)

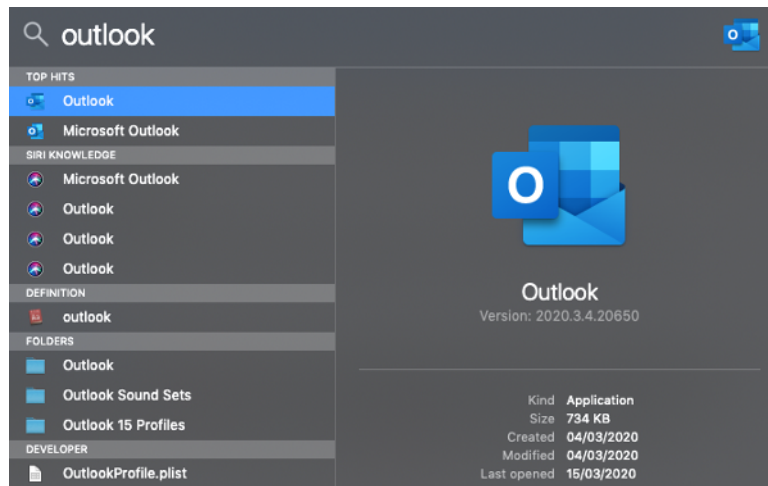
Als een PWA geïnstalleerd wordt op Windows zal deze zich gedragen als een volwaardig programma. De PWA zal opgestart worden in zijn eigen venster. De applicatie kan ook gevonden worden met een zoekopdracht vanuit het startmenu. De applicatie kan ook toegevoegd worden aan de taakbalk en aan het bureaublad.

De tool PWABuilder die gebruikt kan worden om een PWA in de populaire mobiele app-stores te krijgen, is ook ontwikkeld door Microsoft. (PWABuilder, 2020)

Microsoft heeft ook een uitgebreide documentatie die ontwikkelaars helpen bij het ontwikkelen van PWAs. (Microsoft, 2020a)

Apple biedt weinig ondersteuning voor PWAs in zijn mobiele besturingssystemen. Deze trend zet zich door op de besturingssystemen voor desktops. In de standaardbrowser, Safari, van de toestellen kan een PWA niet geïnstalleerd worden. Als de gebruiker Google Chrome gebruikt, kan een PWA wel geïnstalleerd worden. Als deze geïnstalleerd is, kan de applicatie ook gevonden worden in een spotlight search. De applicatie kan ook vastgeplakt worden aan het dock.

Deze algemene trend waarbij Apple PWAs bewust niet ondersteunt, lijkt niet snel te veranderen. John Wilander, een ontwikkelaar binnen Apple, bekritiseerde openlijk PWAs als een technologie die van Google is en dat dit niet in de planning zit van Apple. (Wilander, 2019)



Figuur 2.13: Screenshot van de zoekresultaten van Outlook op Mac Os

2.3 Waarom een PWA

In dit onderdeel van de literatuurstudie zal er onderzocht worden wat de redenen zijn om wel een PWA te ontwikkelen voor een project. De voordelen zullen bekeken worden in vergelijking met traditionele webapplicaties en native applicaties. (Tandel Sunil & Jamadar, 2018)

2.3.1 Bereik

Volgens Google heeft Google Chrome meer dan 1 miljard mobiele gebruikers. In 2016 was dit nog maar 400 miljoen. Steeds meer mensen gebruiken hun smartphone om op zoek te gaan naar informatie.

(Nath, 2017)

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

De acquisitie is bij het web ook groter dan bij native applicaties. De gemiddelde gebruiker download 0 nieuwe applicaties per maand.

Op het web, waar slechts 13% van de tijd van de mobiele gebruikers besteed wordt, worden er echter maandelijks ongeveer 100 websites bezocht. Hier is er voor bedrijven dus een kans om een gebruiker jouw platform te laten ontdekken en nieuwe gebruikers te winnen.

(developers, 2017)

2.3.2 Platformonafhankelijkheid

Een van de grootste voordelen van het web is dat het platformonafhankelijk is. Een webapplicatie hoeft maar eenmaal ontwikkeld te worden en kan dan op meerdere platformen gebruikt worden. Het web is ook niet gebonden aan deze conventionele platformen. De dag van vandaag zijn meer en meer toestellen zoals televisies, game-consoles en e-readers verbonden met het internet.

PWAs kunnen meer geavanceerde functionaliteiten implementeren dan een traditionele webapplicatie. Bepaalde functionaliteiten zijn wel platformafhankelijk. Een functionaliteit die kan toegevoegd worden is push-notifications. Zoals aangetoond in vorig hoofdstuk zal dit niet werken op IOS-toestellen. De PWA zal wel nog werken op IOS maar zal niet genieten van deze functionaliteit. PWAs zijn dus gedeeltelijk platformonafhankelijk.

Bij native ontwikkeling zijn er vaak verschillende codebases voor elk platform die het ondersteunt. Er zijn technologieën die dit proberen op te lossen zoals React Native. Maar ook hier moet er voor bepaalde delen nog native code geschreven worden per platform. Native applicaties zijn dus niet platformonafhankelijk.

2.3.3 Omzet

Conversion rate is een meeteenheid die gebruikt wordt om de omzet van een website te peilen. Hoe hoger de conversion rate hoe beter. De conversion rate wordt bepaald door het aantal conversions die een website heeft ten opzichte van het aantal bezoekers. Een conversion kan voor elke website anders gedefinieerd worden. Voor een e-commerce website zal dit vaak een aankoop zijn. Voor andere websites kan dit het aanmelden van de gebruiker op de nieuwsbrief zijn. (Support, 2020)

Nikkei is een nieuws-website die in 2018 zijn website ombouwde tot een PWA. Door het gebruikmaken van serviceworkers konden ze de laadtijd van hun website drastisch verlagen (14 seconden sneller van +- 20 seconden naar +-6 seconden). Dit had als gevolg dat gebruikers steeds vaker naar Nikkei gingen als nieuwsbron.

