HO GENT

Faculteit Bedrijf en Organisatie

De interactie van een progressive web application met het besturingssysteem en een 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

Faculteit Bedrijf en Organisatie

De interactie van een progressive web application met het besturingssysteem en een 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

Deze proef werd geschreven in het kader van het behalen van het diploma 'bachelor in de toegepaste informatica' aan HoGent.

Het onderwerp van deze proef, PWA's, werd voorgesteld door Bothrs. Ik vond dit een heel interessante case en het was dan ook heel boeiend om te ontdekken wat er wel en niet mogelijk is met PWA's.

Ik heb een grote interesse in nieuwe technologieën maar ook in bedrijfsstrategieën en ondernemen. Het onderzoeken van een nieuwe technologie waarbij de time to market drastisch verlaagd zou kunnen worden, was dan ook een perfecte match voor mij.

Ik zou graag Bothrs bedanken. Dit is de studio die me het onderwerp heeft voorgesteld. Tijdens mijn bachelorproef kon ik bij verschillende personen terecht voor hulp of advies. Ook de oprechte interesse van deze personen in de technologie was een grote motivatie tijdens het schrijven van deze scriptie.

Tenslotte wil ik ook graag mijn promotor Karine Samyn bedanken. Ik kreeg op heel regelmatige basis uitgebreide, gerichte en doordachte feedback. Tevens merkte ik bij mevrouw Samyn een oprechte interesse in het onderwerp. Dit alles heeft zeker een positieve impact gehad op het eindresultaat.

Samenvatting

PWA's zijn webapplicaties die gebruik maken van moderne web-technologieën om een ervaring aan te bieden die gelijkaardig is aan die van een native applicatie. Er is steeds meer interesse in PWA's (**googleTrends2020**)

De technologie belooft veel problemen waar ontwikkelaars en digitale agentschappen al jaren met kampen op te lossen. Een PWA is echter nog steeds gelimiteerd op bepaalde vlakken. Het is dus belangrijk dat er in kaart gebracht werd wat wel en niet bereikt kan worden met de technologie.

Deze scriptie zal in eerste instantie onderzoeken wat technisch wel en niet mogelijk is met een PWA. De voor- en nadelen van de technologie zullen hier opgelijst en besproken worden. In deze sectie zal er ook op zoek gegaan worden naar andere technologieën die wel aan de tekortkoming van PWA's kunnen voldoen. Vervolgens zal er op een praktische wijze onderzocht worden wat er nodig is om een bestaande traditionele webapplicatie om te vormen tot een PWA die installeerbaar is en offline gebruikt kan worden. Ten slotte zal een proof-of-concept uitgewerkt worden waarbij verschillende web-technologieën gebruikt zullen worden om een volwaardige video conference applicatie te ontwikkelen.

Een webapplicatie kan eenvoudig omgevormd worden tot een installeerbare, offline ervaring. De proof-of-concept toonde dat aan de hand van web-API's functionaliteit verkregen kan worden die voordien enkel beschikbaar was voor native applicaties. Echter de ondersteuning van PWA's is nog niet goed op bepaalde platformen en is het dus zeker nog geen vervanger voor native applicaties.

PWA's zijn een relatief nieuwe technologie. Er zijn verschillende web-API's die nog verder onderzocht zouden kunnen worden. Zo zou er verder onderzoek gedaan kunnen

worden naar de impact van push-notificaties op vlak van user engagement van een webapplicatie en naar de prestaties van bepaalde web-technologieën zoals webRTC.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Definities

A2HS	'Add to homescreen' is een functionaliteit waarbij een PWA kan toegevoegd worden aan het startscherm van een toestel.
Third party package	Een herbruikbare functionaliteit die gemaakt is door een onafhankelijke partij.
Native applicatie	Een applicatie die ontwikkeld is met de technologieën specifiek voor een bepaald besturingssysteem. Deze applicaties kunnen gedownload worden van een app-store.
HTTPS	Een HTTPS-verbinding is een verbinding tussen de client en de server waarbij de data die verstuurd wordt versleuteld is.
JSON	JSON of Javascript Object Notation is een bestandsformaat dat vaak gebruikt wordt om data tussen een server en een client te delen.
API	API of Application Programming Interface. Een API zorgt ervoor dat twee programmas met elkaar kunnen communiceren.
DOM	DOM of Document Object Model is de structuur waaruit een Html-bestand opgebouwd is.
Native applicatie	Een native applicatie is een applicatie die ontwikkeld is voor een specifiek platform.
Low end	Low end toestellen zijn toestellen die minder goede prestaties zullen leveren dan een gemiddeld toestel.
HDR	HDR of High Dynamic Range is een techniek waarbij meerdere fotos genomen worden op een heel korte tijd, deze worden dan samengevoegd tot één foto. Dit wordt gedaan om de kwaliteit van de foto te verhogen.
IDE	IDE of Integrated Development Environment is een verzameling van tools die het ontwikkelen van software makkelijker maakt.

Tabel 1: definities

1. Inleiding

In deze thesis zal er onderzoek gedaan woorden naar Progressive Web Applications (PWA).

Een PWA is een webapplicatie die gebruik maakt van moderne webtechnologieën om op deze manier een ervaring aan te bieden die dicht bij die van een native applicatie ligt.

1.1 Probleemstelling

Het uitbrengen van een digitaal product is vaak duur en tijdrovend. Dit komt omdat er voor meerdere platformen vaak verschillende codebases nodig zijn en er dus duplicate code geschreven moet worden.

PWA's kunnen een oplossing bieden voor bedrijven die zo snel mogelijk een applicatie willen publiceren op zoveel mogelijke platformen.

Voor digitale agentschappen en start-ups is het vaak essentieel om een product naar buiten te kunnen brengen in een zo kort mogelijke tijd om een competitief voordeel de creëren.

Het web kan een oplossing bieden voor beide problemen. Veel developers zijn vaardig met webtechnologieën en een webapplicatie kan op alle mogelijke moderne toestellen gebruikt worden.

Echter, het web heeft altijd aantal beperkingen gehad. Een van de doelen van PWA's is deze gebreken verhelpen.

1.2 Onderzoeksvraag

Deze thesis zal een antwoord bieden op de vraag: "Welke applicaties kunnen er ontwikkeld worden als progressive web application?".

Om een antwoord te kunnen formuleren op deze onderzoeksvraag, zal er eerst antwoord moeten gegeven worden op volgende onderzoeksvragen:

- 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?

1.3 Onderzoeksdoelstelling

Om deze onderzoeksvragen goed te kunnen beantwoorden zullen er verschillende onderdelen in deze bachelorproef uitgewerkt worden.

Er zal een literatuurstudie uitgevoerd worden om een beeld te schetsen van wat er functioneel wel en niet mogelijk is met PWA's. De literatuurstudie zal een antwoord bieden op volgende onderzoeksvragen:

- 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?

Na de literatuurstudie zal er onderzocht worden welke stappen er moeten ondernomen worden om van een traditionele website een PWA te maken. Dit zal gebeuren aan de hand van een voorbeeld.

Vervolgens zal er een proof-of-concept ontwikkeld worden. Deze zal een aantal functionaliteiten die beschikbaar zijn voor PWA's testen. Er zal een nadruk liggen op de gebruikerservaring die aangeboden kan worden met PWA's.

Op basis van de literatuurstudie en de bevindingen van de proof-of-concept zal er een besluit geformuleerd worden.

1.4 Opzet van deze bachelorproef

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 zal er onderzocht worden welke stappen er ondernomen moeten worden om een traditionele website toe te kunnen voegen aan het startscherm, en dus om te vormen tot een PWA.

in Hoofdstuk 5 zal er een proof-of-concept uitgewerkt worden die bekijkt welke functionaliteiten er beschikbaar zijn voor PWA's en welke gebruikservaring er aangeboden kan worden.

In Hoofdstuk 6 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.

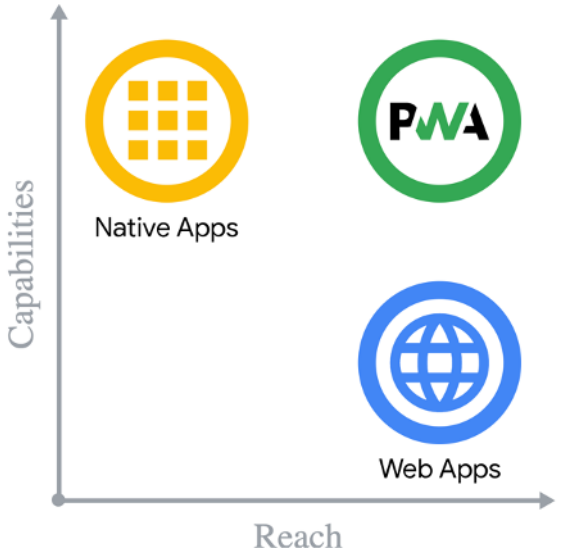
Native applicaties zijn betrouwbaar en bieden een heel goede gebruikerservaring. Ze starten op als een alleenstaande toepassing en ze kunnen uitgebreid gebruik maken van het besturingssysteem: ze kunnen bestanden lezen en schrijven, gebruik maken van usb-connecties en bluetooth, ze hebben toegang tot 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 ' is om de sterktes van webapplicaties (het bereik) en native applicaties (de functionaliteit) te combineren. (**Richard2020**) (**Google2020**)

Voorbeelden van deze extra functionaliteiten die PWA's kunnen hebben zijn:

- Een PWA kan toegevoegd worden aan het startscherm van een toestel.
- Een PWA kan push notificaties ontvangen.
- Een PWA kan offline ook gebruik worden.



Figuur 2.1: voorstelling wat een PWA is (**Richard2020**)

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 voor een PWA 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. (**Mozilla2020**)

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 de webapplicatie die er gebruik van maakt. Web workers hebben geen toegang tot het Document Object Model (DOM) van een webapplicatie, ze kunnen de inhoud van een website dus niet rechtstreeks manipuleren. (**Verdu2015**) (**Hiltunen2018**)

Services workers werken dus constant op de achtergrond, maar de manier waarop service workers opgebouwd zijn (zie hoofdstuk 2.1.1) heeft geen significante invloed op de batterijduur van een mobiel toestel. (**Malavolta2016**)

Functionaliteiten die een serivce worker mogelijk maakt

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

Offline gebruik

Als een PWA voor een eerste keer geladen wordt op een toestel zullen de bezochte pagina's offline beschikbaar blijven. Dit gebeurt door de HTML, CSS en JavaScript bestanden die nodig zijn om de pagina's te creëren lokaal op te slaan. Applicaties die geen gebruik maken van netwerkverzoeken om data te laden, zullen dus offline volledig werken.

Met service workers kunnen netwerkverzoeken en paginas ook gecached worden. Als een pagina geladen wordt, kunnen alle elementen opgeslagen worden op het toestel. Als deze pagina later opnieuw bezocht wordt, hoeft deze niet meer aan de server gevraagd te worden. Hierdoor wordt de applicatie sneller en minder afhankelijk van de netwerkverbinding. Volgens onderzoek, dat uitgevoerd werd door Google, verlaten 53% procent van de gebruikers een website als deze niet geladen is binnen 3 seconden. Service workers kunnen dus helpen om het aantal gebruikers op jouw website te verhogen. (**Google2017**)

De twee mechanismes die gebruikt worden om data offline beschikbaar te maken zijn indexedDB en de cache API. (**Osmani2019**) (**Mozilla2020a**)

Cache API 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. (**Scales2019**)

IndexedDB IndexedDB is een mechanisme dat gebruikt wordt om lokaal gestructureerde data op te slaan. Het kan vergeleken worden met object georienteerde databasemanagementsystemen die gebruik maken van JavaScript objecten om data op te slaan. Een IndexedDb maakt gebruik van indexen. Dit heeft als voordeel dat het uitlezen van data snel kan gebeuren (**Mozilla2019**)

Notificaties

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

Binnen lokale notificaties kunnen we nog het onderscheid maken tussen persistente en niet-persistente notificaties. Niet-persistente notificaties zijn notificaties die enkel getoond kunnen worden als de applicatie geopend is. Dit type notificaties heeft geen service worker nodig. Persistente notificaties zijn notificaties die nog steeds geactiveerd worden vanuit de code op het toestel, maar de applicatie moet niet meer actief zijn. Hier is wel een service worker nodig. Push notificaties worden niet geactiveerd binnen de applicatie, maar worden geactiveerd door een server. Om push notificaties te gebruiken, moet er gebruik gemaakt worden van twee webAPIs: de notifications API en de Push API.

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

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

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

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

body	De boodschap die in de melding staat
icon	Het icoontje dat in de notificatie wordt getoond.
vibrate	Het vibratiepatroon dat de melding zal maken in milliseconden.
data	Data is een object dat gebruikt kan worden als de gebruiker op de notificatie klikt. Dit object zal dan ontvangen worden in de applicatie. Hier zal vaak het id van de gebruiker teruggevonden worden.
actions	Er kunnen ook acties toegevoegd worden aan de melding. Elk object in deze array zal een knop worden op de melding met een andere functie. Het gedrag van de knoppen wordt bepaald in de applicatie aan de hand van de action.

Tabel 2.1: beschrijving Notifications API

(**Developers2019**) (**Mozilla2019a**)

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 API zal dan gebruik maken van de notifications API om een melding op het toestel van de eindgebruiker te tonen. (**Mozilla2019b**) (**Gaunt2020**)

Achtergrondsynchronisatie

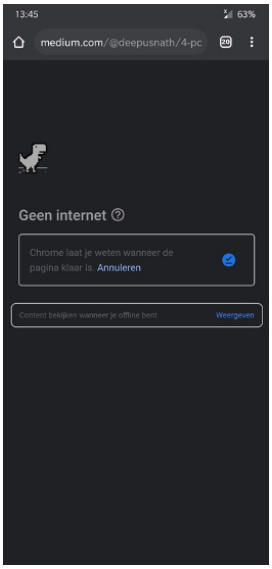
Een PWA kan gebruik maken van de background sync API om achtergrondsynchronisatie toe te passen.

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

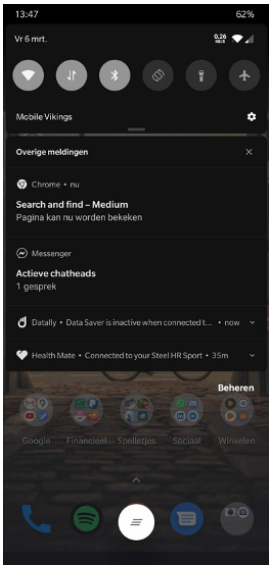
Achtergrondsynchronisatie is het proces waarbij een netwerkverzoek, dat uitgevoerd werd als er geen of een te zwakke internetverbinding was, wordt opgeslagen in de service worker en wordt uitgevoerd als er wel een stabiele internetconnectie is.

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 terug een internetverbinding heeft, en het de pagina heeft kunnen downloaden, zal de gebruiker een melding krijgen die met de boodschap dat de pagina bekeken kan worden.



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

Service worker lifecycle

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

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

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

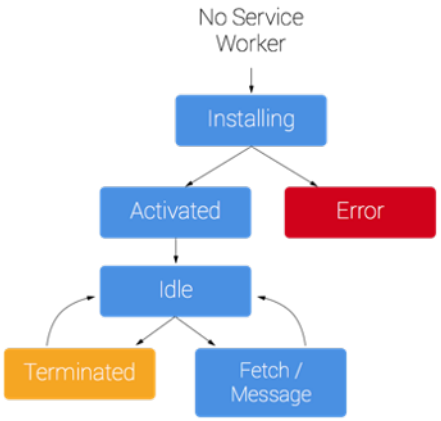
Als een service worker wordt geïnstalleerd, worden de opgegeven statische bestanden (fotos, css-bestanden, javascript-bestanden) gedownload. Als dit slaagt, wordt er naar de activatiefase gegaan, als dit niet slaagt zal dit proces zich herhalen tot het slaagt.

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

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

Nu het oude cachegeheugen up-to-date is, zal de service worker overgaan naar een rust-toestand, hierbij wacht de service worker op netwerkverzoeken van bestanden die binnen zijn scope vallen.

Als er een netwerkverzoek wordt verstuurd, zal de service worker deze verzoeken afhandelen. Na een bepaalde tijd zal de service worker terug naar de rust-modus gaan tot er een nieuw netwerkverzoek is. De service worker gaat naar deze rust-toestand om zowel cpu-kracht als geheugen te sparen. (**Gaunt2019**)



Figuur 2.4: schema levenscyclus van een service worker (**Gaunt2019**)

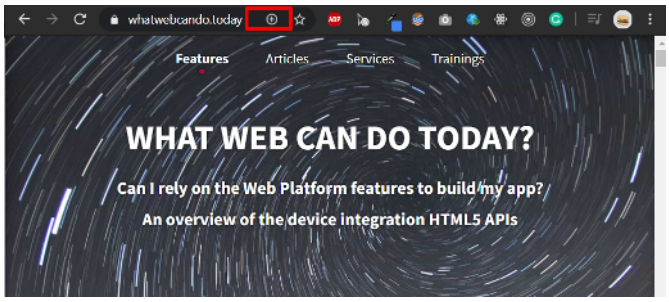
2.1.2 A2HS

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.

