HO
GENT

Faculteit Bedrijf en Organisatie

Progressive web application vs Native application

Robbie Verdurme

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

Promotor:
Lieven Smits
Co-promotor:
Bart Delrue

Instelling: Digipolis

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Progressive web application vs Native application

Robbie Verdurme

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

Promotor:
Lieven Smits
Co-promotor:
Bart Delrue

Instelling: Digipolis

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

In deze bachelorproef wordt er een vergelijking gemaakt tussen een native applicatie en een progressive web applicatie. De reden hiervoor is mijn stageplaats bij Digipolis. Digipolis wil graag onderzoeken wat de mogelijkheden van een progressive web applicatie zijn, zodat deze technologie eventueel gebruikt kan worden binnen het bedrijf. De reden waarom digipolis dit wil onderzoeken is om te weten te komen of applicaties een meerwaarde kunnen bieden voor de doelstellingen van digipolis (burgers informeren en helpen participeren) en welke methode best past binnen hun ecosysteem.

Graag wil ik mijn promotor Lieven Smits bedanken voor de feedback en de steun tijdens het schrijven van de bachelorproef. Daarnaast wil ik ook Bart Delrue bedanken voor de technische feedback en de begeleiding tijdens het verloop van de bachelorproef.

Samenvatting

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Codevoorbeeld

1. Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (Pollefliet, 2011):

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

1.1 Probleemstelling

Digipolis ontwikkelt verschillende webomgevingen voor de stad Gent. Voorbeelden hiervan zijn de website van de stad Gent (https://stad.gent) en visit Gent (https://visit.gent.be). Graag wil digipolis weten of een applicatie van stad Gent handig zou zijn voor de bezoeker, zodat deze de site niet meer moet opzoeken. Daarnaast heeft een applicatie heeft het voordeel van offline beschikbaar te zijn. Hierdoor kan de bezoeker zonder enige data te verbruiken de applicatie toch raadplegen.

Voor dit probleem kan een progressive web applicatie de oplossing bieden. Een progressive web applicatie is een applicatiesoftware die via het web wordt geleverd en is gebouwd

met veelgebruikte web technologieën zoals HTML, CSS en JavaScript. (Wikipedia, 2020c). Door deze technologie is er maar één codebase nodig voor de verschillende platformen(Android, IOS, Web) en moeten de aanpassingen en het onderhoud maar op één plaats toegepast worden.

De native applicatie is een andere oplossing voor dit probleem. Een native applicatie wordt specifiek ontwikkeld voor een platform (Android, iOS, Windows Phone) in een eigen codeertaal (Verwey, 2018). Deze applicaties zijn enkel verkrijgbaar in de app store of play store. De applicatie zal niet beschikbaar zijn via de browser.

Vervolgens wil men weten welke van deze twee oplossingen beter is op vlak van performance en benodigde ruimte. Bijkomend is het belangrijk om de applicatie snel terug te vinden in een vertrouwde omgeving en dat de applicatie er vertrouwd uitziet om de gebruikerservaring te garanderen.

1.2 Onderzoeksvraag

Dit onderzoek zal nagaan hoever de progressive web applicatie de native applicatie kan benaderen. Hiervoor zijn volgende onderzoeksvragen opgesteld:

- Wat zijn de voordelen van PWA versus Native Apps?
- Welke frameworks komen hiervoor in aanmerking?
- Wat is de impact van een PWA op de gebruikerservaring en toegankelijkheid?
- Kan de PWA de native app vervangen?

1.3 Onderzoeksdoelstelling

De doelstelling van deze bachelorproef is een vergelijking maken tussen de native applicatie en een progressive web applicatie. Deze zullen getest worden op verschillende vlakken namelijk de performance, de benodigde ruimte en de gebruikerservaring. Uiteindelijk zal er een besluit gevormd worden waarin duidelijk zal gemaakt worden voor welke applicaties een progressive web applicatie de beste keuze is.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3, 4 en 5 worden de belangrijkste aspecten toegelicht van de gekozen frameworks

In Hoofdstuk 6 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 7, 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. Stand van zaken

2.1 Progressive web applicatie

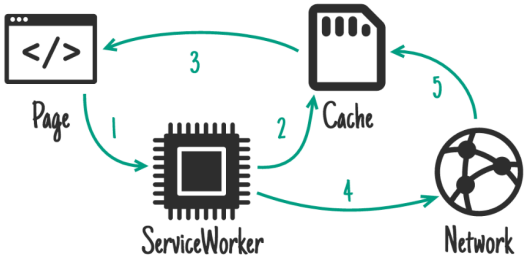
In deze sectie wordt er beschreven wat een progressive web applicatie allemaal extra nodig heeft naast een ontwikkelde website(Snipcart, 2019)(Saltis, 2020).