De conversion rate steeg met 58% (premium abonnement) en er was een stijging van 49% in het aantal dagelijkse gebruikers. Deze gebruikers lazen gemiddeld het dubbel aantal artikelen dan voordien. (Developers, 2018)

Dit voorbeeld toont aan dat de omzet met een PWA hoger kan zijn dan bij traditionele webapplicaties.

Met native applicaties kan er nog steeds een meer gepersonaliseerde ervaring geboden worden en zal er gemiddeld gezien ook een hogere conversion rate zijn. Maar PWAs scoren beter dan traditionele websites. (Anastasia, 2019)

2.3.4 Bundle size

bundle size is de grootte van de applicatie als deze geïnstalleerd wordt. Het is positief om een zo klein mogelijke zoals televisies, game consoles en e-readers. bundle size te hebben. (Scott, 2019)

Tinder besliste om zijn service ook aan te bieden als een PWA. Tinder slaagde erin om alle functionaliteiten die hun native applicaties hebben over te nemen in de PWA. Ze slaagden hierin door gebruik te maken van verschillende web APIs. (Osmani, 2017)

Een van de grootste voordelen is dat de PWA (versie 118) op het moment van schrijven slechts een grootte heeft van 397kb. De native Androidapplicatie (versie 11.8.1) daarentegen neemt 130mb in beslag op een toestel.

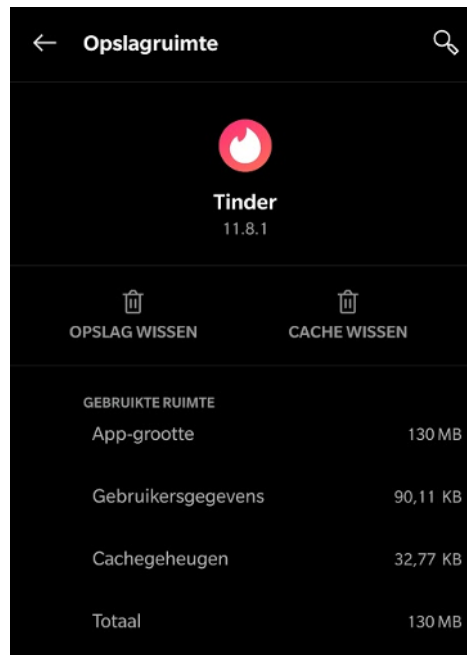
In Westerse landen is dit handig, maar zal het zelden bepalen of een gebruiker een applicatie effectief gebruikt. In andere markten zoals Afrika is dit heel belangrijk. De toestellen die er gebruikt worden zijn vaak verouderd en hebben mindere specificaties dan dat de toestellen gebruikt in het Westen.

De gemiddelde prijs van de top 5 meest verkochte smartphones in Afrika was 135,6 USD. Deze toestellen beschikken vaak over slechts 8 of 16GB opslagruimte. (netAdmin, 2017)

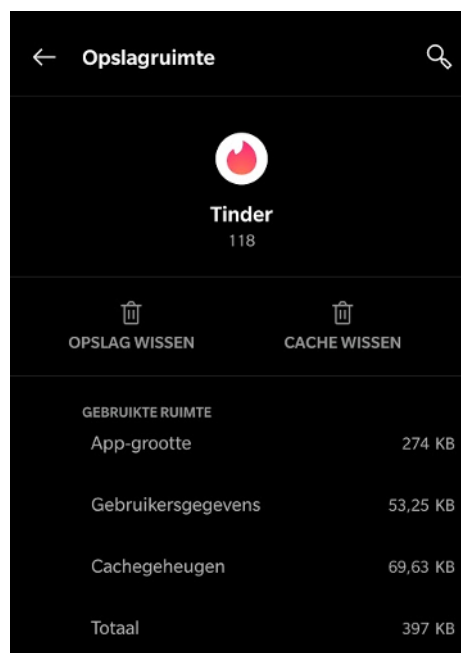
Slechts 7% van Afrika beschikt over een 4g connectie. De andere gebieden moeten het vaak doen met een tragere 3g connectie.

***** BRON TOEVOEGEN - ZIE WORD DOCUMENT - SERVER WAS DOWN

Hier is het dus zeer handig en belangrijk om zo klein mogelijke applicaties te leveren.



Figuur 2.14: Android native applicatie versie 11.8.1



Figuur 2.15: PWA-versie 118

2.3.5 Offline gebruik

Een webapplicatie kan nu geïmplementeerd worden met een offline-first benadering. Offline first is gelijkaardig aan progressive enhancement. Eerst wordt er een applicatie gebouwd die volledig offline beschikbaar is, die vervolgens uitgebreid met online functiona-

liteiten.

Door gebruik te maken van de fetch API in een service worker kunnen API-calls onderschept worden. De service worker kan vervolgens controleren als de gebruiker online is. Als dit niet het geval is, kan de service worker de API-call annuleren en zelf een antwoord sturen met een 200-status en een melding dat er geen internet is. Op basis hiervan kan er een gepaste boodschap getoond worden aan de gebruiker en zal de applicatie niet crashen. (Developers, 2019b)

Data van API-calls kan ook lokaal opgeslagen worden. Als een bepaalde API-call herhaald wordt, kan in plaats van de backend aan te spreken de lokale data gebruikt worden. Dit zal de ervaring voor gebruikers met een zwakke of inconsistente netwerk-connectie verbeteren. (Vanhala, 2017)

2.3.6 Betrokkenheid

Een van de redenen om een native applicatie te maken, was vaak betrokkenheid. Via het web heb je een groot bereik, maar gebruikers op een bepaald platform houden was moeilijk met de

(Google, Microsoft & Awwwards, 2019)

Push notifications zijn meldingen die op het toestel van een gebruiker tevoorschijn komen. Deze worden verstuurd van een server en zijn dus niet afhankelijk van input van een gebruiker. Door gebruik te maken van de Push API en de Notifications API kunnen deze meldingen gebruikt worden bij een PWA.