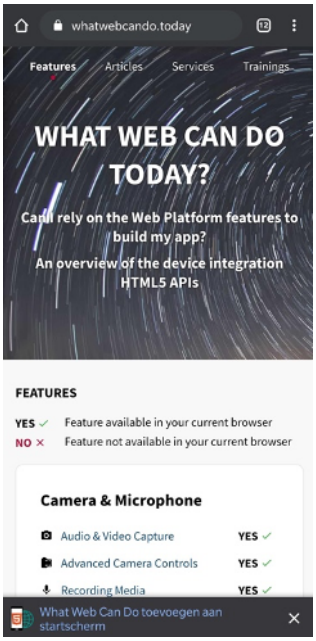
Een website moet voldoen aan volgende criteria:

- 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: beforeinstallprompt in Google Chrome op windows 10



Figuur 2.6: beforeinstallprompt in Google Chrome op android

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. Niet alle gebruikers zullen weten hoe ze dit moeten doen, het is dus aangeraden om de gebruiker hierin te begeleiden door hem duidelijke instructies te geven. (**PWAbuilder2020**)

2.1.3 Application shell

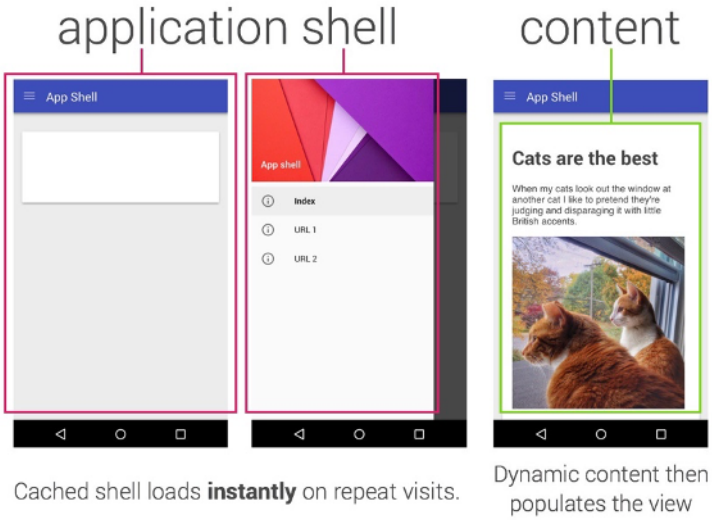
De application shell of app shell is de minimale HTML, CSS en JavaScript die nodig is om een userinterface te tonen op een toestel. Dit is vaak de header, footer en de navigatie.

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

(**Osmani2019a**)



Figuur 2.7: voorbeeld van een application shell architectuur (**Osmani2015**)

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. (**Vanhala2017**)

2.1.5 Application manifest

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

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

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

short_name	Naam van de applicatie die op het startscherm gebruikt zal worden. Deze mag maximaal 12 karakters lang zijn.
name	Naam die op alle andere plekken gebruikt zal worden: vb bij de vraag als de app geïnstalleerd mag worden. Deze mag maximaal 45 karakters lang zijn.
icons	Een lijst vna objecten die het icoon bepaalt dat de applicatie zal gebruiken. Verschillende platformen vragen verschillende type's iconen Dit object heeft volgende eigenschappen: • src - bron van het icoont • type - het bestandstype van het icoon • size - de afmeting van het icoon
start_url	De url naar waar de PWA moet gaan als de applicatie gestart wordt vanaf het startscherm van een toestel.
background_color	Hier wordt een kleur gedefinieerd. Dit kleur zal gebruikt worden voor het opstartscherm.
display	Dit bepaalt in wat voor webview de PWA getoond zal worden. Mogelijkheden zijn: • fullscreen opent de browser zonder UI-elementen (adresbalk, terug knop, .) • standalone opent de applicatie als een native applicatie los van de browser. Er worden geen UI-elementen van de browser getoond. • nimal-ui opent de applicatie in de browser maar toont slechts beperkte UI elementen van de browser. De adresbalk is weg maar de vorige knop is er nog. • browser opent de PWA in een normaal browser tabblad.
scope	De scope bepaalt alle links die binnen de PWA vallen.
theme_color	De kleur die de adresbalk zal innemen.

Tabel 2.2: beschrijving minimum application manifest

(**LePage2020**)

2.1.6 Geschiedenis van PWA's

"One last thing"

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

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

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*

(**Jobs2007**)

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. (**Strieb2016**)

De visie van Apple veranderde echter toen ze het volgend jaar de app-store lanceerden. (**Silver2018**)

Crome dev summit 2015

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

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

Ook werd het concept van de application shell voorgesteld. (**Russel2015**)

IOS 13

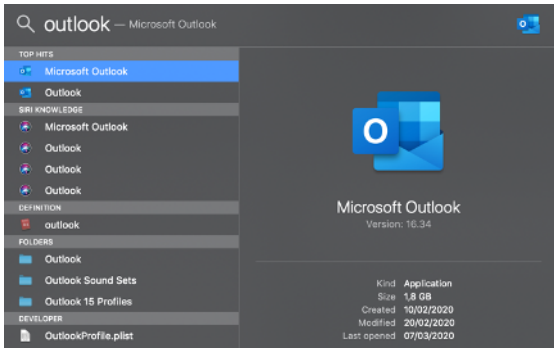
Op 19 september 2019 stelde Apple IOS 13 voor. Deze software-update zorgde ervoor dat de IPhone gebruik kon maken van een service worker. De applicaties kunnen nu ook geïnstalleerd worden op het toestel, de gebruiker moet dit wel nog zelf doen aan de hand van het menu. (**Apple2020**)

2.1.7 Voorbeelden van PWA's

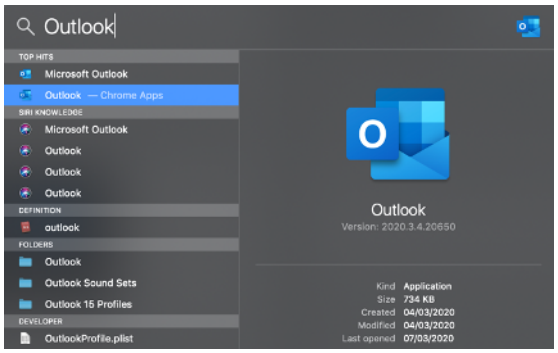
Outlook

Google is niet de enige grote speler die PWA's wil promoten en implementeren. (**Microsoft2020**)

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. (**Developers2020**)

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. (**Developers2020a**) (**Love2018**)

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. (**Formidable2020**) (**Kawatka2020**)

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. (**Croll2017**)

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%. (**Osmani2019b**)

The coronavirus app

Deze thesis is geschreven op het moment van de corona-crisis. The coronavirus app is een PWA waar real-time data in verband met het virus kan geraadpleegd worden. Deze PWA was tijdens de eerste dagen van de uitbraak al online. Dit bewijst hoe snel een PWA online gepubliceerd kan worden. Na de eerste release van de applicatie werden er dagelijks nieuwe functionaliteiten toegevoegd.

2.2 Besturingssystemen en PWA's

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 PWA's.

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

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

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

Een web-API is een API die wordt aangeboden door de browser. Het verschil met web-API's en traditionele API's is dat web-API's lokaal worden aangeboden door de browser en er dus geen internetverbinding nodig is om van deze functionaliteiten te genieten. (**Mozilla2019c**)

Als er meer informatie nodig was over de web API's werd deze gevonden op developers.google.com of op MDN web docs

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 (**DzungDTran2012**) 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. (**Fransson2017**)

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. (**Scales2020a**)

Ook meer geavanceerde functionaliteiten zijn beschikbaar. Het gedrag van de zoom en de flashlight kan ook programmatisch bepaald worden. (**Oberhofer2017**) (**Ogundipe2018**)

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 2.3: ondersteuning van de Media Capture API op 7 maart 2020

Foto's

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

Foto's 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

Foto's 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 (**Mandyam2017**) 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. (**Bhaumik2019**)

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

Tabel 2.4: ondersteuning van de Image Capture API op 7 maart 2020

Geluidopname

De mediarecorder API, (**CasasSanchez2017**) 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 en IOS verwacht. Voor Safari voor IOS bestaat deze functie al maar is het nog een experimentele functie die de gebruiker zelf moet activeren.

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

Tabel 2.5: ondersteuning van de Media Capture API op 7 maart 2020

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. (**Kinlan2019**)

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 (**Jennings2019**) 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 2.6: ondersteuning van WebRTC op 7 maart 2020

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 op een tv.

Op Apple toestellen kan een PWA nu ook AirPlay implementeren. (**Apple2020a**)

De Chrome Sender API (**Developers2020b**) zorgt ervoor dat alle modern toestellen media kunnen delen op een tv of ander scherm die dit ondersteunt.

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

Tabel 2.7: ondersteuning Apple AirPlay op 7 maart 2020

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

Tabel 2.8: ondersteuning van Chrome Sender API op 7 maart 2020

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 (**Beaufort2019**) 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 2.9: ondersteuning van Media Session API op 7 maart 2020

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 (**Grant2020**) 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. (**Beaufort2019a**)

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

Tabel 2.10: ondersteuning van Web Bluetooth API op 7 maart 2020

USB

Verkopers van toestellen met USB kunnen nu gebruik maken van de Web USB API, (**Rockot2020**). 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. (**Beaufort2019b**)

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

Tabel 2.11: ondersteuning van Web USB API op 7 maart 2020

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. (**Paus2007**)

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

Tabel 2.12: ondersteuning van Web NFC API op 7 maart 2020

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 (**RohdeChristiansen2020**), 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 (**Lamouri2014**) 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 2.13: ondersteuning van Network information API op 7 maart 2020

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.

Dit is een belangrijke eigenschap bij applicaties die real-time data gebruiken. Als de internetconnectie van de gebuiker wegvalt kan er een melding getoond worden. Op deze manier weet de gebruiker dat de data mogelijks niet up-to-date is.

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

Tabel 2.14: ondersteuning online status op 7 maart 2020

Vibratiemotor

De Vibration API (**Kostionen2018**) 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 2.15: ondersteuning vibratiemotor op 7 maart 2020

Batterijstatus

Aan de hand van de Battery Status API (**Kostiainen2016**) 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 2.16: ondersteuning batterijstatus op 7 maart 2020

Toestelgeheugen

de Device Memory API (**Panicker2018**) 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 2.17: ondersteuning toestelgeheugen op 7 maart 2020

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 (**Gregg2015**). 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 2.1.1.

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 2.18: ondersteuning lokale notificaties op 7 maart 2020

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 (**Sullivan2020**) 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 2.1.1.

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 2.19: ondersteuning push notificaties op 7 maart 2020

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 2.1.1..

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

Tabel 2.20: ondersteuning A2HS op 7 maart 2020

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 PWA's aan de hand van de Badging API. (**LePage2020a**)

Op sommige applicatie zal hier een indicatie staan van het aantal meldingen.

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

Tabel 2.21: ondersteuning Badging API op 7 maart 2020

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 (**Grigorik2017**) 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 2.22: ondersteuning voorgrond detectie op 7 maart 2020

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 (**Caceres2017**) ontwikkeld. Er kan toestemming gevraagd worden op een gelijkaardige manier voor verschillende functies.

Functies waarvoor toestemming gevraagd kan worden:

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

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

Tabel 2.23: ondersteuning Permissions API op 7 maart 2020

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. (**Hickson2016**)

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

Tabel 2.24: ondersteuning web storrage op 7 maart 2020

IndexedDB Een API voor het opslaan van grote hoeveelheden gestructureerde data op het toestel van de eindgebruiker. De data kan snel gelezen worden omdat er indexen gebruikt worden. (**Alabbas2018**)

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

Tabel 2.25: ondersteuning IndexedDB op 7 maart 2020

Cache API Deze API is gespecialiseerd in het opslaan van netwerkverzoeken. Dit is heel handig in samenwerking met een serviceworker. API-calls kunnen opgeslagen worden voor offline gebruik. (**vanKesteren2008**)

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

Tabel 2.26: ondersteuning Cache API op 7 maart 2020

Storage API Data die is opgeslagen in een van vorige technologieën kan eenvoudig verwijderd worden door de browser. Met de storage API kan data opgeslagen worden op het systeem voor een langere periode . (**Mozilla2020b**)

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

Tabel 2.27: ondersteuning Storage API op 7 maart 2020

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 (**Kruisselbrink2019**) 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 2.28: ondersteuning File API op 7 maart 2020

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 (**Tibbett2014**).

In theorie zouden PWA's 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 2.29: ondersteuning Contacts API op 7 maart 2020
* 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 (**Fullea2015**) kan er gekeken worden naar inkomende smsen. Het voorbeeld van de PayPal applicatie kan dus ook geïmplementeerd worden als PWA. PWA's 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 2.30: ondersteuning Messaging API op 7 maart 2020
* 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 (**Kulkarni2015**) kan ervoor zorgen dat taken zoals alarmen, herinneringen en gelijkaardige taken kunnen ingepland worden in het systeem. Deze API is slechts een voorstel en heeft dus nog geen ondersteuning.

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

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

Tabel 2.31: ondersteuning Task Sheduler API op 7 maart 2020

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 2.32: ondersteuning touch gebaren op 7 maart 2020

Klembord toegang

De Clipboard API (**Kacmarcik2019**) 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 2.33: ondersteuning Clipboard API op 7 maart 2020

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 2.34: ondersteuning offline gebruik op 7 maart 2020

Achtergrondsynchronisatie

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

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

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

Tabel 2.35: ondersteuning achtergrondsynchronisatie op 7 maart 2020

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 (**Giuca2019**) en de Web Share Target API (**Williger2019**) 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 2.36: ondersteuning Web Share API en Web Share Target API op 7 maart 2020

Betalingen

Aan de hand van de Payment Request API (**Denicola2019**) 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 2.37: ondersteuning Payment Request API op 7 maart 2020

Credentials

De Credential Management API (**West2019**) 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 2.38: ondersteuning Credential Management API op 7 maart 2020

Locatie en positionering

Geolocatie

De Geolocation API (**Popescu2018**) 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.

Zowel de lengtegraad als de breedtegraat kunnen opgevraagd worden, deze twee coördinaten vormen de locatie van de gebruiker.

De Geolocation API voorziet een methode "watchPosition". Elke keer dat de locatie van een gebruiker verandert, zal deze methode een functie aanroepen. Op deze manier kan de locatie van een gebruiker gevolgd worden.

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 2.39: ondersteuning Geolocation API op 7 maart 2020

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 (**Kruisselbrink2017**) uit te werken voor het web, maar op het moment van schrijven heeft geen enkele browser dit geïmplementeerd. Dit is eenfunctie die enkel beschikbaar is voor native IOS en Android-applicaties.

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

Tabel 2.40: ondersteuning Geofencing API op 7 maart 2020

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 (**Tibbett2019**) 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 2.41: ondersteuning Device Orientation API op 7 maart 2020

Toestelbeweging

De Generic Sensor API (**Waldroon2019**) 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 2.42: 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 2.43: 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 2.44: ondersteuning Gyroscope op 7 maart 2020

Magnetometer Meet het magnetische veld rond het toestel.

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

Tabel 2.45: ondersteuning Magnetometer op 7 maart 2020

Lineaire accelerometer 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 2.46: ondersteuning Lineaire accelerometer op 7 maart 2020

Nabijheid-sensoren

De Proximity API (**Kostiainen2019**) 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 2.47: ondersteuning Proximity API op 7 maart 2020

Scherm en output

Virtual en augmented reality

De webXR Device API (**Jones2019**) 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 2.48: ondersteuning WebVR en WebXR op 7 maart 2020

Fullscreen

De Fullscreen API (**Kesteren2014**) 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 2.49: ondersteuning Fullscreen API op 7 maart 2020

Wake lock

eel toestellen gaan het scherm dimmen na een bepaalde tijd van inactiviteit. Dit kan onhandig zijn voor bepaalde applicaties zoals een gps. Met de Wake Lock API (**Bogdanovich2017**) kan het automatisch dimmen van het scherm tegengegaan worden.

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

Tabel 2.50: ondersteuning Wake Lock API op 7 maart 2020

2.2.2 Concluderende tabel

	Edge	Firefox	Chrome	Safari	Android	IOS
Video met audio	Ja	Ja	Ja	Ja	Ja	ja
Foto's	Nee	Nee	Ja	Nee	Ja	Nee
Geluidopname	Ja	Ja	Ja	Nee	Ja	Nee
Real time communicatie	Ja	Ja	Ja	Ja	Ja	ja
Airplay	Nee	Nee	Nee	Ja	Nee	ja
Chromecast	Ja	Ja	Ja	Ja	Ja	ja
Bluetooth	Ja	Nee	Ja	Nee	Ja	Nee
USB	Ja	Nee	Ja	Nee	Ja	Nee
NFC	Nee	Nee	Nee	Nee	Nee	Nee
Netwerkinformatie	Ja	Nee	Ja	Nee	Ja	Nee
Onlinestatus	Ja	Ja	Ja	Ja	Ja	ja
Vibratiemotor	Ja	Ja	Ja	Nee	Ja	Nee
Batterijstatus	Ja	Nee	Ja	Nee	Ja	Nee
Toestelgeheugen	Ja	Nee	Ja	Nee	Ja	Nee
Media-controle in de notificatie	Ja	Ja	Ja	Ja	Ja	ja
Lokale notificaties	Ja	Ja	Ja	Ja	Ja	Nee
Push notificaties	Ja	Ja	Ja	Nee	Ja	Nee
A2HS	Ja	Ja	Ja	Nee	Ja	Ja
Badges	Ja	Onbekend	Ja	Nee	Ja	Nee
Voorgronddetectie	Ja	Ja	Ja	Ja	Ja	ja
Toestemmingen	Ja	Ja	Ja	Nee	Ja	Nee
Offline opslag - web storage	Ja	Ja	Ja	Ja	Ja	ja
Offline opslag - indexedDB	Ja	Ja	Ja	Ja	Ja	ja
Offline opslag - Cache API	Ja	Ja	Ja	Ja	Ja	Nee
Offline opslag - Storage API	Ja	Ja	Ja	Ja	Ja	ja
Bestandentoegang	Ja	Ja	Ja	Nee	Ja	Nee
Contacten	Nee	Nee	Nee	Nee	Beperkt	Nee
SMS	Nee	Nee	Nee	Nee	Beperkt	Nee
Taakplanning	Nee	Nee	Nee	Nee	Nee	Nee
Touch gebaren	Ja	Ja	Ja	Nee	Ja	ja
Klembordtoegang	Ja	Ja	Ja	Ja	Ja	ja

	Edge	Firefox	Chrome	Safari	Android	IOS
Offline gebruik	Ja	Ja	Ja	Ja	Ja	ja
Achtergrondsynchronisatie	Ja	Nee	Ja	Nee	Ja	Nee
Inter-app communicatie	Nee	Nee	Nee	Nee	Ja	ja
In app betalingen	Ja	Nee	Ja	Ja	Ja	ja
Credentials	Ja	Nee	Ja	Nee	Ja	Nee
Geolocatie	Ja	Ja	Ja	Ja	Ja	ja
Geofencing	Nee	Nee	Nee	Nee	Nee	Nee
Toesteloriëntatie	Ja	Ja	Ja	Nee	Ja	ja
Toestelbeweging - device motion	Ja	Ja	Ja	Nee	Ja	ja
Toestelbeweging - accelerometer	Ja	Nee	Ja	Nee	Ja	Nee
Toestelbeweging - gyroscope	Ja	Nee	Ja	Nee	Ja	Nee
Toestelbeweging - magnetometer	Nee	Nee	Ja	Nee	Nee	Nee
Toestelbeweging - lineaire accelerometer	ja	Nee	Ja	Nee	ja	Nee
Nabijheid sensoren	Nee	ja	Nee	Nee	Nee	Nee
Virtual en augmented reality	Ja	Ja	Ja	Nee	Ja	Nee
Fullscreen	Ja	Ja	Ja	ja	Ja	Nee
Wake lock	Ja	Nee	Ja	Nee	Ja	Nee

Tabel 2.51: conluderende tabel functionaliteiten van een besturingssysteem

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 API's 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.