2.1.1 Service worker

Het eerste item dat een progressive web applicatie nodig heeft is een service worker. Deze zorgt ervoor dat de website alle aanvragen voor externe data kan opslaan, beter bekend als data caching. De aanvragen worden doorgestuurd naar de service worker die deze dan op zijn beurt haalt van de externe databron online of de lokale gecachte data. De service worker maakt deze beslissing op basis van de ingestelde strategie. Er zijn momenteel vijf strategieën die gebruikt kunnen worden met workbox (Google, 2020). Deze strategieën zijn stale white revalidate, offline first, online first, network only en cache only.

Stale-While-Revalidate

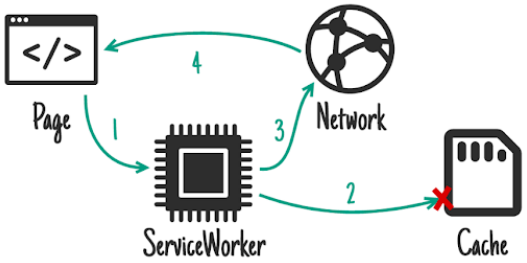
Deze strategie zorgt ervoor dat de gebruiker zo snel mogelijk een antwoord heeft op de aangevraagde data. Dit gebeurd doordat de service worker eerst de gecachte data zal weergeven als deze beschikbaar is. Indien er geen gecachte data is zal de service worker de aanvraag via het netwerk doen. Met het antwoord van het netwerk zal de cache dan weer geüpdatet worden met de laatste versie van de data.



Figuur 2.1: Voorstelling stale-while-revalidate strategie

Offline first

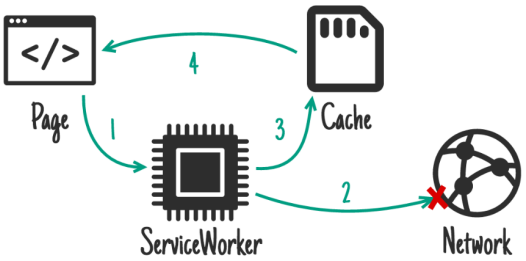
Bij deze strategie gaat de service worker kijken of de aangevraagde data al reeds gecached is op het apparaat. Als de data al reeds opgeslagen is op het apparaat, dan gebruikt de service worker de data die al reeds gecached is. Is er reeds geen data opgeslagen dan gaat de service worker via het netwerk de externe data opvragen. Eén van de grote nadelen van deze strategie is dat de externe data niet altijd up to date is.



Figuur 2.2: Voorstelling offline first strategie

Online first

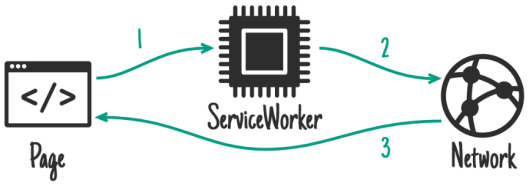
Bij online first zal de service worker eerst de aanvraag doen via het netwerk. Als er geen netwerkverbinding is zal de service worker de gecachte data weergeven indien deze aanwezig is. Om deze reden moet er altijd een netwerkverbinding zijn om alle data binnen te halen voordat de applicatie offline de data kan weergeven. Met deze strategie is de externe data altijd up to date indien er een netwerkverbinding is.



Figuur 2.3: Voorstelling online first strategie

Network Only

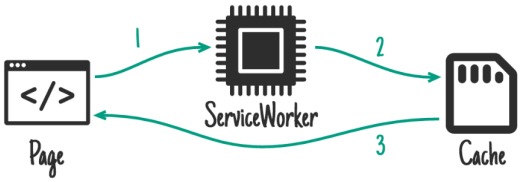
Network only is een strategie die de externe data enkel maar via het netwerk zal ophalen. Als deze niet beschikbaar is zal de gecachte data ook niet weergegeven worden.



Figuur 2.4: Voorstelling network only strategie

Cache Only

Cache only werkt alleen maar met de gecachte data. Deze haalt geen externe data gaan via het netwerk. Deze strategie kan gebruikt worden voor sites die geen externe data nodig hebben zoals een rekenmachine.