Push notifications kunnen gebruikers herinneren om een bepaalde applicatie terug te gebruiken.

Push notifications kunnen ook gebruikt worden om de conversion rate te verhogen. Bijvoorbeeld bij een e-commerce platform kan er een melding gestuurd worden als de klant items in zijn mandje heeft geplaatst maar deze nog niet heeft besteld.

(Gaunt, 2020) (Hiltunen, 2018)

Als een webapplicatie voldoet aan de criteria (zie hoofdstuk wat is een PWA) van een PWA, kan deze toegevoegd worden aan het startscherm van een toestel. Hierdoor wordt het voor de gebruiker gemakkelijker om een PWA herhaaldelijk te gebruiken. Als een applicatie wordt geopend vanaf het startscherm, kan deze het volledige scherm gebruiken en zullen adresbalk en andere tools van de browser weggelaten worden. Dit zorgt ervoor dat de PWA meer zal aanvoelen als een native applicatie dan een website.

Door deze nieuwe functionaliteiten toe te voegen, is de betrokkenheid van een PWA groter dan die van een traditionele webapplicatie. Nog niet alle functies die beschikbaar zijn voor native applicaties zijn beschikbaar voor PWAs.

2.3.7 Kost

Het ontwikkelen van native applicaties is niet goedkoop. Services zoals Spotify moeten native applicaties ontwikkelen voor verschillende platformen (MAC OS, Windows, IOS, Android, Linux, ChromeOS).

Het creëren van een digitaal product voor verschillende platformen is vaak te duur voor een kleiner of startend bedrijf. Progressive web apps kunnen hier een oplossing bieden. Een PWA kan geïnstalleerd worden op al de bovenstaande platformen. In plaats van eenzelfde applicatie te bouwen voor verschillende platformen, kan er met PWAs dus 1 applicatie geschreven worden die op verschillende platformen kan gebruikt worden.

Spotify implementeerde zijn service ook als een PWA. (Spotify, 2020) Al de functies die op de native applicaties beschikbaar zijn, zijn ook beschikbaar via deze webapplicatie. Enkel de offline functie is (nog) niet geïmplementeerd. Ook het gebruik van de functietoetsen op het toetsenbord van een desktop is niet ondersteund.

(Vu, 2019)

2.3.8 Deployment

Het uitgeven van een PWA is ook een stuk gemakkelijker en goedkoper dan het verdelen van native applicaties door middel van de verschillende app-stores. Om een applicatie te publiceren in de Apple app-store moet de ontwikkelaar een Apple developer account hebben. Dit kost 99 euro per jaar. (Apple, 2020d)

Ook het publiceren van een applicatie op de Google play store is niet gratis. De uitgever moet een developer account hebben bij Google play. Dit kost eenmalig 25 USD. Elke app-store (Windows, Apple, Android, Chrome,) heeft zijn eigen specifieke eisen en kosten om een applicatie te kunnen uploaden. Bij een PWA hoeft er slechts één applicatie online gezet te worden. Deze moet niet gevalideerd worden door andere partijen. (GooglePlay, 2020b)

Het publiceren van een applicatie op de Windows store is gratis.

Het publiceren van een PWA is meestal ook niet gratis. Het online brengen van een website heeft meestal 2 kosten: enerzijds aankoop van een domein en anderzijds de hosting van een applicatie met https-verbinding. Deze kosten zijn ook nodig bij traditionele webapplicaties. Een domein aankopen via de website versio.nl kost 28,95 euro voor vijf jaar. (Versio, 2020)

Het hosten van een statische applicatie kan via bepaalde platformen kosteloos gebeuren. Voorbeelden van deze platformen zijn Firebase (Firebase, 2020) en Netlify (Netlify, 2020). Deze platformen bieden ook gratis SSL-certificaten aan zodat er een HTTPS-connectie is.

De kost om een PWA online te brengen is dus even hoog als bij traditionele websites. De

kost is wel lager dan bij native applicaties.

De kost is niet het enige voordeel van het online brengen van PWAs ten opzichte van native applicaties. Als een applicatie gepubliceerd wordt in de Apple App store of in de Google Play Store dan moeten deze voldoen aan een groot aantal richtlijnen. Bij het uploaden van een applicatie wordt deze eerst gecontroleerd door Apple en Google of deze wel voldoet aan deze richtlijnen. (Apple, 2020b) (GooglePlay, 2020a)

Dit betekent dus dat niet elke applicatie gepubliceerd kan worden in deze app-stores. Een PWA moet niet gevalideerd worden door een overkoepelend bedrijf. Elke applicatie die de ontwikkelaar maakt kan dus gepubliceerd worden. Een ander gevolg is dat het gemiddeld 72 uur duurt om een app te laten valideren door beide app-stores. Dit is slechts een gemiddelde, want dit kan uitlopen tot een week en langer. (Siddiqui, 2019)

Dit is een vertraging die webapplicaties en PWAs niet hebben.