Mobiele besturingssystemen

Net zoals bij browsers, is de ondersteuning voor PWA's 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. (**Anastasia2017**)

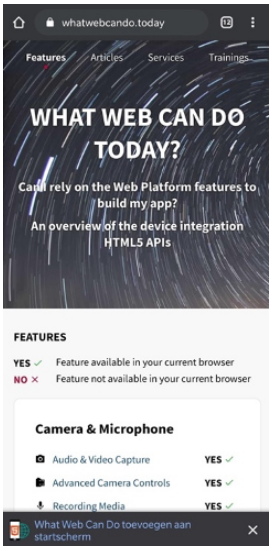
Achtergrondsynchronisatie is een functie die de gebruikerservaring kan verbeteren. Deze functie die mogelijk gemaakt wordt door service workers is ook niet beschikbaar op een IOS-toestel.

Apple laat PWA's 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 2.10: Screenshot van het installatieproces op een IPhone met IOS 13



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. (**Lathiya2020**)

conclusie Er kan geconcludeerd worden dat Android meer open staat voor intergraties met PWA's dan IOS. In deze trend lijkt er niet direct verandering te komen. Google, de ontwikkelaar van Android, is vaak de eerste om nieuwe web API's te ondersteunen.

Langs de andere kant is IOS vaak het laatste besturingssyteem die ondersteuning zal aanbieden voor een web API.

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. (**netMarketShare2020**)

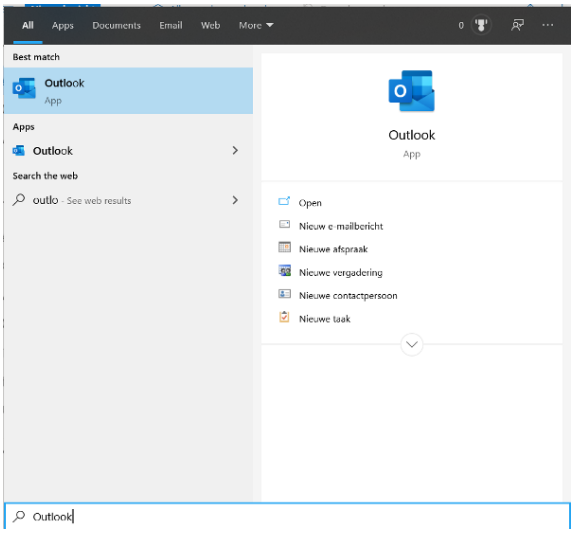
Microsoft wil PWA's zo goed mogelijk ondersteunen. Ze zijn zelf ook PWA's aan het ontwikkelen. Een voorbeeld hiervan is Outlook, de populaire online email-client kan nu ook geïnstalleerd worden als PWA. (**Microsoft2020a**)

PWA's 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 PWA's op het internet en zal deze automatisch toevoegen aan de Windows store. Dit kan wel tegengegaan worden als de uitgever van de PWA dit niet wil. (**Gustafson2017**) (**Gustafson2017a**)

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.

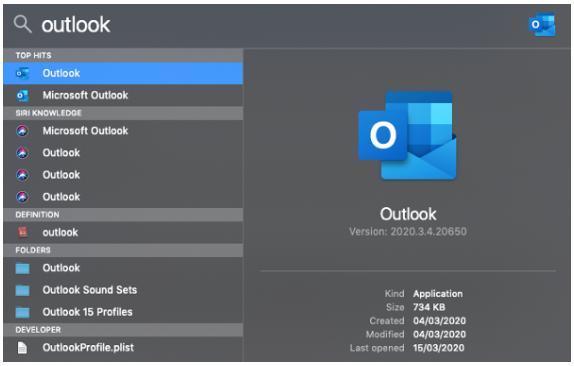


Figuur 2.12: screenshot van de zoekresultaten van Outlook op Windows 10

De tool PWABuilder die gebruikt kan worden om een PWA in de populaire mobiele app-stores te krijgen, is ook ontwikkeld door Microsoft. (**PWAbuilder2020**)

Microsoft heeft ook een uitgebreide documentatie die ontwikkelaars helpen bij het ontwikkelen van PWA's. (**Microsoft2020b**)

Apple biedt weinig ondersteuning voor PWA's 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 vastgemaakt worden aan het dock.



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

Deze algemene trend waarbij Apple PWA's bewust niet ondersteunt, lijkt niet snel te veranderen. John Wilander, een ontwikkelaar binnen Apple, bekritiseerde openlijk PWA's als een technologie die van Google is en dat dit niet in de planning zit van Apple. (**Wilander2019**)

Conclusie Er kan geconcludeerd worden dat Windows en Mac OS een volledig andere filosofie hebben als het aankomt op PWA's.

Windows probeert een zo goed mogelijke ondersteuning te geven aan PWA's. PWA's worden binnen windows behandeld als een volwaardig programma. Als een PWA geïnstalleerd is, is het moeilijk te onderscheiden van een ander native programma. PWA's zijn binnen windows ook eenvoudig om te installeren. Micorosoft probeert ontwikkelaars ook te ondersteunen in het ontwikkelen van PWA's, dit doen ze door een uitgebreide documentatie en tools aan te bieden.

Apple daarintegen is meer gesloten ten opzichte van PWA's. Vanuit de standaard browser kunnen er helemaal geen PWA's geïnstalleerd worden. Via google Chrome is dit wel mogelijk.

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. (**TandelSunil2018**)

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. (**Nath2017**)

Data toont aan dat gebruikers nog steeds 87% van hun tijd op hun smartphone spenderen in native applicaties, 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. (**GoogleChromeDevelopers2017**)

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.

Er kunnen meer geavanceerde functionaliteiten implementeren dan een traditionele we-bapplicatie. 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. Pwa's 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.

Thomas Steiner (zie bijlage A) maakte in een interview ook duidelijk dat de "Build once, run everywhereëen grote troef is voor PWA's.

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. (**GoogleSupport2020**)

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. (**Developers2018**)

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 ' scoren beter dan traditionele websites. (**Anastasia2019**)

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. (**Scott2019**)

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. (**Osmani2017**)

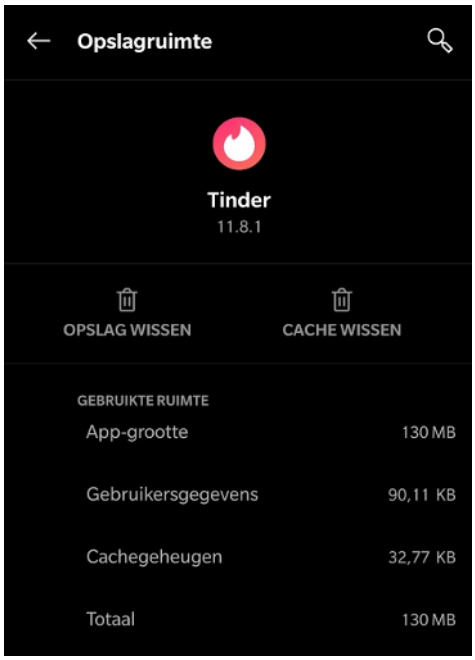
Een van de grootste voordelen is dat de PWA (versie 118) op het moment van schrijven slechts een grootte heeft van 397kb. De native Androidapplicatie (versie 11.8.1) daarentegen neemt 130mb in beslag op een toestel.

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.

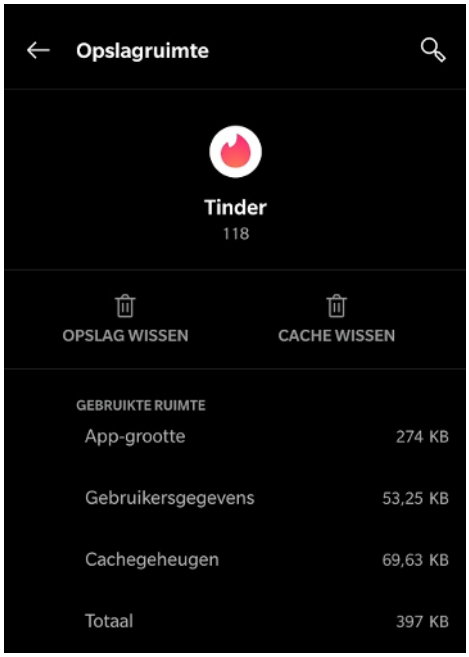
De gemiddelde prijs van de top 5 meest verkochte smartphones in Afrika was 135,6 USD. Deze toestellen beschikken vaak over slechts 8 of 16GB opslagruimte. (**netAdmin2017**)

Slechts 7% van Afrika beschikt over een 4g connectie. De andere gebieden moeten het vaak doen met een tragere 3g connectie. (**gsmArena2020**)

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



Figuur 2.14: Tinder voor Android - 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 functionaliteiten.

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. (**Developers2019a**)

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. (**Vanhala2017**)

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 (**Google2019**)

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. (**Gaunt2020**) (**Hiltunen2018**)

Als een webapplicatie voldoet aan de criteria (zie hoofdstuk 2.1) 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 PWA's.

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 PWA's dus 1 applicatie geschreven worden die op verschillende platformen kan gebruikt worden.

Spotify implementeerde zijn service ook als een PWA. (**Spotify2020**) Al de functies die op de native applicaties beschikbaar zijn, zijn ook beschikbaar via deze webapplicatie. Enkel de offlinefunctie is (nog) niet geïmplementeerd. Ook het gebruik van de functietoetsen op het toetsenbord van een desktop is niet ondersteund. (**Vu2019**)

2.3.8 Deployment

Het uitgeven van een PWA is ook een stuk gemakkelijker en goedkoper dan het verdelen van native applicaties via 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. (**Apple2020b**)

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. (**GooglePlay2020**)

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 de 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. (**Versio2020**)

Het hosten van een statische applicatie kan via bepaalde platformen kosteloos gebeuren. Voorbeelden van deze platformen zijn Firebase (**Firebase2020**) en Netlify (**Netflify2020**). 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 PWA's 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. (**Apple2020c**) (**GooglePlay2020a**)

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. (**Siddiqui2019**)

Dit is een vertraging die webapplicaties en PWA'S niet hebben.

Uit het interview met zowel Thomas Steiner als bij Wassim Chegham blijkt dat het onafhankelijk zijn van een app-store een grote meerwaarde is. Ze haalden beide aan dat de snelheid waarmee een PWA gehost en geüpdatet kan worden een uniek voordeel is.

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. (**Hume2018**)

Kleinere updates die de gebruiker niet zal opmerken, kunnen uitgevoerd worden zonder dat de gebruiker dit weet. (**Sanderson2020**)

Als er grotere updates zijn, is het een betere user experience om de gebruiker te informeren dat er een update zal uitgevoerd worden. (**Wicki2017**)

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. (**Apple2020d**) (**AndroidDevelopers2020**)

2.3.10 Conclusie

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

Tabel 2.52: concluderende tabel 'waarom een PWA'

Positief	Matig	Negatief

2.4 Beperkingen van een PWA

Zoals aangetoond in de sectie besturingssystemen en ' 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. (**Malavolta2016**)

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. (**Fransson2017**)

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 ', 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 '.

Zoals aangetoond in de sectie besturingssystemen en ' 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 hoofdstuk **??**). 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. (**Brousek2017**)

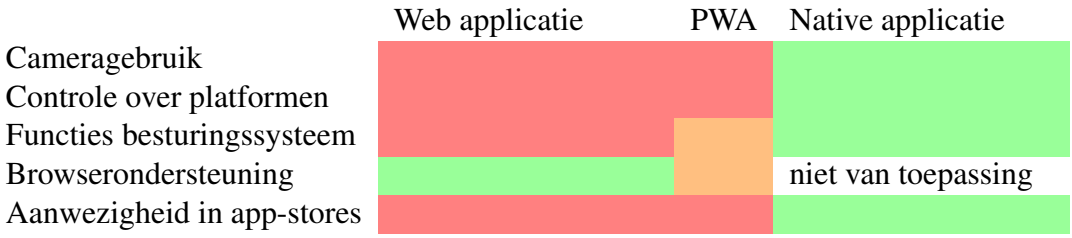
2.4.4 Browserondersteuning

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



Tabel 2.53: 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

(**Lighthouse2020**)

Voor elk van deze onderdelen, op 'PWA' 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.dev2020**)

Vervolgens worden de offline capaciteiten van de website getest. Er wordt getest 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:

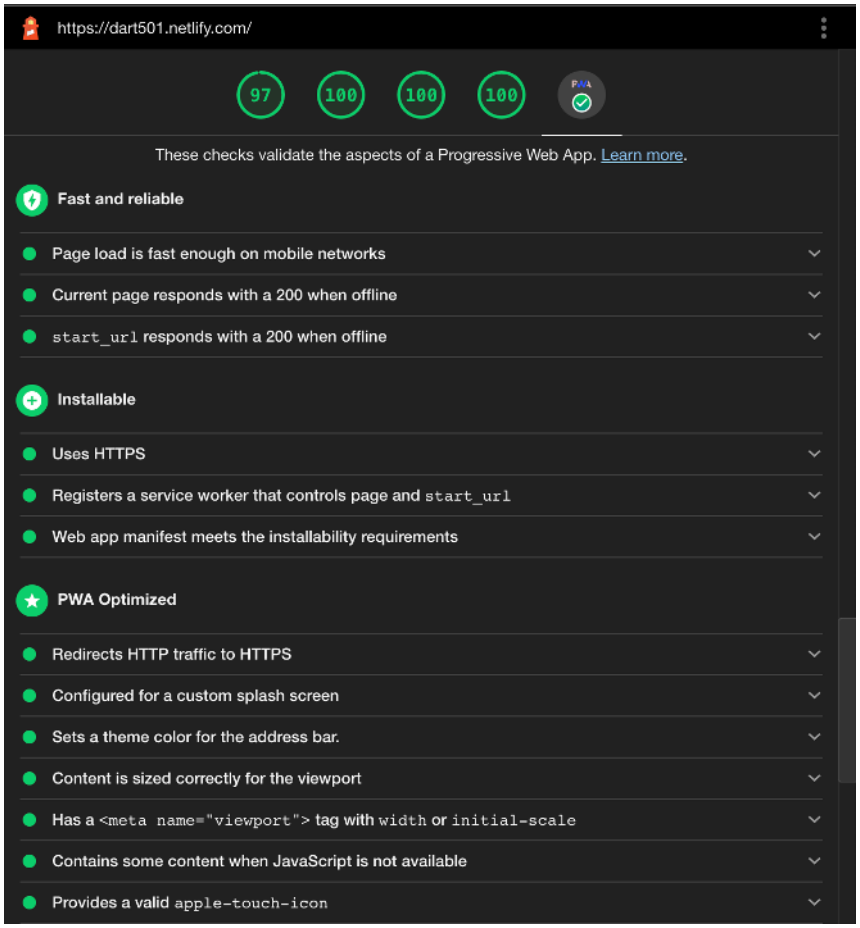
- Er moet een HTTPS-verbinding zijn.
- Er moet een service worker geregistreerd worden.
- Er moet een minimum app-manifest zijn.

(**web.dev2020a**)

PWA optimized

In dit onderdeel wordt er gecontroleerd op kleine optimalisaties die een betere gebruikers-ervaring 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 onderdeel 'PWA' in 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. (**Workbox2020**)

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

(**PWAbuilder2020**)

2.5.4 Chrome developer tools

De Chrome developer tools bieden een grote hulp bij het ontwikkelen van webapplicaties.

Chrome zorgt ervoor dat er eenvoudig gecachete 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

De chrome devloper tools hebben ook een sectie waar de status van de service worker kan bekeken en gemanipuleerd worden. (**Developers2019b**)

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 deze technologieën.

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 weergeven 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. (**Huynh2017**)

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 (**Ionic2020**)

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. (**Asp2017**)

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. Voorbeel-

den 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. (**Ionic2020a**)

De ontwikkelervaring van een Ionic applicatie is ook aangenaam omdat Ionic live reloading ondersteunt. Dit wil zeggen dat de applicatie niet opnieuw gebuild moet worden om de aanpassingen te zien. (**Lucas2020**)

Het Ionic framework zal er ook voor zorgen dat de applicatie meer native aanvoelt door de UI-elementen die het aanbiedt. Als een applicatie gebuild 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. (**Asp2017**)

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. 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. (**Ghinea2018**) (**DeConinck2019**)

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. (**Asp2017**)

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. (**Asp2017**)

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. (**Altexsoft2019**) (**Warcholinski2020**)

De IDE Visual studio voorziet veel tools die het ontwikkelen van Xamarin applicaties makkelijker maakt. (**VisualStudio2020**)

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. (**Microsoft2019**)

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. (**StackOverflow2020**)

React native

React native(**Reactnative2020**)) 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. (**Bartosz2019**)

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. (**StackOverflow2020**)

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. (**Expo2020**)

Flutter

Flutter (**Flutter2020**) 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. (**StackOverflow2020**)

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.