Figuur 2.5: Voorstelling cache only strategie

2.1.2 Manifest

Het tweede item dat een progressive web applicatie nodig heeft is een manifest (web docs, 2020). Dit is een bestand met alle informatie die nodig is om de applicatie draaiende te houden eens deze geïnstalleerd is op het apparaat. Deze informatie bevat maar is niet gelimiteerd tot de auteur, het icoon en de versie.

Auteur

De naam van de auteur van de applicatie. Deze kan ook de naam van het bedrijf zijn.

Icoon

Het icoon waarmee de applicatie wordt weergegeven bij de applicaties op het apparaat. Er kunnen verschillende iconen meegegeven worden. Deze kunnen dienen voor de verschillende besturingssystemen, zodat er bijvoorbeeld een uniek icoon is voor apple en android. De iconen kunnen ook een verschillende grootte hebben, dit zorgt ervoor dat het icoon goed zichtbaar is op elk scherm van de verscheidene apparaten.

Versie

De versie van applicatie kan ook ingesteld worden door de manifest file. Zo weet de gebruiker welke versie van de applicatie er geïnstalleerd is op zijn apparaat.

2.2 Frameworks

2.2.1 Wat is een framework ?

Een Framework is een geheel van softwarecomponenten dat gebruikt kan worden bij het programmeren van applicaties. Ook de afspraken over hoe die componenten gebruikt worden binnen een groep ontwikkelaars en welke code-standaarden en bibliotheken gebruikt worden kunnen onderdeel zijn van een framework. Het framework bepaalt welke software er binnen een organisatie wordt gebruikt en op welke manier (Wikipedia, 2019).

2.2.2 Requirements

Wegens we de performance willen vergelijken tussen de twee frameworks zullen we gebruik maken van dezelfde backend applicatie namelijk google firebase.

De requirements voor de web applicatie zijn de volgende:

- Service worker
- Manifest
- Router
- Store
- Data binding

De requirements voor de native applicatie zijn de volgende:

- Cross-platform
- Data binding

2.2.3 Gekozen frameworks

Progressive web applicatie

Er zijn verschillende frameworks voor een progressive web applicatie te ontwikkelen, maar op basis van bovenstaande requirement is er gekozen voor Vue.js in combinatie met Nuxt.js voor het creëren van de progressive web applicatie. Andere mogelijkheden zijn (Rajput, 2019) Angular, React, Ionic en Polymer.

Angular 2+

Deze heeft ook een package om de site naar een progressive web applicatie te transformeren. De reden waardoor angular niet geselecteerd werd is dat het relatief complex is en er al een kennis moet zijn van typescript voordat er een applicatie kan gebouwd worden.

React

React is geen officieel framework maar een library. Toch wordt deze opgenomen in de mogelijkheden omwille dat er een progressive web applicatie ontworpen kan worden met react. Deze werd niet geselecteerd omdat het relatief complex is in vergelijking met javascript. Daarnaast moet de programmeur een kennis van JSX hebben om react te kunnen gebruiken.

Ionic

Ionic werd niet geselecteerd omwille dat de applicatie frequent moet geüpdatet worden om mee te kunnen met de laatste veranderingen van het framework. Hierdoor kan de applicatie vrij snel verouderd worden. Als de site gebruik maakt van een verouderde package kan deze de site laten vastlopen.

Polymer

De reden waarom polymer niet geselecteerd werd is dat het niet search engine optimised is. Vervolgens heeft deze ook een hoge herlaadtijd voor de site.

Native applicatie

Om de native applicatie te ontwikkelen hebben is er gekozen voor het framework kotlin. Andere frameworks die in aanmerking kwamen waren Java en Xamarin.

Java

Java is een populair framework onder de developers. Deze wordt nog altijd gebruikt voor allerhande applicaties, van bureaublad applicaties tot android applicaties. Kotlin heeft vele aspecten van java maar heeft deze beter gemaakt. Om die reden wordt de performantie hoger bij het gebruik van het Kotlin framework dan het Java framework. (Puysseleyr, 2018)

Xamarin

Xamarin werd niet geselecteerd door dat de performantie op het android platform niet optimaal is. De performantie op ios is veel beter. Bijkomend zijn er maar een paar ondersteunende Integrated development environments (IDE's). Xamarin heeft ook geen

ondersteuning voor android xml files die nodig zijn voor de opmaak. Om deze reden worden veel fouten gemaakt bij de xml bestanden tijdens de programmatie van de applicatie (Babiuk, 2018).