2.3.9 Updates

Als een PWA bezocht wordt, zal deze steeds de meeste recente versie tonen. Als er een update moet gebeuren, kan de ontwikkelaar kiezen om dit wel of niet aan de gebruiker te laten weten. (Hume, 2018)

Kleinere updates die de gebruiker niet zal opmerken, kunnen uitgevoerd worden zonder dat de gebruiker dit weet. (Sanderson, 2020)

Als er grotere updates zijn, is het een betere user experience om de gebruiker te informeren dat er een update zal uitgevoerd worden. (Wicki, 2017)

De ontwikkelaar heeft dus volledige vrijheid in hoe hij omgaat met updates. De gebruiker kan steeds genieten van de laatste versie van de applicatie zonder dat hij deze zelf manueel moet updaten. Deze ervaring is dezelfde voor traditionele websites.

Zowel Android als IOS hebben een functionaliteit om hun applicaties automatisch te updaten. Deze functionaliteit werkt echter enkel voor kleine incrementele updates. Als de applicatie functionaliteiten aanpast, toevoegt of verwijdert, moet deze app nog steeds manueel geüpdatet worden via de app-store. Als ontwikkelaar heb je dus geen controle over welke versie de gebruiker aan het gebruiken is.

(Apple, 2020c) (AndroidDevelopers, 2020)

2.3.10 Conclusie

	Web applicatie	PWA	Native applicatie
Bereik	Green	Green	Red
Platformonafhankelijkheid	Green	Orange	Red
Omzet	Red	Orange	Green
Bundle size	niet van toepassing	Green	Red
Offline gebruik	Red	Green	Green
Betrokkenheid	Red	Orange	Green
Kost	niet van toepassing	Green	Red
Deployment	Green	Green	Red
Updates	Green	Green	Red

Tabel 2.50: concluderende tabel 'waarom een PWA'

2.4 Beperkingen van een PWA

Zoals aangetoond in de sectie besturingssystemen en PWAs kan het web gebruik maken van verschillenden hardware-functies van een toestel. Onderzoek toont echter dat voor sommige toepassingen een PWA niet de oplossing is. (Ivano, 2016)

2.4.1 Cameragebruik

Onderzoek van Rebecca Fransson toont aan dat de videos die opgenomen worden met de mediaCapture API van een veel lagere kwaliteit zijn. Ook duurde het proces van het opnemen van een video statistisch significant langer. De methode waarbij er gebruik gemaakt wordt van de mediaCapture API is wel ondersteund door alle populaire browsers.

De resultaten waarbij videos opgenomen worden met de nieuwere ImageCaptureAPI waren een stuk beter en benaderden de kwaliteit van een native applicatie. Het probleem bij deze techniek is dat deze API enkel ondersteund wordt door Google Chrome. (Fransson & Driaguine, 2017)

2.4.2 Controle over platformen

Met native applicaties kan de ontwikkelaar kiezen voor welke platformen hij de applicatie zal publiceren. Het web, en dus ook PWAs, kunnen van op verschillende toestellen bezocht worden. Dit is een van de voordelen van het web. Dit zorgt er echter voor dat de ontwikkelaar rekening moet houden met de verschillende mogelijkheden van de toestellen. Er kan een applicatie geschreven worden die afhankelijk is van een cameratoepassing. Toestellen die geen camera hebben (oudere telefoons, smart-tv, desktops,) kunnen perfect op deze site terecht komen. Helaas zullen zij niet kunnen genieten van de functionaliteit van de applicatie.

Een webapplicatie kan geopend worden op een scherm van enkele centimeters groot. Deze zelfde applicatie kan ook geopend worden op een groot televisiescherm. De ontwikkelaar moet ervoor zorgen dat de applicatie op al deze groottes bruikbaar is.

2.4.3 Functies van een besturingssysteem

Integratie in het besturingssysteem is ook niet mogelijk. Een PWA kan dan wel geïnstalleerd worden en toegevoegd worden aan het startscherm zodat het aanvoelt als een native app, maar bepaalde toepassingen zijn enkel mogelijk met native applicaties.

Een PWA kan bijvoorbeeld geen widgets plaatsen op het startscherm van een Android toestel.

De instellingen van een toestel kunnen ook niet gemanipuleerd worden vanuit een PWA. Native applicaties kunnen bepaalde acties uitvoeren door de ingebouwde smart assistant (Google Assistant voor Android, Siri voor IOS), Deze assistenten kunnen nog niet interageren met geïnstalleerde PWAs.

Zoals aangetoond in de sectie besturingssystemen en PWAs kan een PWA ook geen toegang krijgen tot de alarmen van een toestel.

Toegang tot de hardware van toestellen is nog steeds beperkt. Er bestaan reeds heel wat web-APIs voor het aanspreken van deze sensoren. De ondersteuning van deze technologieën is nog niet goed (zie sectie besturingssystemen en PWAs). De toegang tot persoonlijke informatie is ook beperkt. Er is voor webapplicaties geen toegang tot belgeschiedenis, berichten, kalender, Dit is data waar native applicaties wel gebruik kunnen van maken.

(Bouek, 2017)

2.4.4 Browserondersteuning

De ondersteuning van de functionaliteiten die een PWA aanbiedt, is niet op elke browser gelijk. Alle moderne browsers ondersteunen service workers, maar meer specifieke functionaliteiten zoals push-notificaties zijn nog niet overal beschikbaar.

2.4.5 Aanwezigheid in app-stores

PWA zijn gemakkelijk te vinden via het web, maar bepaalde potentiële gebruikers zullen een applicatie in een app-store verwachten. Deze zal hier echter niet te vinden zijn.