In een eerste fase moet er bekeken worden als het technisch mogelijk is om de applicatie te ontwikkelen als PWA. Als de applicatie functionaliteiten bevat die enkel beschikbaar baar zijn voor native applicaties, is een PWA geen optie meer.

Als er beslist wordt dat een PWA geen optie is, zijn er nog steeds twee keuzes: een hybride applicatie 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 was nog niet op punt staat bij native cross platform benaderingen. Ionic is het meest gebruiksvriendelijk platform om hybride applicaties te ontwikkelen.

Als de meeste gebruikers verwacht worden op native code, 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.

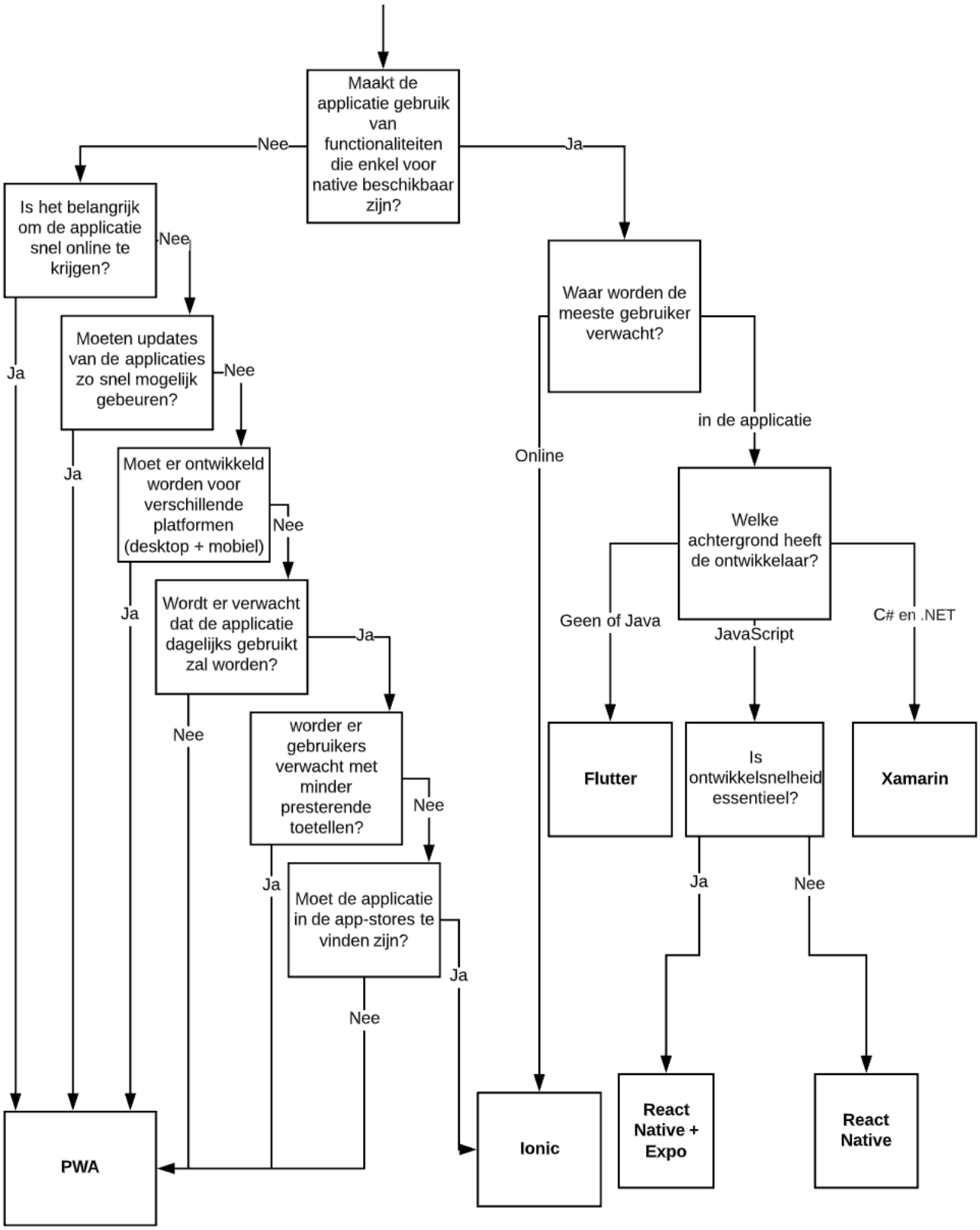
Voor ontwikkelaars met een C# achtergrond is een applicatie in Xamarin een vanzelfsprekende keuze.

Flutter is geschreven in een nieuwe taal, Dart, 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 in geen van deze talen of technologiën een achtergrond is, is Flutter een goeie keuze. (**Leler2017**) (**Wenhao2018**)

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 het onderzoek dat Jakub Fagietto voerde. (**Fagietto2019**)

2.6.4 Concluderende flowchart



Figuur 2.17: flowchart: technologieën met één codebase die op meerdere platformen gebruikt kunnen worden

2.7 Ontsluiten van een PWA

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

In deze sectie van het onderzoek zal er gekeken worden als er een hybride oplossing mogelijk is. Hierbij worden zoveel mogelijk functionaliteiten van de applicatie geïmplementeerd als PWA. Vervolgens kan er een native applicatie gemaakt worden die de ontbrekende functionaliteiten implementeert. Deze native applicatie kan de functionaliteiten die al ontwikkeld zijn in de PWA implementeren als een webview.

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

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

Your app should include features, content, and UI that elevate it beyond a repackaged website. If your app is not particularly useful, unique, or app-like, it doesnt belong on the App Store. (**Apple2020c**)

De Google play store heeft hier geen specifieke regels over. (**GooglePlay2020a**)

3. Methodologie

Hoofdstuk 2 was niet enkel onderbouwd aan de hand van de traditionele bronnen van een literatuurstudie. Er werd ook waardevolle informatie verzameld door een interview af te nemen met Thomas Steiner en Wassim Chegham. In deze interviews werd nogmaals duidelijk dat PWA's een heel sterk concept zijn dat relevant zal blijven de komende jaren binnen de developer community.

Thomas Steiner is een developer advocate bij Google. Hij schrijft regelmatig artikels die gepubliceerd worden op het web.dev platform. Deze artikels gaan vaak over moderne web-API's en PWA's. Ook publiceerde hij reeds verschillende artikels op Medium.

Wassim Chegham is een google developer expert en is lid van het Angular team. Momenteel is hij werkzaam bij Microsoft als senior cloud advocate. Hij ontwikkeldt ook mee aan verschillende open source projecten. Hij was een google developer expert op het moment dat Google voor het eerst de term PWA gebruikte. Hij gebruikt en onderzoekt PWA's sinds dan.

In beide interviews werden verschillende voordelen aangehaald. Zowel de ontwikkelkost als het de onafhankelijkheid van app-stores kwamen in beide interviews naar boven. Er werd ook gevraagd naar de nadelen van PWA's. het grootste probleem met PWA's is de verschillende en inconsistente ondersteuning van de browsers. Wassim Chegham maakte ook duidelijk dat PWA's niet enkel een technologie zijn die voor snellere en goedkopere productie kan zorgen. PWA's zijn een unieke technologie die ook andere strategische voordelen heeft voor een organisatie. Het is een grote troef dat een applicatie kan gebruikt worden zonder geïnstalleerd te moeten worden zoals bij een native applicatie. Volgens hem is het op die manier makkelijker om gebruikers te winnen. De conclusie van beide ontwikkelaars was dat PWA's een technologie zijn die voor elk project overwogen moet

worden. Beide zijn ook overtuigd dat PWA's steeds relevanter zullen worden en niet snel zullen verdwijnen.

De volledige interviews kunnen in de appendix A gevonden worden.

Na het voeren van de brede literatuurstudie in hoofdstuk 2 zullen er 2 cases uitgewerkt worden die meer inzicht zullen geven in de technologie.

Eerst zal er een bestaande webapplicatie omgevormd worden tot een PWA die installeerbaar is en die offline gebruikt kan worden. Dit is een belangrijk onderzoek omdat het aanwezig zijn op het startscherm tot een betere engagement kan leiden. Een gebruiker die een applicatie heeft toegevoegd aan zijn startscherm zal gemiddeld meer en langere sessies hebben dan bij een traditionele webapplicatie. (**LePage2020b**) Door concreet te documenteren welke stappen er nodig zijn zijn om een traditionele web applicatie om te vormen tot een PWA kunnen ontwikkelaars en organisaties beter inschatten hoelang dit zal duren. Op basis van deze inschatting kunnen ze een betere beslissing maken als ze dit willen implementeren of niet. Om een website te kunnen installeren op een toestel moet er voldaan worden aan drie voorwaarden:

- Een HTTPS-connectie hebben
- Een app manifest hebben
- Een service worker registreren

Dit zal gecontroleerd worden aan de hand van een lighthouse audit.

Vervolgens zal er ook een proof-of-concept uitgewerkt worden waarin moderne webtechnologieën gebruikt worden om een PWA te ontwikkelen met een video-bel functionaliteit. In deze proof of concept zal onderzocht worden als een PWA een native waardige ervaring kan bieden en als er gevallen zijn waar een gebruiker een PWA verkiest boven een native applicatie. De concepten besproken in hoofdstuk 2 zullen in deze proof-of-concept ook toegepast worden, voorbeelden hiervan zijn de application shell en progressive enhancement.

Beide applicaties zullen getest worden door de tester, vervolgens zullen er enkele vragen gesteld worden over de ervaring van de gebruiker. Door deze vragen te stellen wordt er verwacht dat volgende inzichten zullen verworven worden.

- Wat weet de gebruiker over wat een webapplicatie kan. Als een gebruiker niet weet dat het web een bepaalde functionaliteit ondersteund, zal hij hier ook geen webapplicatie voor zoeken maar een native applicatie.
- Welke rechten krijgt een PWA gemiddeld krijgt van een gebruiker.
- Hoe vaak wordt een PWA toegevoegd aan het startscherm. Deze vraag werd ook gesteld in de interviews, maar zowel Thomas Steiner als Wassim Chegham kon hier geen antwoord op geven.
- Een PWA verkozen wordt boven een native app. als de gebruiker niet weet wat een PWA is.
- EEn PWA verkozen wordt boven een native app, als de gebruiker alle voordelen van een PWA kent.

- Een PWA kan voldoen aan de verwachtingen van een gebruiker van een native app.

De gestelde vragen en de verkregen antwoorden kunnen in appendix C gevonden worden.

Aan de hand van de literatuurstudie, de interviews en de praktische onderzoeken zullen de onderzoeksvragen beantwoord worden

4. Ombouwen van een web app tot een PWA

Voor sommige webapplicaties is het een grote meerwaarde om geïnstalleerd te kunnen worden en offline bruikbaar te zijn op het toestel van de gebruiker.(**Mozilla2020c**) Eenvoudige applicaties die geen gebruik maken van backend verzoeken kunnen op deze manier volledig functioneel zijn zonder internetconnectie. Aan de hand van dit voorbeeld zal de onderzoeksvraag "Welke stappen zijn nodig om een traditionele website om te vormen tot een PWA?"beantwoord worden.

4.1 De applicatie

Darts is een spel dat gespeeld wordt met drie pijltjes per persoon en een dartbord. De plaats waar het pijltje landt op het bord bepaalt de score van de worp, dit kan variëren van 0 tot 60. De speler die het eerst een score van 501 bij elkaar gooit is de winnaar. Deze scores optellen en bijhouden wordt al snel ingewikkeld. Darts is een spel dat vaak gespeeld wordt in een bar, spelers willen dus niet geconcentreerd moeten rekenen. Spelers zullen dus vaak op zoek gaan naar hulpmiddelen. Dit kan een combinatie zijn van pen en papier met een rekenmachine zijn. Een alternatief is een applicatie die dit doet voor hen. Bij deze applicatie zal deze service aangeboden worden als een PWA. De applicatie kan dus gebruikt worden in de browser, maar de applicatie kan ook gedownload worden.

4.2 Analyse

Er werden user-stories opgesteld zodat de requirements van de applicatie duidelijk werden.

4.2.1 Functionele requirements

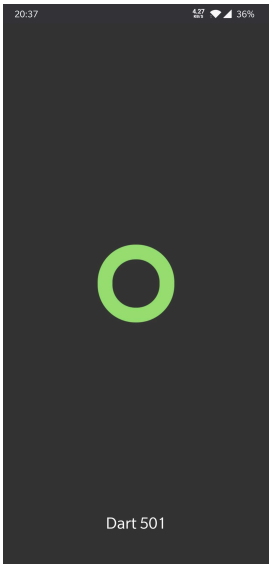
- Als een speler wil ik de score van een dartspel op mijn smartphone kunnen bijhouden zodat ik altijd en overal kan spelen.
- Als een speler wil ik de score van een spel kunnen bijhouden zonder een applicatie te downloaden zodat mijn smartphone niet vol komt met applicaties die ik niet regelmatig gebruik.
- Als een speler wil ik de score van een dartspel bijhouden zonder een internetverbinding te hebben zodat ik niet afhankelijk ben van een eventuele wifi of een netwerkverbinding.
- Als eent groep vrienden willen we een dartspel kunnen starten en de score kunnen bijhouden voor een variabel aantal personen zodat we snel kunnen spelen.

4.2.2 Niet-functionele requirements

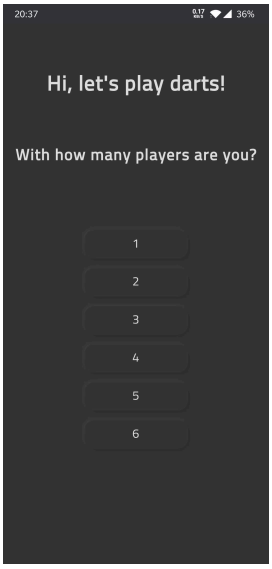
- Als een ontwikkelaar wil ik aan alle voorwaarden van de ligthouse audit op vlak van PWA's voldoen zodat de applicatie volledig geoptimaliseerd is.
- Als een ontwikkelaar wil ik aan alle voorwaarden van de ligthouse audit op vlak van PWA's voldoen zodat de applicatie volledig geoptimaliseerd is.
- Als een gebruiker wil ik dat alle pagina's onmiddellijk geladen worden zodat ik niet hoef te wachten.

4.3 Implementatie

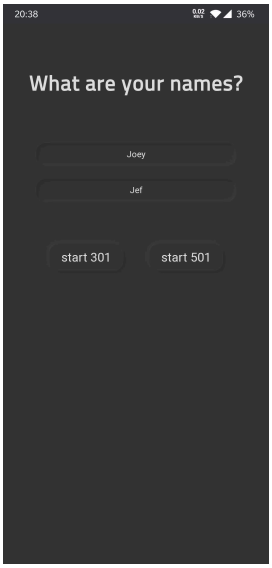
De applicatie werd eerst ontworpen aan de hand van figma, dit is een tool die gebruikt wordt om designs en prototypes te maken. De applicatie zal drie schermen hebben: een die het aantal speler vraagt, een die de spelersnamen vraagt en een scherm waar het spel kan bijgehouden worden. Voor de ontwikkeling van de applicatie werd de javascript library React.js gekozen.



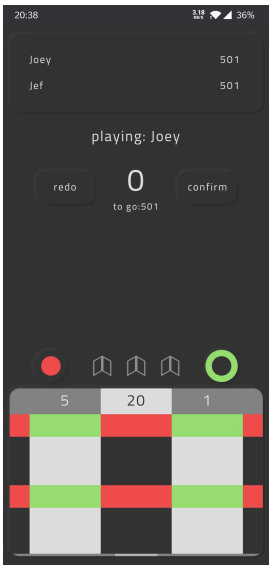
Figuur 4.1: splashscreen



Figuur 4.2: startscherm - vragen naar het aantal spelers



Figuur 4.3: vragen naar de namen van de spelers



Figuur 4.4: het scherm waar de score geteld zal worden

Het doel van deze proof-of-concept is het onderzoeken van de onderzoeksvraag "Welke stappen zijn nodig om een traditionele website om te vormen tot een PWA".

In het begin van deze thesis werd een PWA gedefinieerd als een webapplicatie die gebruik maakt van moderne webtechnologieën om op deze manier een ervaring aan te bieden die dicht die van een native applicatie ligt.

Deze definitie stelt geen duidelijke criteria op waaraan om website moet voldoen om een PWA te zijn.

Voor deze proof-of-concept zal een webapplicatie geoptimaliseerd worden tot deze geïnstalleerd kan worden en deze offline gebruikt kan worden.

De code van deze applicatie kan gevonden worden via github: https://github.com/TijsM/Dart501

4.4 A2HS

4.4.1 HTTPS

Een van de drie vereisten om een applicatie te kunnen toevoegen aan het startscherm is dat er een https-connectie moet zijn. Dit werd bekomen door de website te hosten op Netlify. Dit is een online service die gratis hosting voor statische websites aanbiedt. Hier kan een SSL-certificaat toegevoegd worden zodat er een HTTPS connectie tot stand komt.