2.2.4 Browserondersteuning

Een browser moet een PWA kunnen ondersteunen voor deze uitgevoerd kan worden. Omwille dat de PWA functionaliteit nog maar vier jaar oud is is deze nog niet overal geïmplementeerd. Hieronder wordt bekeken of de bekendste browsers de PWA functionaliteit al ondersteunen. De ondersteuning van de PWA functionaliteit in Safari kon niet gevalideerd worden wegens dat er geen apparaat beschikbaar was met Safari.

Chrome

Wegens dat de PWA functionaliteit ontwikkeld is door Google is chrome de eerste browser die deze ondersteund. Chrome kan de PWA zowel op mobiel als op computer gebruiken. Eens deze applicatie geïnstalleerd is op mobiel zal deze verschijnen tussen de andere applicaties, op de computer zal deze verschijnen bij het tabblad chrome://apps.

Firefox

Firefox ondersteunt standaard de PWA functionaliteit niet op de computer. Deze kan wel aangezet worden door een paar instellingen te wijzigen. Op mobiel ondersteunt Firefox de PWA functionaliteit wel met de Firefox Browser app, dit betekent dat de applicatie installeerbaar is. Na installatie is er een icoon te zien op het home scherm, maar niet in de applicaties zoals bij google chrome.

Opera

De PWA functionaliteit wordt helaas nog niet ondersteund op de computer. De service worker daarentegen wel. Deze zorgt ervoor dat de site nog beschikbaar is als het internet uitvalt. De mobiele versie van Opera (Opera Browser with free VPN) ondersteunt de PWA functionaliteit wel. Na installatie zal er net zoals bij de firefox browser een icoon te zien zijn op het home scherm maar zal deze niet voorkomen in de applicaties op het apparaat. Verder werd er ook gekeken naar de nieuwe mobiele applicatie Opera Touch. Deze ondersteunt PWA functionaliteit niet.

Safari

Safari ondersteunt sinds IOS 11.3 de PWA functionaliteit. Na de installatie is de applicatie terug te vinden bij de andere applicaties op het home scherm. Aangezien de applicatie te installeren is via de browser heeft de applicatie geen quality test van apple ondergaan. Apple heeft een paar limitaties opgelegd aan deze soort applicaties zodat het apparaat nog altijd optimaal zou beschermd zijn. Een paar van deze limitaties zijn dat de applicatie maxi-

maal 50 megabyte aan data mag verzamelen offline, er geen communicatie kan ontstaan tussen verschillende Apple-services zoals app betalingen en er geen push notifications ondersteund worden (Firtman, 2018).

2.2.5 Store

Deze sectie geeft een breder inzicht of de PWA site toegevoegd kan worden in de app store en play store.

Play Store

Google heeft reeds in 2019 beslist om de PWA functionaliteit toe te voegen aan de store (Firtman, 2019). Voor een PWA in de play store te kunnen toevoegen moet er een native applicatie met de nodige certificaten worden aangemaakt met in de applicatie een verwijzing naar de PWA site. Op deze manier zal de applicatie de laatste data tonen van de PWA site zonder elke versie te moeten opladen naar de play store. Enkel voor grote veranderingen binnen de applicatie zoals het icoon, het manifest bestand, en de TWA(Trusted Web Activity) moet er een nieuwe versie geupload worden naar de play store.

App Store

Voor je een PWA kan uploaden in de app store heb je een native wrapper nodig zoals Cordova. Deze zet de code om naar een Xcode project, dit draagt het nadeel met zich mee dat een PWA enkel op een MacOs apparaat kan geüpload worden in de store. (SimiCard, 2020)

3. Kotlin

Kotlin is een cross-platform programmeertaal, ontworpen om naadloos samen te werken met Java. Deze wordt officieel ondersteund door Google voor het ontwikkelen van mobiele apps op Android (Wikipedia, 2020b).