2.4.6 Conclusie

	Web applicatie	PWA	Native applicatie
Cameragebruik	[Red]	[Orange]	[Green]
Controle over platformen			
Functionies besturingssysteem			
Browserondersteuning	[Green]	[Orange]	niet van toepassing
Aanwezigheid in app-stores	[Red]		[Green]

Tabel 2.51: concluderende tabel 'beperkingen van een PWA'

2.5 Tools voor het ontwikkelen van en een PWA

2.5.1 Lighthouse

Lighthouse is een tool die een audit zal uitvoeren op een website en een rapport zal uitgeven. Dit rapport kan een ontwikkelaar veel interessante informatie geven om de ervaring van deze website te verbeteren. De audit geeft inzichten op vlak van:

- Prestaties
- Toegankelijkheid
- Best practices
- SEO
- PWA

(Lighthouse, 2020)

Voor elk van deze onderdelen, op PWAs na, zal er een score op 100 gegenereerd worden. De audit zal ook voorstellen doen om deze scores te verhogen en de ervaring voor de eindgebruiker dus te verbeteren.

Voor het onderdeel PWA zal er een checklist gegenereerd worden met items die moeten voldaan worden zodat de applicatie geïnstalleerd kan worden op een toestel. Dit onderdeel wordt opgedeeld in drie delen: fast and reliable, installable en PWA optimized.

Fast and reliable

De eerste metriek die getest wordt, is de snelheid van de website. Er wordt zowel gekeken naar de snelheid waarmee de eerste inhoud op de pagina komt (FMP first meaningful paint) maar ook naar wanneer de gebruiker voor het eerst kan interageren met de website (TTI time to interactive).

(web.dev, 2020a)

Vervolgens worden de offline capaciteiten van de website getest. Er wordt getest of als de huidige pagina en startpagina reageren met een antwoord die de statuscode 200 bevat. Dit

wil zeggen dat er een succesvol antwoord was en dat de website dus offline kan werken.

Installable

In dit deel van de PWA-audit zal gecontroleerd worden of als de website voldoet aan alle vereisten om geïnstalleerd te kunnen worden. Dit zijn:

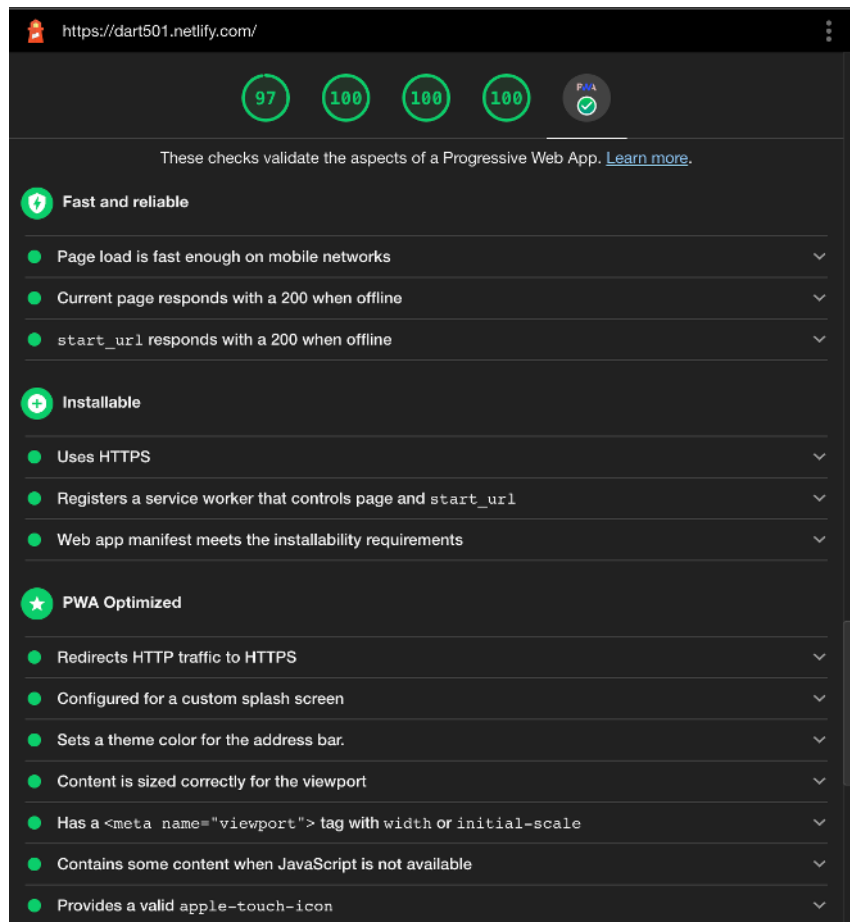
- Er moet een HTTPS-verbinding zijn.
- Er moet een service worker geregistreerd worden.
- Er moet een minimum app-manifest zijn.

(web.dev, 2020b)

PWA optimized

In dit onderdeel wordt er gecontroleerd op kleine optimalisaties die een betere gebruikerservaring kunnen bieden. Een voorbeeld hiervan is dat als een gebruiker naar de HTTP-versie van een website surft, hij herleid wordt naar de https-versie.

Een andere controle is dat het themakleur van de applicatie is ingesteld. Dit kan gebeuren in de app-manifest. Ook dit zorgt voor een betere user experience en een meer native feeling. Deze app manifest moet een specifiek icoon bevatten voor de iPhone. Ook deze controle wordt uitgevoerd.



Figuur 2.16: Screenshot van het PWA-onderdeel van een lighthouse audit op de site dart501.netlify.com

2.5.2 Workbox

Workbox is een verzameling van libraries die helpt bij het ontwikkelen van service worker gerelateerde functionaliteiten.

Workbox wil het gemakkelijker maken voor de ontwikkelaars om middelen te cachen en op deze manier een snellere en minder netwerk-afhankelijke applicatie te maken.

Het debuggen van een PWA kan ook moeilijk worden omdat code lokaal opgeslagen kan worden en je als ontwikkelaar niet altijd weet welke versie van de code voor een bepaald gedrag zorgt. Ook om deze problemen te debuggen voorziet Workbox tools.