4.4.2 App manifest

```
{
  "short_name": "Dart 501",
  "name": "Dart 501",
  "icons": [
    {
      "src": "/icons/android-icon-36x36.png",
      "sizes": "36x36",
      "type": "
      /png",
      "density": "0.75"
    },
    {
      "src": "/icons/android-icon-48x48.png",
      "sizes": "48x48",
      "type": "image/png",
      "density": "1.0"
    },
    {
      "src": "/icons/android-icon-72x72.png",
      "sizes": "72x72",
      "type": "image/png",
      "density": "1.5"
    },
    {
      "src": "/icons/android-icon-96x96.png",
      "sizes": "96x96",
      "type": "image/png",
      "density": "2.0"
    },
    {
      "src": "/icons/android-icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png",
      "density": "3.0"
```

```
    },
    {
      "src": "/icons/android-icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png",
      "density": "4.0"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#95df71",
  "background_color": "#363636"
}
```

De naam en de verkorte naam (naam die op splashscreen getoond zal worden) worden als eerste vastgelegd. Deze zijn gelijk omdat de volledige naam al kort genoeg is. Het maximumaantal karakters voor 'short_name' is 12.

Vervolgens worden alle iconen voor de verschillende platformen bepaald. Dit deel van het manifest en de bestanden waarnaar het refereert kunnen gegenereerd worden met de tool appicon.co.

Vervolgens wordt het gedrag en het uitzicht van de applicatie gedefinieerd. Er wordt ingesteld dat de applicatie start op het scherm waar het aantal spelers geselecteerd kan worden, er wordt ook bepaald dat de applicatie het volledige scherm in beslag moet nemen.

Ook het kleurenschema wordt bepaald, deze kleuren worden gebruikt bij het splashscreen en bepalen ook de kleur die de statusbalk van het toestel zal innemen.

Er moet een link gemaakt worden van de html-bestanden naar dit manifest bestand. Dit gebeurt binnen de head tag van de index.html file

```
<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
```

4.4.3 Service worker

Als een react applicatie gecreëerd wordt met het 'create-react-app' commando wordt er een basis service worker aangemaakt. Standaard wordt deze niet gebruikt, om dit te veranderen moet de regsiter methode van de service worker aangeroepen worden in het index.js bestand.

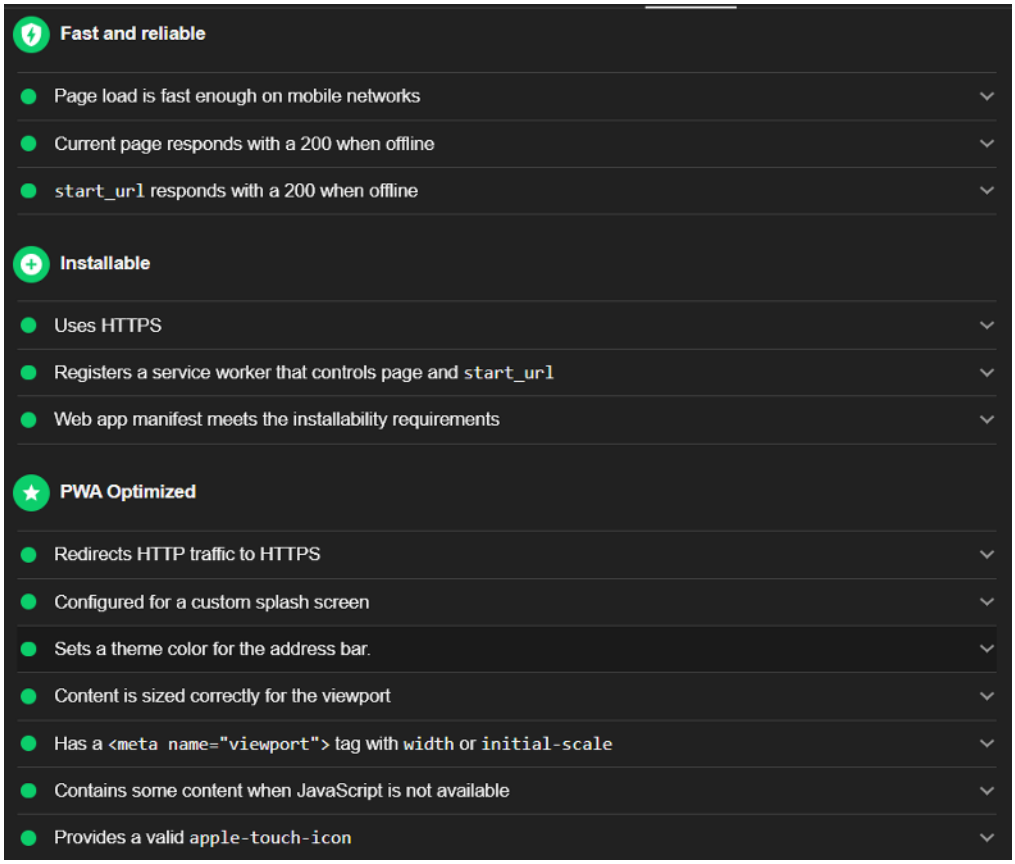
```
    serviceWorker.register();
```

Deze service worker zorgt ervoor dat alle statische bestanden offline beschikbaar gemaakt worden en dat de gebruiker op een Android toestel de vraag krijgt om deze applicatie aan het startscherm toe te voegen.

Voor deze applicatie biedt deze service worker genoeg functionaliteit.

4.5 Controle

Aan de hand van een lighthouse audit kan er gecontroleerd worden als alles werkt zoals verwacht.



Figuur 4.5: lighthouse audit van de website

Onder de sectie installable kunnen we zien dat de applicatie inderdaad een https-connectie heeft, dat er een service worker is en dat er een manifest aanwezig is.

De pagina's worden ook onmiddelijk geladen, dit komt omdat alle bestand lokaal gecached zijn en er dus niet op een antwoord van de server gewacht moet worden.

4.6 Conclusie

We kunnen concluderen dat een traditionele website snel kan omgevormd worden tot een PWA die geïnstalleerd kan worden op het startscherm van de gebruiker.

Zoals eerder aangehaald moet een website aan volgende 3 zaken voldoen om geïnstalleerd te kunnen worden:

- Een HTTPS-connectie hebben
- Een app manifest hebben
- Een service worker registreren

Een HTTPS-connectie opzetten gebeurt bij verschillende hosting services automatisch en gratis. Dit is dus eenvoudig op te zetten.

Het app manifest is een JSON-bestand die de eigenschappen van de applicatie beschrijft. Door gebruikt te maken van bepaalde tools kan dit ook snel gecreëerd worden.

Als een webapplicatie ontwikkeld wordt met React.js kan ook een serviceworker eenvoudig geregistreerd worden. Er moet slecht één methode aangeroepen worden.

Een webapplicatie kan dus heel snel en eenvoudig omgezet worden tot een PWA.

Dit is ook interessant voor applicaties die wel gebruik maken van een backend. Het cachen van bestanden heeft volgende positieve gevolgen:

- De pagina's worden onmiddelijk geladen
- De applicatie toont geen error als de gebruiker eventjes geen connectie meer heeft met een netwerk. Dit kan gebeuren als de gebruiker in zich in een tunnel of in de metro begeeft.

5. Proof-of-concept

In deze proof-of-concept zal er onderzocht worden als een PWA kan voldoen aan het verwachtingspatroon van een gebruiker op een native applicatie.

Er zal een applicatie ontwikkeld worden waarbij een aangemelde gebruiker een andere persoon zal kunnen uitnodigen om een video gesprek te hebben.

5.1 Analyse

5.1.1 Functionele requirements

Authenticatie

- Als een gebruiker wil ik me kunnen registreren met externe platformen (facebook, google, ...) zodat ik minder wachtwoorden hoef te onthouden.
- Als een aangemelde gebruiker wil ik aangemeld blijven als ik de applicatie sluit zodat ik tijd bespaar door niet niet bij elk gebruik te moeten aanmelden.

Bellen

- Als een aangemelde gebruiker wil ik een 'room' aanmaken waar ik een video-gesprek kan starten zodat ik in contact kan blijven met iemand waar ik niet fysiek bij kan zijn.
- Als een aangemelde gebruiker wil ik eenvoudig een link kunnen kopiëren naar mijn room zodat ik deze kan delen met iemand.

- Als een aangemelde gebruiker wil ik mijn 'rooms' kunnen beheren.
- Als een aangemelde gebruiker wil ik een notificatie ontvangen als iemand in mijn 'room' komt.

Gebeld worden

- Als een gebruiker wil ik, als ik een link van een room ontvang, eenvoudig deelnemen aan een videogesprek.

Algemeen

- Als een gebruiker wil een de toepassing toevoegen aan het startscherm van mijn toestel zodat ik deze snel kan openen.
- Als een aangemelde gebruiker wil ik de applicatie kunnen openen zonder internetverbinding zodat ik mijn overal en altijd mijn rooms kan bekijken.

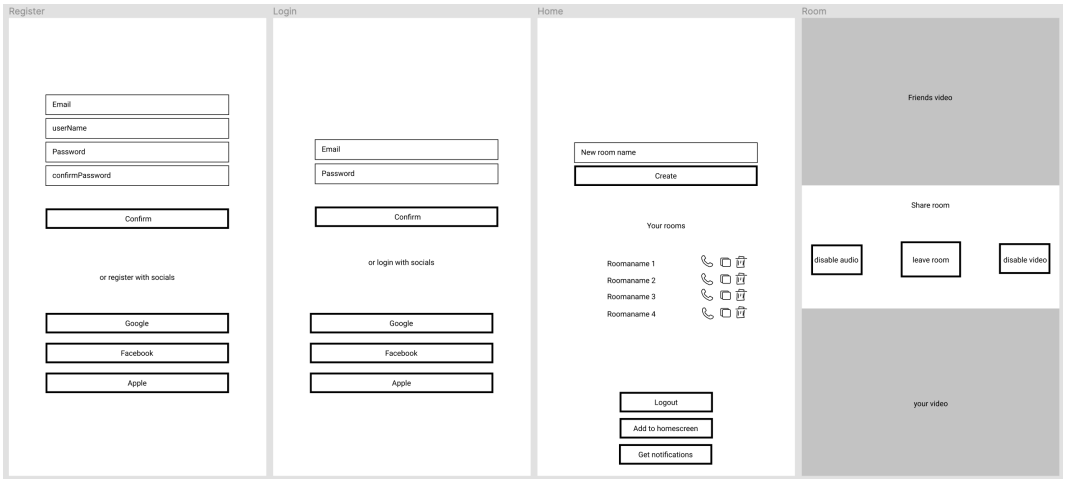
5.1.2 Niet-functionele requirements

- Als een gebruiker wil ik een video-gesprek kunnen hebben met een zo laag mogelijke vertraging zodat ik geen problemen heb met communiceren.
- Als een gebruiker wil ik video-gesprekken voeren op een veilige manier zodat niemand informatie kan achterhalen die ik niet publiek wou maken.

5.2 Design

5.2.1 Mockup

Voor er een design gemaakt werd, werden er mockups opgesteld. Aan de hand van deze mockups met weinig detail, kon er snel gestart worden met het ontwikkelen van de functionaliteiten. (**Tate2019**)

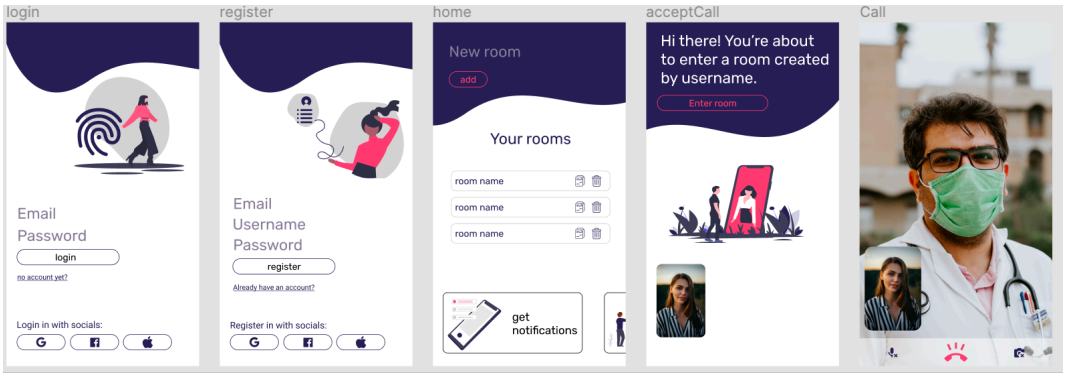


Figuur 5.1: mockups in figma

5.2.2 Prototype

Vervolgens werd er een klikbaar prototype gemaakt. In de dit prototype werd de stijl van de applicatie bepaald.

Er werd gekozen om een eenvoudig design te implementeren met illustraties die van unDraw verkregen werden.



Figuur 5.2: prototype in figma

5.3 Implementatie

Voor het ontwikkelen van de proof-of-concept werd er opnieuw react.js gebruikt voor de frontend. De backend werd ontwikkeld in node.js en als database werd de firestore van firebase gebruikt.

De code van dit project kan op github gevonden worden: https://github.com/TijsM/videocall-pwa

5.3.1 Registreren service worker

In Hoofdstuk 4 werd vermeld dat een standaard react applicatie al een service worker heeft die voor caching van statische bestanden zorgt. Deze service worker wordt gegenereerd tijdens het build proces door de library sw-precache-webpack-plugin. (**Mester2019**)

Door een service worker te genereren zorgt react ervoor dat het heel eenvoudig is om statische bestanden te cachen en een website offline beschikbaar te maken.

Deze strategie heeft echter één groot nadeel, de service worker die react genereert kan niet worden aangepast door de ontwikkelaar. Als de applicatie een meer geavanceerde service worker nodig heeft, zal deze automatisch gegenereerde service worker moeten overschreven worden.

Het artikel van Chinmaya Pati dat gepubliceerd werd op Medium beschrijft dit proces. (**Pati2019**)

5.3.2 A2HS

De applicatie voldoet aan alle criteria om geïnstalleerd te kunnen worden op toestellen. Echter zal de gebruiker geen standaard melding te zien krijgen om de applicatie te installeren.

Als de applicatie de gebruiker vraagt om de de applicatie toe te voegen aan het startscherm en de gebruiker staat dit niet toe, zal het beforeInstallPrompt nooit meer opnieuw opgeroepen worden. Deze melding kan dus maar 1 maal getoond worden.

Om de kans te verhogen dat de gebruiker de applicatie toevoegt aan het startscherm zal deze melding enkel getoond worden als de gebruiker daar expliciet naar vraagt. (**Mclachlan2020**)

Als de applicatie opstart zal het beforeInstallPrompt opgevangen worden en opgeslagen worden in de state. Door het standaard gedrag te overschrijven zal de melding nog niet getoond worden aan de gebruiker. (**LePage2020b**)

```
1 window.addEventListener("beforeinstallprompt", (event) => {
2   event.preventDefault();
3   setPrompt(event);
4 });
```

Vervolgens kan, als de gebruiker dit wenst, de melding toch getoond worden.

```
1 const addToHome = () => {
2   prompt.prompt();
3 };
```

5.3.3 Caching

Cache strategieën

Cache first Er zal eerst naar het cache geheugen gekeken worden, als het gevraagde bestand hier niet aanwezig is zal er een netwerk verzoek verstuurd worden.

Dit is ee populaire strategie als de data weinig veranderd.

Network first Er zal eerst een netwerk verzoek verstuurd worden, als deze niet succesvol was zal er gekeken worden als het bestand in het cache geheugen aanwezig is.

Dit is een populaire strategie als de data vaak veranderd.

Network only Deze strategie wordt toegepast als er geen caching opgezet wordt.

Cache only Deze strategie zal nooit een netwerkverzoek versturen. Er kan enkel data opgehaald worden via het cache geheugen.

Dit cache geheugen moet door de applicatie gevuld worden.

Stale-While-Revalidate De staleWhileRevalidate strategie houdt in dat eerst het gecachete bestand gebruikt wordt, vervolgens wordt er een netwerkverzoek verstuurd. Als het lokale bestand en het bestand van de server niet volledig gelijk zijn, zal het gecachte bestand overschreven worden. En zal de content op in de applicatie aangepast worden.

Dit is de meest populaire strategie omdat de pagina snel geladen wordt, en de inhoudt is up to date.

Statisch cachen

Statisch cachen is het cachen van bestanden die nodig zijn voor het opbouwen van de webapplicatie. Voorbeelden hiervan zijn foto's, style sheets, javascript bestanden, html bestanden, ...

Op deze manier zal ook de application shell offline beschikbaar zijn.

Door deze bestanden te cachen wordt de applicatie offline beschikbaar.

In deze applicatie werd dit bereikt door workbox te gebruiken.

In een eerste fase wordt het index.html bestand gecached.

```
1 workbox.precaching.precacheAndRoute([
```

```
2   {url: '/index.html', revision: '1'},
3 ]);
```

Workbox zal de functie 'precacheAndRoute' uitvoeren als de service worker geïnstalleerd wordt. De bestanden die worden meegegeven in de array zullen vanaf dat moment offline beschikbaar zijn. (**Workbox2020a**)

Bestanden die hier toegoevoegd worden zullen geleverd worden aan de hand van de 'cache first' strategie. Dit wil zeggen dat dit bestand enkel aan de server gevraagd zal worden als het niet beschikbaar is in het cache geheugen.

Een andere manier waarop statische caching kan gedaan worden is aan de van de 'registerRoute' die Workbox aanbiedt.

```
1  workbox.routing.registerRoute(
2    /\.(?:js|css)$/,
3    workbox.strategies.staleWhileRevalidate({
4      cacheName: "static-resources",
5      plugins: [
6        new workbox.expiration.Plugin({
7          maxEntries: 60,
8          maxAgeSeconds: 20 * 24 * 60 * 60, // 20 Days
9        }),
10     ],
11   })
12 );
```

Aan de hand van een regular expression kunnen er bestanden gechachet worden. In dit voorbeeld worden alle javascript en css bestanden offline beschikbaar gemaakt.

In dit voorbeeld wordt de Stale-While-Revalidate strategie toegepast. (**Workbox2020b**)

dynamisch cachen

Dynamisch cachen houdt in dat data die de gebruiker vraagt ook offline opgeslagen wordt. Dit zullen vaak requests van REST-API's zijn.