3.1 Basisprincipes

3.1.1 Null safety

Null saftey of null veiligheid is een feature die ingebouwd is in Kotlin (JetBrains, 2020). Deze feature zorgt ervoor dat variabelen niet de waarde null kunnen bevatten. Als een variabele null zou kunnen bevatten geeft Kotlin hierop een error. Door deze feature kan je met Kotlin bijna nooit een null pointer error hebben.

```
1 var a: String = "abc"
2 a = null // error
```

Codevoorbeeld 3.1: Null safty variablen

Dit is echter wel mogelijk als er een livedata variable of varianten hiervan worden aangemaakt. De livedata waarde List<String> wordt altijd geïnitieerd met null, indien dit niet gewenst is kan men deze altijd meegeven met een beginwaarde tussen de haakjes.

```
1 var c = MutableLiveData<List<String>>()
```

Codevoorbeeld 3.2: Null safty LiveData

Om de null veiligheid te overschrijven moet er een vraagteken geplaatst worden naast het datatype. Het vraagteken slaat erop dat de variabele nu de waarde null of een waarde van het datatype kan bevatten.

```
var b: String? = "abc" // kan met null ge nstalleerd worden
b = null // ok
```

Codevoorbeeld 3.3: Overschrijf null safty

Indien Kotlin een waarschuwing geeft bij een waarde die geen null kan zijn kunnen er twee uitroeptekens bij geplaatst worden Door dit te doen gaat kotlin ervanuit dat de waarde b nooit een null kan bevatten op die lijn code.

```
val l = b!!.length
```

Codevoorbeeld 3.4: Waarschuwing null safty overschrijven

3.1.2 LiveData

Livedata (Developers, 2020) is een observable die lifecycle-aware is. Dit betekent dat livedata de lifecycle van de verschillende app componenten zoals activiteiten, fragmenten en services respecteert.

```
var c = MutableLiveData<List<String>>()
```

Codevoorbeeld 3.5: LiveData

Aangezien livedata lifecycle-aware is moet ervoor het observeren van de livedata variable geen rekening gehouden worden met de verschillende staten dat de applicatie in komt. Hieronder is er een voorbeeld zichtbaar van een observatie van een livedata variable c.

```
c.observe(viewLifecycleOwner, Observer {
  a = it
})
```

Codevoorbeeld 3.6: Observe liveData

3.1.3 Databinding

Databinding (Developers, 2019) zorgt ervoor dat de data in de UI automatisch wordt geüpdatet indien er een nieuwe waarde is van de variabele userName. Om deze te kunnen gebruiken moet er gebruik gemaakt worden van de layout tag, met hierin de data tag met welke variabele dat er moet gebind worden. Op deze manier wordt de scope van de data tag afgebakend.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto">

<data>
  <variable
```

```
  name="viewmodel"
  type="com.myapp.data.ViewModel" />
</data>
<ConstraintLayout>
  <TextView android:text="@{viewmodel.userName}" />
</ConstraintLayout>
</layout>
```

Codevoorbeeld 3.7: Databinding

3.1.4 Coroutines

Coroutines (Kotlin, 2020) zorgen ervoor dat er acties asynchroon worden uitgevoerd. Dit is nodig voor de main thread waar de applicatie op draait vrij te houden voor IO inputs zoals touch commands. Zoals te zien is in onderstaand voorbeeld wordt er gebruik gemaakt van coroutines om een viewModelScope te lanceren. Alles binnen deze viewModelScope zal synchroon uitgevoerd worden op de applicatie.

```
var c: List<String> = Arraylist<String>()

viewModelScope.launch {
  val repositoryResponse = async {
    repsitory.doSomething()
  }
  val dataOrError = repositoryResponse.await()
  if(dataOrError.hasError()){
    requestError.value = genericErrorMessage
  }else{
    c.clear()
    c.addAll(dataOrError.data)
  }
}
```

Codevoorbeeld 3.8: Coroutine

3.1.5 Koin

Koin (Koin, g.d.) is een framework dat zorgt voor de dependency injection voor in een applicatie. Dependency injection is een ontwerppatroon om klassen te koppelen. Dit wil zeggen dat ze data kunnen uitwisselen zonder dat deze relatie vastgelegd is (Wikipedia, 2020a).