(Workbox, 2020)

2.5.3 PWABuilder

Pwabuilder.com is een website, die gecreëerd werd door Microsoft, die ervoor zorgt dat een PWA toch in de app-stores kan terechtkomen. Als ontwikkelaar moet je juist een link van een PWA opgeven en de tool maakt van deze PWA vier pakketten die kunnen geüpload worden naar hun bijhorende app-store.

Volgende platformen zijn ondersteund: - Android - Samsung (eigen Samsung store) - Windows - IOS

- Android
- Samsung (eigen Samsung store)
- Windows
- IOS

(PWABuilder, 2020)

2.5.4 Chrome developer tools

De Chrome developer tools bieden een grote hulp bij het ontwikkelen van web-applicaties. Maar het onderscheidt zich van andere browsers als het op PWAs aankomt.

Chrome zorgt ervoor dat er eenvoudig gecachte items tijdelijk verwijderd kunnen worden. Hierdoor kan de ontwikkelaar testen als een service worker de juiste bestanden offline beschikbaar maakt.

Google Chrome biedt tools om eenvoudig een website offline te testen of om bepaalde service workers uit te schakelen. Ook de volledigheid van het app-manifest bestand kan in de Chrome developer tools bekeken worden

(Developers, 2019c)

2.6 PWA-alternatieven

In deze sectie zal er onderzoek gedaan worden naar technologieën waarbij er een applicatie ontwikkeld kan worden die op meerdere platformen kan werken waarbij er ook maar 1 codebase is. Er zal gekeken worden wat de voor- en nadelen zijn van de methodes.

Het doel van cross platform ontwikkeling is dat het gemakkelijker en sneller moet zijn om een applicatie te creëren en te onderhouden voor meerdere platformen.

Het ontwikkelen van een applicatie aan de hand van een cross platform development benadering zal minder tijd in beslag nemen, aangezien er minder code moet geschreven worden, dan het ontwikkelen van native toepassingen voor meerdere platformen.

Een ander voordeel van deze technologie is dat het gemakkelijker is om consistent te zijn met een applicatie over meerdere platformen. Als een applicatie op de traditionele manier wordt ontwikkeld, is er vaak een team voor elk platform. Deze teams maken dezelfde applicatie, maar voor een ander platform. Dit heeft als gevolg dat beide applicaties niet 100% gelijk zullen zijn. Dit kan de gebruiker verwarren.

2.6.1 Cross platform hybrid development

Hybride mobiele applicaties zijn applicaties waarbij de userinterface weergegeven wordt in een minimale versie van een webbrowser.

Dit type applicatie kan gebouwd worden met de technologieën die beschikbaar zijn voor het web.

Bij het creëren van een hybride applicatie wordt er een minimale browser aangemaakt in het package van de applicatie. De applicatie wordt vervolgens in deze minimale browser gebouwd. Deze combinatie van bestanden kan vervolgens geüpload worden naar platformen zoals de Google Play store of de App Store van Apple.

(Huynh, Ghimire & Truong, 2017)

Voordelen

Het grote voordeel van hybride applicaties is dat de overstap van web-development naar het ontwikkelen van hybride applicaties heel klein is. De technologieën die gebruikt worden bij web-ontwikkeling kunnen overgenomen worden om mobiele applicaties te ontwikkelen. Er moeten dus geen nieuwe frameworks of technologieën geleerd worden.

Dit betekent ook dat de vele libraries en packages die beschikbaar zijn voor het web ook gebruikt kunnen worden voor app-ontwikkeling.

Hybride applicaties hebben in tegenstelling tot webapplicaties en PWAs wel toegang tot alles wat een besturingssysteem beschikbaar stelt voor applicaties. Een hybride applicatie heeft bijvoorbeeld wel volledige toegang tot het sms-verkeer van een toestel (Ionic, 2020b)

Nadelen

De userinterface die gemaakt wordt aan de hand van een hybride applicatie is op elk toestel dezelfde. Dit zorgt ervoor dat de app niet intuïtief aanvoelt. Een native Android-applicatie heeft bijvoorbeeld een ander navigatiesysteem dan een IOS-applicatie. Bij het maken van een hybride oplossing zal deze navigatie op beide platformen dezelfde zijn.

De prestaties van een website die in een webview werkt zijn minder goed dan bij een native applicatie.

(Asp, 2017)

Apache cordova

Apache Cordova is een technologie die een website ontsluit in een webview. Deze webview kan gezien worden als een basisversie van een gewone mobiele browser zonder interface-elementen zoals een url-veld of een statusbar. Er bestaan ook veel plugins die het gemakkelijk maken om de functies van het besturingssysteem te gebruiken. Voorbeelden hiervan zijn het gebruik van de camera of de flashlight.

Ionic

Ionic is een framework voor het bouwen van hybride applicaties. Het maakt gebruik van de webview die cordova aanbiedt.

De functionaliteiten die Ionic aanbiedt kunnen gebruikt worden met de populaire frontend frameworks: Angular, React en Vue.

Ionic framework heeft ook een ruime bibliotheek aan plug-ins die ervoor zorgen dat een applicatie gebruik kan maken van de functies die een besturingssysteem heeft.

Een applicatie die ontwikkeld is in Ionic, kan gepubliceerd worden in de Apple app-store, de Google Play-store en ook als PWA. Verschillende plugins die gebruikt kunnen worden voor de native applicaties kunnen ook gebruikt worden voor PWAs. Ionic is dus heel geschikt voor cross platform development.

(Ionic, 2020a)

De ontwikkelervaring van een Ionic applicatie is ook aangenaam omdat Ionic live reloading ondersteunt. Dit wil zeggen dat de applicatie niet opnieuw gebouwd moet worden om de aanpassingen te zien.