In deze applicatie werd dynamisch cachen toegepast om de laatste versie van de rooms nog steeds te tonen aan de gebruiker, ookal als deze offline.

```
1  workbox.routing.registerRoute(
2    /.*(?:googleapis|)\.com.*$/,
3    workbox.strategies.staleWhileRevalidate({
4      cacheName: "firebase",
5    })
6  );
```

Opnieuw wordt er gebruik gemaakt van een regular expression om workbox duidelijk te maken welke request er offline beschikbaar moet zijn.

5.3.4 Authenticatie

Het invullen van van een form om een mobiel toestel is voor veel gebruikers een relatief grote inspanning.

In het artikel van Nick Babich wordt de term ïnteraction cost"gebruikt. De interaction cost is de som van alle cognitieve en fysieke inspanningen die een gebruiker moet leveren om tot zijn doel te komen. Hoe lager de interaction cost hoe beter. (**Babich2018**)

Het invullen van formulieren met data heeft een hoge interaction cost. Er moet veel getypt worden (fysieke inspanning). De gebruiker moet hierbij aandachtig zijn om zeker te zijn dat hij geen fouten maakt (cognitieve inspanning). het is dus belangrijk om de gebruiker hiervan te besparen.

Dit kan gedaan worden door de gebruiker te laten registreren via een ander platform, hierbij moet er vaak helemaal niet, of veel minder getypt worden.

Op deze applicatie kan de gebruiker zich aanmelden met Google en Facebook.

Dit werd bereikt door gebruikt te maken van "Firebase authentication".

```
1 export const auth = firebase.auth();
2 const provider = new firebase.auth.GoogleAuthProvider();
3
4 export const authWithGoogle = () => {
5   return auth.signInWithPopup(provider)
6 };
```

Deze code geeft alle relevante informatie terug die google over de gebruiker heeft. Dit bevat volgende eigenschappen:

- Voornaam
- Familienaam
- Volledige naam
- Email
- Profielfoto
- Voorkeurstaal
- ...

Gelijkaardige code kan geschreven worden voor het aanmelden met andere platformen zoals:

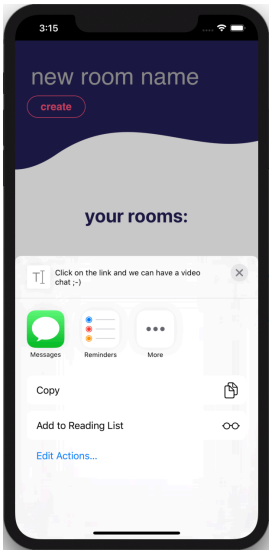
- Twitter
- Github
- Apple
- Microsoft
- ...

5.3.5 Share API en Clipboard API

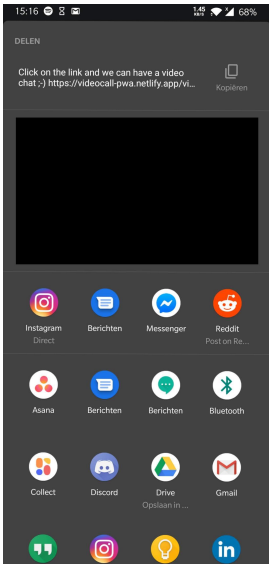
In de applicatie is het belangrijk om op een intuïtieve manier een link te kunnen delen naar een room. Dit werd bekomen door de Share API te gebruiken. Op mobiele toestellen zal deze een native menu openen waar de gebruiker de link kan delen via verschillende platformen.

Echter bieden enkel chrome voor android en safari dit aan. Er kan er dus niet vanuit gegaan worden dat de gebruiker dit kan implementeren.

```
1  const shareData = {
2        title: 'Join my room!',
3        text: 'Click on the link and we can have a video chat ;-)',
4        url: `https://videocall-pwa.netlify.app/visitroom/${owner}/${
             room}`,
5     }
6     navigator.share(shareData)
```



Figuur 5.3: native share-menu op IOS



Figuur 5.4: native share-menu op Android

Als de gebruiker een browser gebruikt die deze functie niet ondersteunt, zal de clipboard API gebruikt worden.

De clipboard API kan gebruikt worden om items in het klembord van de gebruiker te plaatsen of om het klembord uit te lezen.

Deze web-API werd gebruikt om eenvoudig de link van de room te kunnen delen met een persoon die gecontacteerd zal worden.

Volgende code zal de link genereren die de uitgenodigde persoon kan gebruiken en zal deze in het klembord van de gebruiker plaatsen.

De eerste keer dat dit proces uitgevoerd wordt zal de gebruiker expliciet toestemming moeten geven om het klembord te mogen gebruiken.

```
1 navigator.clipboard
2   .writeText(`http://localhost:3000/visitroom/${roomownername}/${
      roomname}`)
3   .then(() => {
4     alert("coppied!! share the coppied link with somebody");
5   });
6 };
```

Progressive enhancement

Bij deze feature werd er progressive enhancement toegepast. In het beste geval wordt het native deel-menu getoond, als dit niet beschikbaar is zal de link gekopiëerd worden in het klembord. Als ook dit niet beschikbaar is, of de gebruiker heeft geen toestemming gegeven zal er een boodschap getoond worden waar de gebruiker de link zelf kan kopiëren en delen.

```
if('share' in navigator){jesu
  // ... code voor het delen via een native menu ...
}
else{
  //... code voor het kopiëren van de link ...
  .catch(() => {
    //tonen van een boodscahp met de link die gedeeld moet worden
    swal(`you can share this link:
      https://videocall-pwa.netlify.app/visitroom/${owner}/${room}
    `);
  });
}
```

5.3.6 Push notificaties

Een van de grote voordelen van een PWA ten opzichte van een traditionele website is dat het push notificaties kan sturen naar de gebruiker.

Er wordt aangeraden om de "web-push"library te gebruiken om push notificaties te implementeren

In deze applicatie worden push notificaties gebruik in twee use-cases.

- Als een bezoeker in een room komt, zal de eigenaar van die room een meding krijgen
- De administrator van de applicatie kan een notificatie naar alle gebruikers versturen aan de hand vanuit een admin scherm

Abbonneren op push notificaties

De gebruiker moet expliciet toegang geven aan de webapplicatie om push notificaties te mogen gebruiken.

```
const sw = await navigator.serviceWorker.ready;
let pushSub = await sw.pushManager.subscribe({
      userVisibleOnly: true,
      applicationServerKey: 'public VAPID key'
})
```

Dit stukje code zal een subscription teruggeven dat opgeslagen moet worden in de databank bij de gebruiker. deze subscription is uniek voor de gebruiker, de browser en het toestel. (**Gaunt2019a**)

voorbeeld van een subscription:

```
{
   "endpoint":"https://fcm.googleapis.com/fcm/send/fOkirMCCHhM:..."
   "expirationTime":null,
   "keys":{
      "p256dh":"BBwYLrPvd2IIAo4cWsJhXRD2g9aFyL1Q3K9cuWh_...",
```

```
6         "auth":"0orvSHRZrMb3zP0yIpUAcg"
7     }
8 }
```

Backend

De backend moet in een eerste fase gebruikt worden om de VAPID keys te genereren.

Vapid keys of application server keys zijn uniek voor de server.

Deze vapid keys zorgen ervoor dat de push services van de verschillende browsers weet van welke applicatie de notificaties afkomstig zijn. Hierdoor kan het ook de applicatie beveiligen en enkel deze server toestaan notificaties te sturen naar de applicatie.

Elke browser heeft zijn eigen push service waar een applicatie server aan kan vragen om een notificiatie te sturen naar een bepaald toestel. (**Gaunt2020**)

Deze vapid keys kunnen in de applicatie server gegenereerd worden aan de hand van volgende code:

```
1 console.log(push.generateVAPIDKeys)
```

Dit zal een publieke en een private sleutel weergeven in de console. Deze moeten dan op een veilige plaats opgeslagen worden.

Eens de server de Vapid keys heeft kan deze een notificatie verzenden.

```
1 push.setVapidDetails(
2     "mailto:mai",
3     secrets.vapIdKeys.publicKey,
4     secrets.vapIdKeys.privateKey
5   );
6
7  push.sendNotification( sub,  JSON.stringify({payload})
```

Er moet ook een geldig email adres opgegeven worden. Dit email adres zal gebruikt worden door de push service als er problemen zijn met jouw applicatie.

De push notificatie kan dan verzonden worden met eventueel extra informatie in de payload.

Service worker

Op basis van de subscription die in de backend werd verzonden zal de service worker van de juiste gebruiker geactiveerd worden.

De service worker zal aan de hand van een 'eventListener' geactiveerd worden en een notificatie tonen op het toestel.

```
1 self.addEventListener("push", function (e) {
```

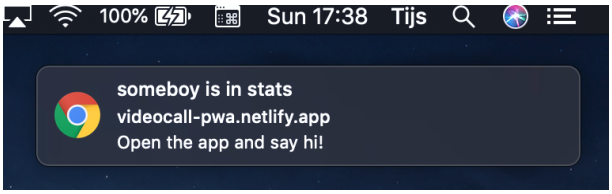
```
2   //handle notification
3 }
```

Als de backend extra informatie heeft meegegeven kan deze uitgelezen worden.

```
1   const payload = JSON.parse(e.data.text())
```

Vervolgens kan er een notificatie gevormd worden zoals eerder toegelicht in hoofdstuk 2 van deze scriptie.

```
1 options = {
2       body: "Open the app and say hi!",
3       icon: "images/example.png",
4       vibrate: [300, 50, 100],
5       data: {
6         dateOfArrival: Date.now(),
7         primaryKey: "2",
8       },
9     };
10
11    e.waitUntil( self.registration.showNotification(‘notification
        title‘, options)
```



Figuur 5.5: notificatie op macOS

5.3.7 Local storage

Om een ervaring aan te bieden zoals op native applicaties is het belangrijk om de gebruiker aangemeld te houden tussen sessies.

Als een gebruiker zich aanmeldt, wordt deze opgeslagen in het local storage.

Zolang de gebruiker het local storage niet manueel verwijdert zal dit blijven bestaan. (**Mozilla2020d**)

5.3.8 Web RTC

Web RTC is een verzameling van Web API’s die real time commicatie op het web mogelijk maken.

In deze applicatie werd Web RTC gebruikt om:

- de camera van het toestel te gebruiken en een video-stream op het scherm te tonen.

- een peer-to-peer connectie op te zetten tussen twee gebruikers
- video tussen beide gebuikers te streamen

(**webRTC2020**)

Web RTC maakt gebruik van een peer-to-peer connectie tussen de twee gebruikers, tijdens het bellen is er dus geen server nodig. Echter, er is wel een server nodig om de connectie tussen beide tot stand te brengen.

video-stream

De video-stream zal verkregen worden via het navigator object. Bij het opvragen van een video-stream zal de browser automatisch toestemming vragen aan de gebruiker als de camera en de microfoon gebruikt mogen worden.

Als de applicatie toestemming gekregen heeft wordt er een video-stream verkregen. Deze stream zal opgeslagen worden in de state van de applicatie en zal in een HTML-5 video element geplaatst worden.

```
1 navigator.mediaDevices.getUserMedia({
2     video: true,
3     audio: true
4   }).then((stream) => {
5     setYourVideoStream(stream);
6     if (yourVideo.current) {
7       yourVideo.current.srcObject = stream;
8     }
9   });
```

peer-to-peer verbinding met streaming

Bij deze applicatie werd simple-peer gebruikt, dit is een abstractie van webRTC die het opzetten van peer-to-peer verbindingen eenvoudiger maakt.

Voor beide gebruikers zal er een peer object aangemaakt worden.

```
1 const peer = new Peer({
2   initiator: true,
3
4   stream: yourVideoStream,
5 });
```

vervolgens zullen er listeners toegevoegd worden aan deze objecten die geactiveerd zullen worden op basis van het andere peer object

```
1 peer.on("signal", (data) => {
2   // code wordt uitgevoerd als er een andere peer aangemaakt wordt
3 });
4
5 peer.on("stream", (stream) => {
```

```
6   // code wordt uitgevoerd als de andere peer start met het
        streamen van een video
7 });
```

Performantie

De performantie van de applicatie werd gemeten aan door het de statistieken van de video elementen op te halen waarint de stream getoond wordt. Bij beide tests was een macbook Pro de administrator van de meeting en werd een Oneplus6 toegevoegd aan het gesprek als gast. De statistieken werden gemeten op het toestel van de administrator.

	Zoom	PWA
gemiddelded aantal frames per seconde van ontvangende video	14	34
gemiddelded aantal frames per seconde van verzonden video	26	30
resolutie ontvangen video	480x640	480x640
resolutiie verzonden video	640x360	640x480

Tabel 5.1: performantie vergelijking met Zoom

5.3.9 Socket.io

Zoals eerder vermeld is er een server nodig om een peer-to-peer connectie tot stand te brengen.

Deze backend werd geschreven in node.js en werd gehost op glitch.

De library socket.io werd gebruikt websocktets te implementeren. Websockets zorgen ervoor dat een server met de frontend kan cummuniceren zonder dat deze er expliciet om vragen.

In traditionele applicaties krijgt de frontend enkel data als hij hier een request voor stuurt naar de backend, bij websockets kan de backend de frontend met data voorzien in real time. (**Mozilla2020e**)

Deze real-time communicatie is nodig voor het opzetten van een peer-to-peer connectie.

De backend zal de frontend informeren als er een gebruiker in een room komt, vervolgens kan er een peer-to-peer connectie opgezet worden.

Zowel de frontend als de backend kunnen een event sturen en naar events luisteren. Op deze manier wordt er gecomuniceerd.

5.3.10 Masked icons

Het startscherm van android toestellen kan in veel verschillende vormen komen. De producent van een android toestel kan zelf een thema ontwikkelen voor het startscherm.

In verschillende thema's zullen de icoontjes er anders uitzien, in sommige thema's zullen ze rond zijn, in andere vierkant met afgeronde hoeken.

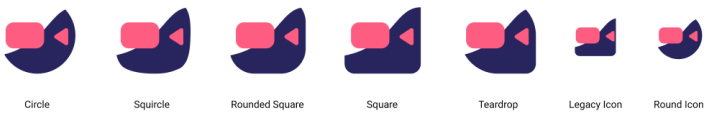
Als het icoontje niet geoptimaliseerd wordt, zal Android er steeds voor zorgen dat het icoontje volledig zichtbaar is. Indien nodig zal er een achtergrond toegevoegd worden.

Door ervoor te zorgen dat alle belangrijke elementen va het icoon binnen een 'safe zone' vallen, en een extra property toe te voegen in het app manifest kan dit opgelost worden.

```
1 "purpose": "any maskable"
```

Figuur 5.6: niet geoptimaliseerde iconen

Figuur 5.7: masked iconen

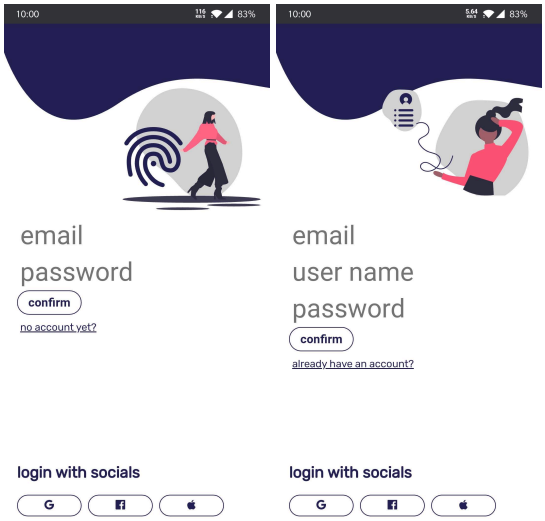


Figuur 5.8: Geoptimaliseerde iconen van proof-of-concept

5.4 De Applicatie

5.4.1 Authenticatie

De gebruiker kan zich registreren aan de hand van een email en wachtwoord combinatie. Er is ook een optie om zicht aan te melden aan de hand van Google en Facebook.



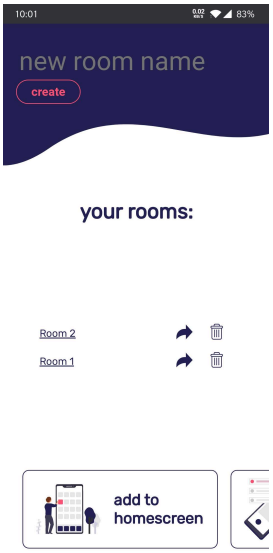
Figuur 5.9: autenticatie-schermen

5.4.2 Home

Dit is het scherm waar de gebruiker zal terechtkomen als hij is aangemeld. Op dit scherm kan de gebruiker zijn rooms beheren, hij kan nieuwe rooms aanmaken, rooms verwijderen en rooms delen.

Onderaan de pagina is er een horizontale scroll, in deze lijst kan de gebruiker volgende opties:

- De applictie installeren en toevoegen aan het startscherm
- Notificaties aan zetten
- Afmelden



Figuur 5.10: homescherm

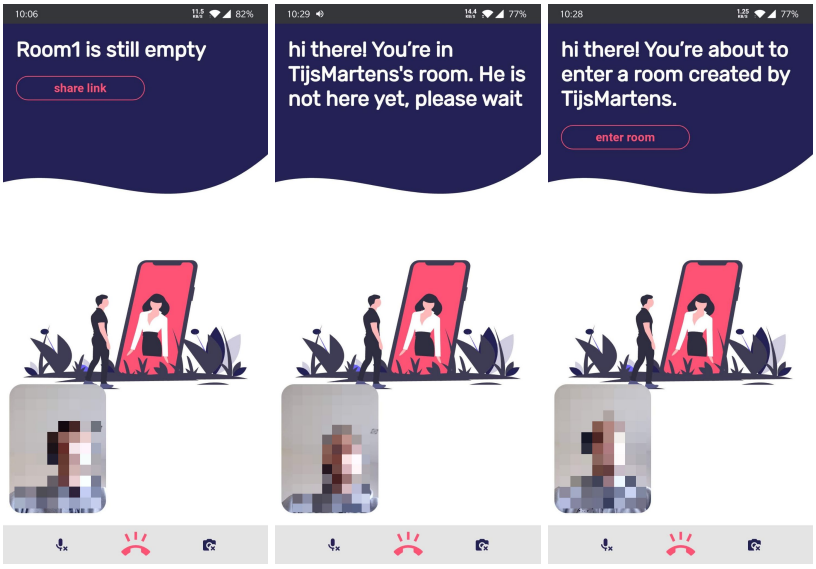
5.4.3 Wachtschermen

Als er slechts 1 persoon is in de room zal de gebruiker op dit scherm terechtkomen.

De eigenaar van de room heeft een optie om de room te delen.

Als een bezoeker in de room komt, zal de eigenaar een melding krijgen dat er iemand in zijn room is. Door op de notificatie te klikken gaat de eigenaar direct naar de room.

Als beide gebruikers aanwezig zijn, zal de bezoeker de optie krijgen om het gesprek te starten.

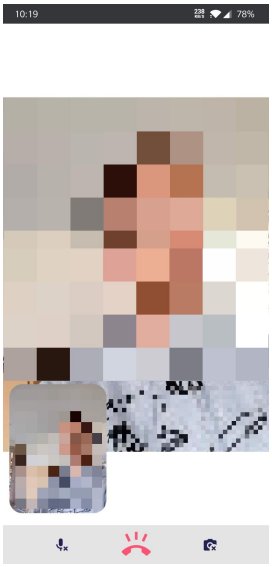


Figuur 5.11: wachtschermen

5.4.4 Room

In dit scherm kunnen de gebruikers een conversatie hebben.

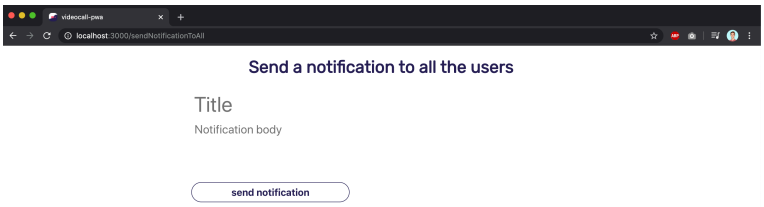
Er is een optie om het gesprek te beëindigen, het geluid uit te schakelen en de video stream uit te schakelen.



Figuur 5.12: belscherm

5.4.5 Notificaties

Op dit scherm kan de administrator van de applicatie een zelf gekozen notificatie versturen naar alle gebruikers die hun goedkeuring hebben gegeven om notificaties te ontvangen.



Figuur 5.13: scherm om manueel een notificatie te versturen

5.5 Conclusie

Al de vooropgestelde requirements kunnen bereikt worden door een PWA te implementeren. Native features zoals camera, microfoon, delen, notificatie zijn relatief eenvoudig te

implementeren in een webapplicatie.

5.5.1 Compatibiliteit en besturingssystemen

Zoals eerder aangehaald is een van de nadelen van PWA's de sterk variërende ondersteuning. Voor deze reden is het belangrijk dat de applicatie ontwikkeld wordt aan de hand van progressive enhancement.

De hoofdfunctionaliteit van de applicatie moet beschikbaar zijn voor alle gebruikers. Vervolgens kan de applicatie uitgebreid worden met functionaliteiten die de ervaring verbeteren maar die niet noodzakelijk zijn.

In deze applicatie is de hoofdfunctionaliteit het videobellen. Zoals aangetoond in sectie 2.2 wordt webRTC ondersteund op alle veelgebruikte browsers en toestellen. Er werden vervolgens extra functionaliteiten toegevoegd zoals het delen van de link via het native deel-menu, de clipboard API, push notificaties en het automatisch toevoegen aan het startscherm.

- Het deel-menu is enkel beschikbaar op mobiele toestellen en safari op macOS.
- De clipboard API is beschikbaar op alle browsers en toestellen maar heeft wel de toestemming van de gebruiker nodig.
- Push notificaties zijn beschikbaar op alle browsers en toestellen met uitzondering van:
  - alle browsers op IOS
  - safari op macOS
- automatisch toevoegen aan het startscherm is beschikbaar op alle browsers en toestellen met uitzondering van:
  - alle browsers op IOS
  - safari op macOS

5.5.2 Beperkingen van een PWA

Het is niet eenvoudig om consistent een goeie ervaring te bieden met één codebase voor alle platformen die ondersteund worden.

Er werd beslist om de maximale hoogte van de applicatie vast te zetten op 100vh, op deze manier is het in principe onmogelijk om een scroll van de volledige pagina te hebben.

dit werkte goed op een desktop browser maar gaf problemen op mobiele toestellen.

Op Android bijvoorbeeld maakt de adresbalk geen deel uit van de viewport. Hierdoor was er dus wel een scroll van enkele centimeters aanwezig als de applicatie bezocht werd in de browser. Dit probleem wordt opgelost als de applicatie geïnstalleerd wordt, dan is er geen adresbalk aanwezig.

5.5.3 Verwachting gebruiker

Een gebruiker die een applicatie installeert op een IOS toestel zal een andere user interface verwachten dan een gebruiker die een applicatie installeert op een Android toestel.

Met een PWA kan er slechts 1 user interface geïmplementeerd worden, het is dus niet mogelijk om volledig aan de verwachting van elke gebruiker te voldoen.

Een gebruiker met een toestel dat op IOS werkt is gewoon om naar de vorige pagina te navigeren door naar rechts te swipen op het toestel. Dit is een manier van navigeren die een Android gebruiker niet gewoon is. In deze proof of concept kan er niet met touch-gebaren genavigeerd worden. Dit kan niet 'native' aanvoelen voor de IOS gebruiker.

Uit de resultaten van de user testen bleek dat veel mensen niet weten welke functionaliteit beschikbaar is voor het web. Zo wist X% niet dat een webapplicatie gebruik kan maken van de camera en X% wist niet dat een website ook offline kan werken.

Dit is relevant omdat als een gebruiker niet weet dat een PWA dit kan doen, dit waarschijnlijk ook niet zal testen.

5.5.4 Gebruik van PWA's

In de user testen werd er gevraagd als een gebruiker een PWA zou verkiezen boven een native applicatie. De meeste mensen zouden een native applicatie nog steeds verkiezen boven een PWA. De reden hiervoor is dat veel mensen enkel op zoek zijn naar een app enkel in de app stores zoeken.

In het onderzoek werd vervolgens dezelfde vraag opnieuw gesteld, maar nu werd er meer info gegeven aan de testpersoon over wat PWA's zijn. De tester werd verteld dat een PWA ook het startscherm kan staan, notificaties kan versturen, en dat de app-grootte veel kleiner is dan die van een native app. Er was slechts X% van de gebruikers die een PWA zou verkiezen boven de native versie.

De testers hadden uiteenlopende redenen waarom ze een native applicatie verkozen:

- 'Heb geen reden om geen native gebruiken'
- 'Native heeft een beter gevoel'
- 'Ik vertrouw een webapplicatie minder dan een native applicatie'
- 'Heb geen reden om geen native gebruiken'
- 'Ik prefereer snelheid en integratie van native'

Als de tester beter weet wat de voordelen zijn van een PWA, waren er opvallend meer mensen die de PWA versie zouden verkiezen over de native versie. Hun grootste motivatie was de grootte van de applicatie. Er waren X% van de gebruikter die wel een PWA zouden verkiezen als ze weten wat de voordelen zijn voor hen.

Pus notificaties zijn een sterk middel om user engagement te verhogen. (**Gaunt2020**) Als

een PWA hier gebruik wil van maken moet de gebruiker hier expliciet toestemming voor geven.

In de tweede proof-of-concept werd deze functie geïmplementeerd. De gebruiker kan deze zelf aanzetten, hij zal dus niet lastig gevallen worden met popup's die om toestemming vragen. Voor dat de gebruiker de notificaties aanzet, weet hij voor welke doelen dit gebruikt zal worden .Dit is volgens Matt Gaunt de beste user experience en zal dan ook resulteren in een hoog aantal mensen die notificaties toestaan. (**Gaunt2019c**)

Er werd aan de gebruikers gevraagd als ze voor deze applicatie de notificaties zouden aan zetten. Een opvallen hoog percentage (X%) zou toestemming geven aan een PWA om push notificaties te versturen.

5.5.5 A2HS

Bij het user testen werden de twee proof-of-concepts getoond aan de gebruiker. Bij de eerste proof-of-concept wordt er een standaard A2HS ervaring aangeboden. Op sommige browsers (Chrome, Edge, Firefox) zal de gebruiker een boodschap krijgen als hij de website voor het eerst bezoekt dat deze geïnstalleerd kan worden. Op andere browsers (Safari) zal dit helemaal niet aangeven worden.

Bij de tweede proof-of-concept stond er een knop op de website waar de gebruiker altijd op kon klikken om de applicatie toe te voegen aan het startscherm.

Het was opvallend dat de gebruikers de tweede applicatie sneller zouden installeren dan de eerste. De testers gaven volgende feedback

- -'Voelt niet veilig'
- 'ik weet nog niet wat deze app zal doen'
- 'ik klik alle popups direct weg zonder te lezen wat er staat'

Onderzoek (**LePage2020b**) toont aan dat de gebruiker meer en langere sessies heeft als de PWA te vinden is op het startscherm. Het is dus belangrijk om een goede A2HS experience te ontwikkelen.

De A2HS ervaring op een IOS toestel is minder eenvoudig dan die op andere toestellen. De gebruiker moet manueel in de instelling van de website gaan om de app toe te voegen aan het startscherm. Dit is een omslachtig proces dat niet veel gebruikers kennen.

Dit was ook terug te zien in de user testen. Tester op IOS voegen bijna nooit een PWA toe aan hun startscherm, terwijl mensen op Android dit wel zouden overwegen.

5.5.6 Native feeling

Er werd onderzoek gedaan naar de user experience van een PWA die werd toegevoegd aan het startscherm. Deze applicaties hebben geen browserelementen en zien er dus meer

uit als een native applicatie.

De testers kregen de vraag "Vond je de ervaring in de demo's gelijkaardig als die in een mobiele applicatie?"Hier antwoordde X% dat ze de ervaring van de PWA's gelijkaardig vonden aan die van een native applicatie.

Ook hier valt op dat de tester die vonden dat de ervaring niet gelijkaardig was aan die van een native applicatie allemaal gebruikt maakten van een toestel met IOS als besturingssysteem.

6. Conclusie

6.1 Resultaten

6.1.1 Welke applicaties kunnen er ontwikkeld worden als PWA

In sectie 2.6 werd beschreven in welke situaties het gebruik van PWA's het meest voordelig is en in sectie 2.2 werd er een overzicht gemaakt van welke functionaliteit met een PWA geïmplementeerd kan worden en welke niet.

Door deze twee onderzoeken samen te leggen kan er geconcludeerd worden welke use-cases optimaal zijn voor PWA's.

6.1.2 Functionaliteit van PWA's en besturingssystemen

In hoofdstuk 2 werd er aangetoond dat PWA's kunnen genieten van veel verschillende web-API's om native functionaliteiten te implementeren. Applicaties die afhankelijk zijn van de camera of van locatievoorzieningen kunnen perfect geïmplementeerd worden als PWA. Ook meer geavanceerde functionaliteiten zoals in app betalingen en virtual reality kunnen in theorie geïmplementeerd worden.

De ondersteuning van deze krachtige web-API's is helemaal niet consistent, wat het moeilijk maakt om een PWA te ontwikkelen die afhankelijk is van verschillende web-API's.

In het algemeen kan er gesteld worden dat Google en Microsoft een goede ondersteuning bieden voor PWA's. Ze ondersteunen niet enkel PWA's maar beide bedrijven werken actief aan het uitbreiden van de functionaliteiten die beschikbaar zijn voor het web. Apple

daarentegen is conservatiever op vlak van het ondersteunen van PWA's. Het bedrijf is trager in het ondersteunen van nieuwe web-API's. Het duurde lang voor IOS service workers ondersteunde, dit gebeurde pas in 2020. Ook worden bepaalde belangrijke web-API's niet ondersteund, het grootste voorbeeld hiervan is de push API.

6.1.3 Beperkingen van een PWA

content

6.1.4 PWA alternatieven

Er zijn verschillende technologieën en frameworks waarbij er met één codebase een applicatie ontwikkeld kan worden voor verschillende platformen.

Er zijn verschillende parameters die bepalen welke benadering te juiste is voor een bepaald project.

Een PWA is een goede oplossing voor applicaties waarbij de go to market time zo laag mogelijk moet zijn en er snel en eenvoudig updates moeten uitgevoerd worden. Een groot voordeel van PWA's is ook dat het gevonden kan worden in zoekmachines.

Een hybride oplossing is interessant als er zowel een webapplicatie als een native applicatie nodig is. Hier worden zowel prestaties en user experience opgeofferd, maar de applicatie is wel op alle platformen beschikbaar.

Als er een native applicatie verwacht wordt zijn er verschillende benaderingen om dit te implementeren. Hier zal de achtergrond en voorkennis van de ontwikkelaar een grote impact hebben op wat het juiste framework is.

6.1.5 Welke stappen zijn er nodig om een traditionele website om te vormen tot een PWA

Bij deze onderzoeksvraag werd een PWA gezien als een webapplicatie die kan geïnstalleerd worden op een toestel en die offline gebruikt kan worden.

Dit is kan eenvoudig bereikt worden met aan de hand van moderne frameworks zoals React. De service worker wordt automatisch gegenereerd. Ook het app-manifest kan gegenereerd worden. Er zijn verschillende platformen die https-hosting aanbieden.

De drie criteria om een PWA te installeren kunnen dus eenvoudig bereikt worden.

6.2 Niet-kwantificeerbare resultaten

In de proof-of-concept werd aangetoond dat geavanceerde concepten zoals peer-to-peer verbindingen en real-time communicatie mogelijk zijn voor PWA's. Een PWA kan dus veel functionaliteiten implementeren die voordien enkel beschikbaar waren voor native applicaties.

Uit de proof-of-concept bleek ook dat een PWA geen 'native-gevoeld' kan bieden. Dit heeft verschillende oorzaken, zo heeft de gebruiker op een bepaald platform een verwachtingspatroon van hoe een applicatie zich zal gedragen op dat platform. Dit is voor elk platform anders. Bij PWA's wordt er 1 user interface gemaakt voor alle platformen.

Een andere oorzaak is dat het niet eenvoudig is om een consistente user interface te ontwikkelen die native aanvoelt voor alle mogelijke schermgroottes.

6.3 Vergelijking met verwacht resultaten

Er werd verwacht dat het ontwikkelen van een PWA met service worker functionaliteit een complexe opdracht zou zijn. Deze verwachting klopte niet, de meest voorkomend functionaliteiten is er een duidelijk documentatie. Ook maakt workbox het heel eenvoudig om caching toe te passen in een applicatie.

6.4 Verder onderzoek

Veel technologieën die gebruikt kunnen worden door PWA's zijn relatief nieuw en zijn nog niet grondig onderzocht. In deze thesis werd de nadruk gelegd op welke functies er wel en niet kunnen gebruikt worden door PWA's. Om hier een goed beeld van de schetsen kan er nog onderzoek gedaan worden naar de performantie van deze web-API's.

In de proof of concept in deze scriptie werd webRTC gebruikt om een peer-to-peer connectie op te zetten. Dit is een technologie met heel veel mogelijkheden, er kan onderzoek gedaan worden naar wat er bereikt kant worden met webRTC en hoe performant dit is.

A. Interviews

A.1 Intro

When did you first hear of PWA's and when did you first came in touch with them. What were your initial thoughts.

A.1.1 Thomas Steiner

must have heard about PWAs the first time at some point in 2015. I was working on a team that focused on the mobile web and how Googles publishing and advertising customers could profit from it. Google makes most of its money from ads, so it seemed (and still seems) natural that it makes more sense to build a great website that you buy ads for than to build a great Android or iOS (or back in the days Windows phone) app that you need to buy ads for only for people to install it.

A.1.2 Wassim Chegham

I am a engineer specialized in Javascript. I have mainly been involved in Angular and back in the day Angular.js. I'm also part of the google developer expert program. You can get this title by being part of the open source community and do a lot of talks. You get to try out the cutting edge technologies.

I heard about PWA's when Alex Russel wrote his infamous blogpost. In the beginning it was really experimental and it didn't have many features. I was playing with in my own side projects, I wrote a lot of really bad manifests and service workers just to figure

out how it works under the hood. But today progressive web apps are mainstream and production ready. So, so that's how I get to get to try out this new way of building applications. And that's how I discovered the technology.

A.2 Selling points

How and why would you convince your team to make you next application instead of a regular site of native app.

what are for you the main "selling"points of a PWA.

A.2.1 Thomas Steiner

- Build once, run everywhere (mobile, desktop).
- Less maintenance compared to native code bases.
- Easier discovery. A website just becomes an app progressively.
- Less memory consumption, better security story.
- No dependency on native app stores and their approval processes.

A.2.2 Wassim Chegham

So, um, well, first of all, I don't like convincing people that's a bad thing to do. Ah, but however, I mean, if I need to convince someone, I would probably, talk with the, with the business people because, having a progressive web app, I mean, yeah, I mean, it's cool. It's a new technology you get to play with around new stuff, but the real benefits of progressive web apps are. For business. mainly because, um, you get when you're right, a web app that is a progressive web app and like behaves and fees, like a native app. That's why what people are looking for.

For a company it's a huge benefit because it's cheaper to build, uh, cause like you don't have to build a, an Android app, an iOS app, a windows for app, ... You can just build like a web application with web technologies and then deploy it on a websites. Uh, so, and it's easy also to update because you just update website and everyone would have the new version.

Um, if you're building apps for mobile, uh, with Android and iOS. You go, you have to go through the store. So you have to pay fees and you have to pay Apple and maybe Google, whatever. Uh, whereas like when you build a JavaScript web application, it's on your server. You updated whenever you want. You don't pay anyone.