(Lucas, 2020)

Het Ionic framework zal er ook voor zorgen dat de applicatie meer native aanvoelt door de UI-elementen die het aanbiedt. Als een applicatie gebouwd wordt zal Ionic ervoor zorgen dat de juiste UI-componenten voor elk toestel gebruikt worden. Zo zal een Androidapplicatie het material design volgen en een IOS-applicatie het human design.

Een Ionic applicatie zal meer middelen vragen van een toestel. Als prestaties een belangrijke factor zijn bij de applicatie die gebouwd moet worden, is deze hybride benadering dus niet ideaal.

Een Ionic applicatie is zowel intensief voor de CPU als voor de batterij. Wat wel opvallend is, is dat de grootte van een applicatie kleiner is dan bij cross platform native development.

(Asp, 2017)

Electron

Electron is een framework dat gebruikt kan worden om desktopapplicaties te bouwen, gebruik makende van web technologieën. De code kan gecompileerd worden naar een native programma voor Linux, Mac en Windows.

Een applicatie die geschreven is met Electron heeft, in tegenstelling tot een website, wel de rechten om bestanden op het toestel aan te passen of aan te maken.

Electron voorziet ook APIs om gebruik te maken van geavanceerde functionaliteiten zoals het bestandssysteem en bluetooth.

Een groot voordeel van ontwikkelen met Electron is dat Chromium gebruikt wordt als webview. Dit zorgt ervoor dat een applicatie zich op elk besturingssysteem gelijk zal gedragen. Dit zorgt er ook voor dat de ontwikkelaar toegang heeft tot de Developer Tools die normaal gebruikt worden bij het ontwikkelen van een webapplicatie.

Een van de grote nadelen van Electron is dat het startproject al heel groot is. De Chromium webview heeft alleen al meer dan 20 miljoen lijnen code. Dit zorgt ervoor dat Electron enkel interessant wordt voor heel grote projecten waarbij dit gerechtvaardigd kan worden. De code is ook niet geëncrypteerd. Dit wil zeggen dat een gebruiker die een programma downloadt aan de broncode kan van dit programma kan. Als de code van een applicatie niet beschikbaar mag zijn, kan Electron niet gebruikt worden.

Voorbeelden van software die ontwikkeld zijn met Electron zijn:

- Slack
- Visual studio code
- Discord

2.6.2 Cross platform native development

Het grote verschil met hybrid applications is dat bij cross platform native development de code gecompileerd wordt naar native componenten. Er wordt dus geen webview gebruikt.

(Ghinea & Biørn-Hansen, 2018) (De Coninck, 2019)

Voordelen

De meeste cross platform development frameworks hebben een functie hot reload. Dit wil zeggen dat de app tijdens het ontwikkelen op een emulator kan uitgevoerd worden. Elke keer een verandering aangebracht wordt, zal deze app direct updaten. Dit vereenvoudigt het ontwikkelproces.

Bij cross platform development wordt de code omgevormd tot native componenten. Dit zorgt ervoor dat een applicatie een meer native feeling zal hebben dan bij hybrid applications. Een knop die gecompileerd wordt naar Android zal er anders uitzien dan een knop

die gecompileerd wordt naar IOS.

(Asp, 2017)

Nadelen

De prestaties zijn dan wel beter dan bij een hybrid applicatie, maar voor applicaties die veeleisend zijn op vlak van CPU en geheugen zijn cross platform development frameworks niet de beste oplossing. Hier wordt beter een native oplossing gebruikt waarbij er toch nog twee of meer codebases zijn.

(Asp, 2017)

Xamarin

Xamarin is een platform die gebruikt kan worden om applicaties voor Android, IOS, Windows en Mac OS te maken. Xamarin is ontwikkeld en wordt onderhouden door Microsoft. De taal en technologie die bij dit platform gebruikt wordt is C#. Naar schatting kan 90% van de code in C# geschreven worden. De resterende 10% van de code moet nog steeds geschreven worden in de taal van het platform zoals Java voor Android, Swift voor IOS. De code die niet met C# geschreven is, is de code die de userinterface maakt.

(Altexsoft, 2019) (Warcholinski, 2020)

De IDE Visual studio voorziet veel tools die het ontwikkelen van Xamarin applicaties makkelijker maakt. (Studio, 2020)

Bij ontwikkeling met Xamarin kan er ook gebruik gemaakt worden van Xamarin forms. Dit is een library die native componenten voorziet voor verschillende besturingssystemen. Zo zal elk component er verschillend uitzien op verschillende platformen, dit om aan het verwachtingspatroon van de gebruiker te voldoen. (Microsoft, 2019)

Xamarin is het framework die de cross-platform trend heeft gestart. Het platform werd begin 2013 uitgebracht. Doordat het al enkele jaren bestaat, zijn er al veel vragen gepost op fora, waardoor de kans groot is dat, als een ontwikkelaar een probleem heeft, dit gedocumenteerd staat op een forum.

Dit is ook terug te zien in het aantal zoekopdrachten op Stackoverflow. Bij de categorie frameworks staat Xamarin op de 10e plaats. Enkel React native en Cordova staan hoger op deze lijst.

(StackOverflow, 2020)

React native

React native(Reactnative, 2020)) is een framework gebaseerd op React.js. Beide frameworks zijn ontwikkeld en worden onderhouden door Facebook en worden geschreven in

JavaScript. Bij React native moeten bepaalde specifieke onderdelen nog geschreven worden in de native codeertaal.

Het voordeel van React.js en React native is dat er een learn once, write anywhere benadering is. React.js en React native zijn heel gelijkaardig, dit zorgt er dus voor dat een ontwikkelaar maar 1 technologie moet leren en dan voor zowel web als mobiel kan ontwikkelen.