It's also good for discoverability cause it's a web. Web page with SEO keywords and everything. So it's easy for people to find using Google search engine or Bing or Yahoo. So I would rather convince the business guys because it has more benefits for them than the engineering side of it.

A.3 Disadvantages

as the support is for PWA's is growing and the web is able to access more and more features of the OS (camera, gps, ...) there are still some limitations.

what types of applications benefit the most from the features that PWA's offer what applications wouldn't you implement as a PWA what are according to you the main disadvantages of PWA .

A.3.1 Thomas Steiner

The biggest issue is indeed varying browser support. Its hardest to compete on iOS/iPadOS, since Apple does not allow other browser engines on its platform. Were working on improving the support situation when it comes to device APIs and lower level features like file system access. Check out web.dev/fugu-status

A.3.2 Wassim Chegham

not asked

A.4 Is not being in the app-store an advantage or a disadvantage?

you don't depend on apple or google to distribute the app. But you also loose a lot of traction by not being in a store.

Some people will think an app in the app store is more credible then a webapp

A.4.1 Thomas Steiner

As you can see above, Google allows PWAs in the Play Store. Apple does not allow PWAs in the App Store, though. Yes, some developers want to be included in the stores with their apps, because people have learned to search there for apps. It does not have to be this way, though. Especially when it comes to new apps: the barrier to engaging with a website is way lower than the barrier to downloading an app and then engaging with it.

A.4.2 Wassim Chegham

So, yeah, that's true. Because I mean, even for like indie developers who wants to sell their applications on play store or on the Apple store, so it's not really great experience

for them. Um, however, you can already start publishing your progressive web apps on the windows store and play store.

Apple is coming slowly to this, to this game, and they're, they're allowing people to, um, like a host or publish a progressive web apps. doing this, you can like review apps, give them like a stars, whatever you want. Um. And, uh, even probably, I, I haven't, I haven't witnessed that. But you could call some mix and match, like native with progressive stuff. So you can also, all right, you can publish it on an app store and have some of the part paid. You know, you can pay for services and, but uh, I personally, I don't see any benefits of doing that either. You go progrissve and get fully progressive web stuff. Or you can do it native because it's just going to complicate things on the engineering side if you combine them.

Previously in the past, if you want to install progressive web apps, you had to go to web a website and. Use the app for many, many times. Then you get the notification to install the application. Now you can download them from the first visit.

A.5 In what types of applications do you see PWA's excelling

A.5.1 Thomas Steiner

was not asked

A.5.2 Wassim Chegham

The first category is eCommerce and, AliExpress is the best example of this. I think last time I, uh, I heard the news about that they, like, they did like plus 100% in revenues or three of the So, yeah, that definitely a great,

Other use cases are consumer based. for instance there is, there is a Twitter progressive web app., there is a Starbucks to progressive web app. There is also Pinterest, these entertainment applications are also great examples. Because they they provide a service like you can consume a service. we can, yeah, we can imagine other, other categories around these like entertainment and music. I guess Spotify also has a progressive web app.

Maybe even things like communication applications like video calls, like Slack,... I don't know if they have a progressive web app at the moment, but they will at some point. I'm sure they will do it in the future. I mean, when you take Slack or teams, for instance, it's already an electron app. There is no reason for it not to be a progressive web app. They're already doing all the heavy lifting and heavy engineering. Um,

However, there are applications that's are not well suited for progressive web app are things like games for instance. Um, games are really, uh, like, uh, like heavy. I mean, they are, heavy in terms of resources, you know. it's better for them to be written in native C

plus plus or these kinds of technologies instead of web. Because one of the characteristics of a progressive above is it needs to be smooth, you know, a smooth UI. Kind of animation and pleasant to use. And for games. I mean, of course, if you're building like, um, casual games, you know, that's, that's fine. But yeah, I mean, you cannot build Fortnite in as a progressive web app.

A.6 Adoption

Some features that provide a better experience are fairly easy to implement. Why are most of the big websites not making use of these features? Why does facebook not have some basic caching which caches the application shell and provides some offline functionality?

react.js makes it really easy to implement to convert a regular application to a offline capable PWA. But writing your own custom service worker is really cumbersome. Can you explain why this is not easier with a big framework as react?

A.6.1 Thomas Steiner

Facebook is a complex site that runs many multivariate tests on its users (not everyone sees the same Facebook) and that needs to support tons of platforms and browsers. They do have some efforts in the direction of PWA,for years actually]

A.6.2 Wassim Chegham

So my opinion, i Don't know this but it might be politic reasons or internal reasons. They might not have any benefit of doing this because as you said, like on the engineering side, it's easy to do. I mean, anyone can do it. If they don't want to do it is theirs. For sure there is a politicel reason or financial reason for this. Do it's definitely not technical, but it's, I don't know, it's maybe business or political stuff, which is obviously not public and they won't say it.

It's probably not their priority right now. And Facebook is growing it started as like a social network, but now it's growing. Like it's, maybe it's a market. You can sell stuff on it. You can have chatbots, you can have lots, a lot of stuff. So maybe that's not compatible with progressive about what you're doing. I'm just guessing here. I don't really know the main reason.

Working in a big corporation, like I do right now. Most of these reasons are usually political and not technical. When it comes to politic, most of us, we don't know the reason.

A.7 Future of PWA's

One of the biggest reasons why windows on mobile didn't work was convincing developers to create applications for windows.

When Huawei couldn't use google services anymore this was probably also the biggest problem for huawei.

Both of these parties have resources enough to develop their own OS. But convincing developers to develop for their OS is not that easy.

At the moment there is no doubt that Google is pushing the PWA movement the most. And it's really 'googly' thing to do, but I don't really understand why they are doing this.

Isn't PWA's a thread to google, because this could mean to losing their oligopoly with Apple.

How do you see this this evolving for other parties and do you think that google will keep on innovating on PWA's?

A.7.1 Thomas Steiner

Google runs Android, too, which has the Play Store. Many developers want to be present and discoverable in the Play Store, but without building an Android app. This is why we allow PWAs in the Play Store now through a technology called Trusted Web Activity.

We see a huge desire from developers to write an app once, and then distribute it everywhere. In general, more and more efforts are being made in the direction of multi-platform frameworks like Flutter, React Native (Web), or PWA.

A.7.2 Wassim Chegham

So one thing I learned, during this all this time, talking with, uh, Google engineers or working with them on some open source projects or all the things I've heard, like I'm not, I'm not working for Google obviously, but the things I've heard like, And it's the same thing for Microsoft. So, uh, one thing to learn to know is that like Google and Microsoft and other big corporations, have many engineering teams with different specialties, for instance, there's an Android team, the Chrome team, the angular team, the GCP, ... And every organization has their own priorities, their own budgets, their own ... , it's like a small company inside the company .

Google is not making money of Android. Google is an ad search company. So they are making big money on that, So having an operating system like Android, just help them push more ads to people. That's why Android is an open source project and parts of the OSP organization and everything, uh, they're also making money from GCP, cloud. So that's why it's not open source.

Having the Chrome team, it's not Google, but the Chrome CR pushing progressive web apps. And of course from the outside, you don't see the Chrome team, the underlying team. You see just Google the Google brand. It, same thing for Microsoft. If you have, like, if you have like the TypeScript team pushing TypeScript. Whereas like Microsoft, they have like C# and all the other technologies. Uh, it's like small teams are pushing hard, their work. and from the outside just you just see the peer corporation. So coming back to your question, I don't think it's there is any conflict here between progressive web apps on Android. They have the biggest market share, you know, it's open source. Anyone can take it and build their own, uh, as you said. So having the Chrome team pushing progressive web apps and especially web in general. So that's the main purpose behind this progressive web app set of technologies. By pushing progressive web apps, like indirectly, they're also pushing products like Chrome, like other things like specifications they're working on,,for instance, the web USB they are working on with many Googlers and Google engineers. Uh, of course there are other, like there, there is like a Samsung internet there, Microsoft, some of them are also pushed by Apple. So it's also a way for them to push other that services and I don't personally see any conflict in here between Android as an operating system and progressive web apps.

A.8 Installation of PWA's

We, as 'tech-savvy' people know that if an installation prompt is fired, that it is a pwa and nothing harmful.

The problem though is that most non 'tech-savvy' people just continue using the webapp without installing it.

probably also not using the features that a pwa can provide

- example: they won't use a PWA offline, because they don't know that it's capable of doing that.

Do you have a solution for this and do you expect a shift in user behavior?

A.8.1 Thomas Steiner

Its something that requires learning. People used apps on the web, until Apple invented the App Store and established the theres an app for that mantra. This pattern can be unlearned over time, too.

A.8.2 Wassim Chegham

If the users sees that message on the bottom And they can simply click on the close and they won't see it ever. Um, so that shouldn't be a problem. Most of them, maybe they would just see the message and ignore it I guess. Or if they are coming to a website to

read something or do something on the website it's like when you visit a site and you have ads, swerving at your face, and you just click and say, okay, leave me alone. I think they would do the same thing with the install banner, uh, at the bottom. Uh, but yeah, for the future. I don't have any information about that if, if this is going to evolve, but. One thing. I'm definitely sure about that, if there is any problem related to this feature, I'm sure. specification would evolve too, to deal with this new kind of, of issue. That's one of the benefits of working on the web. I don't see any issue with that because many people would just ignore the message message

For a company like AliExpress or like a big company. If they want to add this kind of feature, they probably do a lot of specific research that this feature will be used by the consumers.

And in the end, it's really easy to make a website offline capable, so it's not a big investment to do that.

A.9 Apple

Apple is as often not that keen on giving access to their OS. This is the case for PWA's. Do you expect better support for IOS and macOS and safari?

A.9.1 Thomas Steiner

The support situation will slowly improve, but we have to respect that they have different views on some things, and thats fine.

A.9.2 Wassim Chegham

I don't know, i just use Apple products, but i don't know anything about Apple.

However, I was going through Twitter recently. I saw that they hired a lot of people I know. Most of them are joining the open source organization inside Apple. So maybe there is a message hidden. They're there. They're probably starting to build a open source community around the products they're building. But, but that's, that's the only thing I can guess from what's happening. UYou should probably ask someone who is really familiar with Apple, uh, like from the beginning.

Look how the web specs works, it's like aevery big company, every company has a chair into the web spec. And they all can have like their veto. So if Google for instance tells, okay, I have this new API I want to push, I want to be, make it public and everything. so Microsoft can say, okay, I will build it in edge. But if Apple says, I won't do it in, in Safari. You can do anything you want, they won't do it. And that's it. That's end of story. If you can convince them, you can. if you can't you cant

But I've been following some of the meetings of the W3C. There is a lot of discussion going on and some of them are really public, you can probably check them out.

But to conclude the question, it's all about politics

A.10 Beginning of pwa's

you have a nice amount of experience, what are the biggest shifts you have experienced on the web.

Do you think that pwa's will stay a 'niche'? Or do you think it will become the mainstream for webapplications to register a service worker and add service worker functionality?

A.10.1 Thomas Steiner

Google Gears has piloted a lot of the HTML5 APIs that we take for granted today. Getting those APIs was probably one of the biggest steps in the past.

Its niche, but its going up. A lot of the tooling is still in its infancy, but libraries like Workbox.js make it easier and easier to develop efficient PWAs.

A.10.2 Wassim Chegham

So, I'm witnessing a lot of all the work that, um, not even Google, but also other engineers, even Microsoft, some of people at Apple and all the major companies and also other companies like Samsung, like Intel, they're all doing a lot of work around the web and the underlying layers of the web.

for instance, eh, so if you want to define what's the progressive web app, it's an app that behaves and then feels like a native app. This. When you think about a native application, i has access to camera, has access to a microphone, it has access to all the sensors, to your, your contacts or SMS, everything. I mean, it's a native app. You have access to the whole OS.

Intel is, has been working on an WEB NFC. There's also web Bluetooth, there's a lot for the. I think there is a new standard. It's called, um. Web of things. it's like an umbrella spec around all this hardware access.

A lot of companies are doing really great job for this, and I can see that progressive web app to benefit from that. Obviously they're doing this to expose API to javascript and on the web in general.

So I think even like in , maybe three, four coming years. We'll have even more features for progressive web apps. I mean, today we have lots of stuff.

So my prediction is progressive web apps are, are, are here to stay. I'm talking about the technology and not the politics behind it. These will probably change.

I even think that in the future you wil be able to monetize you progressive web app. Like we do today with native apps. It's becoming a big part of the web.

PWA's is not one API or one technology, it will probably evolve. And that's why I believe PWA's are here to stay

Anytime a feature is added to the web, it's there forever. It's impossible to remove, because then a lot of websites would brake. So now they can't stop supporting service workers and service worker functionalities.

A.11 Extra

A.11.1 Thomas Steiner

Do you believe that one day, all applications (mobile/ desktop) will be written with web technologies?

No. But a lot of the apps will. Just look at how many Electron/Cordova/ apps out there are almost web apps, but rely on the wrapper for just a small amount of their functionality.

favorite PWA

Twitter.com. Its hands-down one of the best experiences so far.

favorite WEB API

Fetch(). Its made XMLHttpRequest a lot easier.

How do you see the monetization options for PWA's

For selling stuff in the PWA: there are no limits, since you control it and theres no store tax. For selling the actual PWA: You can do this today by requiring a paid log-in, or by charging for the app when you put it in the Play Store (albeit I dont know of anyone doing this).

A.11.2 Wassim Chegham

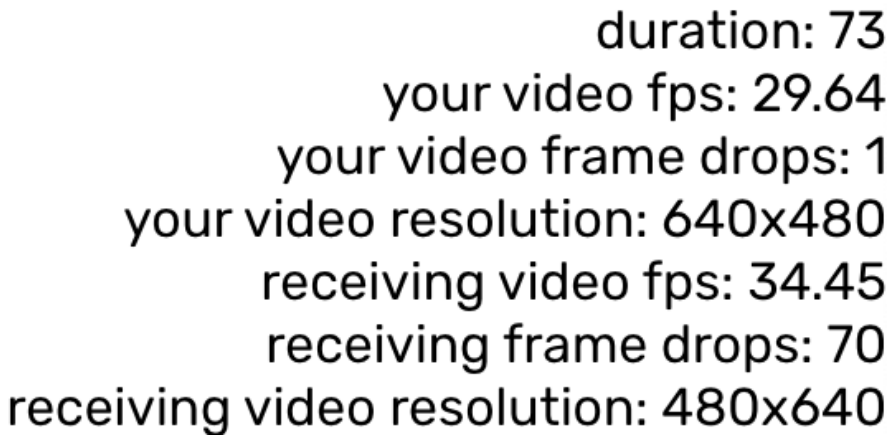
was not asked

B. Performantietesten video stream

B.1 Code voor het testen

```
1  const measureFrames = (vid, vidname) => {
2    let timer = 0;
3    setInterval(() => {
4      timer++;
5      let fps;
6      if (vid) {
7        fps = vid.webkitDecodedFrameCount / timer;
8        const _stats = { ...stats };
9        if (vidname === "yourVid") {
10         _stats.yourVid.fps = fps;
11         _stats.yourVid.dropped = vid.webkitDroppedFrameCount;
12         _stats.yourVid.resolution = `${vid.videoWidth}x${vid.
               videoHeight}`;
13       } else {
14         _stats.receivingVid.fps = fps;
15         _stats.receivingVid.dropped = vid.webkitDroppedFrameCount;
16         _stats.receivingVid.resolution = `${vid.videoWidth}x${vid.
               videoHeight}`;
17       }
18       _stats.timePassed = timer;
19       setStats(_stats);
20     }
21   }, 1000);
22 };
```

B.2 Screenshots van de resultaten



Figuur B.1: statistieken proof-of-concept

Item Name	Send	Receive
Latency	95 ms	92 ms
Jitter	2 ms	0 ms
Packet Loss - Avg(Max)	1.5% (4.3%)	0.0% (0.0%)
Resolution	640×360	480×640
Frame Per Second	26 fps	14 fps

Figuur B.2: statistieken zoom

C. 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.

C.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. (**Hiltunen2018**)

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

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

In deze thesis zullen volgende onderzoeksvragen uitgewerkt worden:

- Welke stappen zijn nodig om een traditionele website om te vormen tot een PWA?

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

C.2 State-of-the-art

C.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.

(**Harris2017**)

C.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

(**Malavolta2016**) (**Destrebecq2018**)

C.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. (**Google2017**)

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. (**Google2019**)

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

- 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.

(**GooglePwa2019**)

C.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. (**Durumeric2013**) Dit is een stap in de juiste richting maar het zorgt er dus niet voor dat elke PWA een veilige toepassing is. ' worden vandaag gebruikt om gebruikers op te lichten. Een vaak gebruikte techniek is dat een bestaande native application wordt nagemaakt. Op deze manier proberen ze wachtwoorden en betalingsdetails te verkrijgen. (**Lee2018**)

C.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. (**Richard2019**) (**Malavolta2016**)

C.2.6 Apllication shell architecture

Om een snelle ervaring te bieden aan de eindgebruiker moet een applicatie opgedeeld worden in twee delen: de inhoud en de application shell. De application shell is het deel van de interface die op elke pagina terugkomt. Dit kan bestaan uit achtergronden, navigatie, Deze application shell moet volledig lokaal opgeslagen worden zodat deze niet steeds opnieuw gedownload moet worden. (**Hiltunen2018**)

C.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.

C.3.1 Technologieën

PWA's kunnen gemaakt worden met de tools en technologiën die gebruikt worden om traditionele webapplicaties te bouwen. Voor de onderzoeken zal voor de eerste fase HTML,

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

C.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. (**Gaunt2019**) (**Osmani2016**)

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 '. Het is daarentegen bekend dat Apple toegang tot het besturingssysteem zoveel mogelijk wil beperken. Een duidelijk verschil tussen deze twee besturingssystemen wordt verwacht. (**Hansen2017**)

C.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 ' 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.