React native maakt gebruik van een concept genaamd bridges. Dit is de vertaling die gemaakt wordt van JavaScript naar native. React native voorziet deze bridges voor de veel voorkomende items zoals een button, een text field, een calendar. Als er meer specifieke en minder courante functies gebruikt worden, moeten deze bridges zelf geschreven worden. Deze bridges moeten geschreven worden in de taal van het platform. (Bartosz, 2019)

React native was het eerste cross-platform framework dat grote successen kende. Hierdoor is er al een grote online community waar je terecht kan bij eventuele problemen. Dit is ook terug te zien in het aantal zoekopdrachten op stackoverflow. Bij de categorie frameworks staat React native op de 6e plaats. Dit is het hoogst genoteerde framework dat mobiele applicaties produceert. (StackOverflow, 2020)

Expo

Expo is een verzameling van tools die aan React native toegevoegd worden. Het levert bijvoorbeeld een command line interface. Expo zorgt ervoor dat een volledige applicatie geschreven kan worden met enkel gebruik te maken van JavaScript. Hierdoor is de combinatie van React native en Expo populair bij webdevelopers.

Expo heeft het doel om de ontwikkeltijd van een applicatie zo laag mogelijk te houden. Een voorbeeld hiervan is dat de ontwikkelaar een applicatie kan downloaden op zijn toestel waarin de applicatie die ontwikkeld wordt, getest kan worden. Ook zijn functionaliteiten zoals push-notifications voorzien door Expo.

Expo heeft verschillende APIs om toegang te krijgen tot de verschillende functies van een toestel. De meeste applicaties kunnen gebouwd worden door gebruik te maken van deze APIs.

Het gebruik van Expo kan echter ook limiterend zijn voor een applicatie. Als ontwikkelaar ben je volledig afhankelijk van deze APIs. Er kan enkel JavaScript geschreven worden en dus geen native code zoals dit bij een traditioneel React native project wel mogelijk is. Een React native project dat gebruik maakt van Expo heeft nu ook webondersteuning. Dit houdt in dat een applicatie geschreven wordt in JavaScript en kan geëxporteerd worden naar een IOS app, Android app en een PWA.

Deze functionaliteit is nog in de Beta fase en er wordt aangeraden dit nog niet te gebruiken in productie.

(Expo, 2020)

Flutter

Flutter (Flutter, 2020) is een relatief nieuw framework voor het ontwikkelen van mobiele applicaties. De programmeertaal die gebruikt wordt bij Flutter is Dart. Zowel Flutter als Dart zijn gecreëerd en worden onderhouden door Google.

Net zoals React native is het nu ook mogelijk om een PWA te maken aan de hand van Flutter. Ook bij Flutter is het nog niet aangeraden om dit al effectief te gebruiken omdat PWA-ondersteuning nog in de beta fase is.

Flutter zorgt er ook voor dat een project 100% in Dart geschreven kan worden. Er is dus geen nood om native talen te leren.

Een nadeel is dat Flutter een heel nieuw framework is en dat er online minder hulp te vinden zal zijn voor de ontwikkelaar dan bij React native. Dit is ook terug te zien in het aantal zoekopdrachten op stackoverflow. Bij de categorie frameworks staat Flutter slechts op de 12e plaats. Dit is slechts het 4e framework voor app development. (StackOverflow, 2020)

2.6.3 Conclusie

Er kan geen duidelijk antwoord geformuleerd worden over welke technologie er de beste is. Dit zal steeds afhangen van het doel van de applicatie en de achtergrond die de ontwikkelaar heeft.

De eerste keuze die gemaakt moet worden is of er een hybride applicatie gemaakt zal worden of een native applicatie. Bij deze beslissing is het belangrijk om te kijken naar wat het doel en het doelpubliek is van de applicatie. Als de meeste gebruikers verwacht worden op het web, is een hybride applicatie een goede oplossing. De ondersteuning van hybride applicaties is heel goed voor websites. Dit is iets wat nog niet op punt staat bij native cross platform benaderingen. Als de meeste gebruikers verwacht worden op native applicaties, is cross platform native ontwikkeling de beste optie. De applicatie wordt omgezet naar native componenten en dit zorgt voor een meer native gevoel. De gebruikerservaring zal beter zijn.

Als de keuze gemaakt is om native cross platform development toe te passen, zijn er nog steeds verschillende opties. Hier zal de achtergrond van de ontwikkelaar een grote rol spelen in welke technologie de optimale zal zijn. Ontwikkelaars met een web en JavaScript achtergrond kunnen gemakkelijk de overstap maken naar React native en Expo.

Voor ontwikkelaars met een C# achtergrond is een applicatie in Xamarin een logische keuze.

Flutter is geschreven in een nieuwe taal waar niet veel ontwikkelaars een achtergrond

in hebben. Maar Dart leunt dicht bij Java, dus voor ontwikkelaars die hier ervaring in hebben, is Flutter de logische keuze.

Als er geen achtergrond in een van deze technologieën is, is Flutter een goeie keuze. (Leler, 2017) (Wenhao, 2018)

Flutter belooft een snellere ontwikkeltijd en is heel performant. Het is ook een groot voordeel dat er geen native code geschreven moet worden. Deze belofte wordt ook bevestigd door Jakub Fagietto. (Fagietto, 2019)

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 agent-schappen, 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 & awwards, 2017)

Gebruikers raken gefrustreerd als het gedrag van een mobiele applicatie anders is op de verschillende platformen. Dit verschil komt er omdat dit totaal verschillende 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 e.a., 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 nageemaakt. 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 Application shell architecture

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

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 technologieën die gebruikt worden om traditionele webapplicaties te bouwen. Voor de onderzoeken zal voor de eerste fase HTML,

CSS en JavaScript gebruikt worden. Eventueel kan er ook gebruik gemaakt worden van een javascript library zoals React. Er zijn verschillende tools beschikbaar voor het creëren van 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.