```
override fun onCreate() {
  super.onCreate()
  setupKoin()
  }

/**
* Setup DI with Koin.
*/
private fun setupKoin(){
```

```
10    startKoin {
11    androidLogger()
12    androidContext(this@App)
13    modules(listOf(repositoryModule))
14  }
15  }
16  /**
17  * Setup the repository module.
18  * Is public since we mock the repositories in tests.
19  */
20  private val repositoryModule = module {
21    single<IUserRepository> {
22      UserRepository(get(), get(), get(),get(),get())
23    }
24  }
```

Codevoorbeeld 3.9: Koin

4. Vue.js

Vue is een progressieve framework voor het bouwen van user interfaces. In tegenstelling tot andere monolithische kaders, wordt Vue vanaf de grond opgebouwd om stapsgewijs adoptable te zijn. De kernbibliotheek richt zich op de view layer alleen, en is gemakkelijk op te pikken en te integreren met andere bibliotheken of bestaande projecten. Aan de andere kant is Vue ook perfect in staat tot het voeden van geavanceerde Single-Page Applications bij gebruikt wordt in combinatie met moderne gereedschappen en ondersteunende bibliotheken (Vue.js, 2020d).

4.1 Basisprincipes

4.1.1 Declarative Rendering

Declarative Rendering (Vue.js, 2020c) zorgt ervoor dat de data in de html automatisch word geüpdatet indien er een nieuwe waarde is van de variabele message. Dit komt omdat de data en de DOM gelinkt zijn met elkaar, hierdoor is alles reactief.

```
<div id="app">
  {{ message }}
</div>
```

Codevoorbeeld 4.1: Declarative rendering html

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
```

```
})
```

Codevoorbeeld 4.2: Declarative rendering javascript

We kunnen de data ook op een andere manier linken met een html component. Deze gebruikt de v-bind: tag met de naam van het attribuut erachter. Dit zorgt ervoor dat de span title gebind is aan de waarde van message.

```
<div id="app">
  <span v-bind:title="message">
    Test bericht
  </span>
</div>
```

Codevoorbeeld 4.3: Declarative rendering html alternatief

4.1.2 Conditionals

In vue is er een mogelijkheid om verschillende html tags niet te renderen. Deze zijn gekend als conditionals (Vue.js, 2020b). Een conditional kan toegevoegd worden door een v-if attribuut toe te voegen aan de html tag. Deze zal valideren of de gegeven html tag moet gerenderd worden.

```
<div id="app-3">
  <span v-if="seen">Now you see me</span>
</div>
```

Codevoorbeeld 4.4: Conditionals html

4.1.3 Loops

Loops (Vue.js, 2020b) kunnen toegevoegd worden aan html tags zodat deze meerdere keren worden gerenderd. Deze kan gebruikt worden voor lijsten weer te geven zonder voor elk element in de lijst een html li tag aan te maken.

```
<div id="app-4">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```

Codevoorbeeld 4.5: Loops html

```
var app4 = new Vue({
  el: '#app-4',
  data: {
    todos: [
      { text: 'Learn JavaScript' },
```

```
      { text: 'Learn Vue' },
      { text: 'Build something awesome' }
    ]
  }
})
```

Codevoorbeeld 4.6: Loops javascript

4.1.4 Components

In Vue.js kan je gebruik maken van verschillende components (Vue.js, 2020a). Een component is een klein gedeelte van de site, dit kan bijvoorbeeld een footer zijn. Het is de bedoeling dat de programmeur de site opdeelt in kleine componenten. Door dit te doen kunnen de verschillende componenten hergebruikt worden op verschillende pagina's.

```
// Define a new component called button-counter
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">You clicked me {{ count
   }} times.</button>'
})
```

Codevoorbeeld 4.7: Components javascript

```
<div id="components-demo">
  <button-counter></button-counter>
  <button-counter></button-counter>
  <button-counter></button-counter>
</div>
```

Codevoorbeeld 4.8: Components html

5. Nuxt.js

Nuxt.js is een framework voor vue.js applicaties met als doel om vue developers eerste klasse technologieën te laten implementeren zoals server side rendering (SSR), code splitting en pre rendering (VueSchool, g.d.). Nuxt heeft ook een ingewerkte PWA library. Deze kan gebruikt worden om een site in een PWA te veranderen.

5.1 Basisprincipes

5.1.1 Server side rendering

Server side rendering (East, 2017) wordt gebruikt om je applicatie te renderen naar statische html en css bestanden. Na het doorsturen van de belangrijkste html en css bestanden worden deze weergegeven op het apparaat. Vervolgens zal de javascript bundel doorgestuurd worden. Dit betekend dat het apparaat geen gebruik kan maken van de javascript functionaliteit zolang deze bundel nog niet gedownload is. Pas na de parsing en uitvoering van de javascript bestand(en) op het apparaat kan hiervan gebruik gemaakt worden.

5.1.2 Pre rendering

Pre rendering (Simon, 2018) is gelijkaardig aan server side rendering. Het grootste verschil is dat pre rendering de pagina zal renderen en deze al opslaan als statisch bestand. Als deze dan wordt opgevraagd door het apparaat zal dit bestand niet nog eens gerenderd worden maar zal de een statische pagina teruggestuurd worden.

5.1.3 Configuration

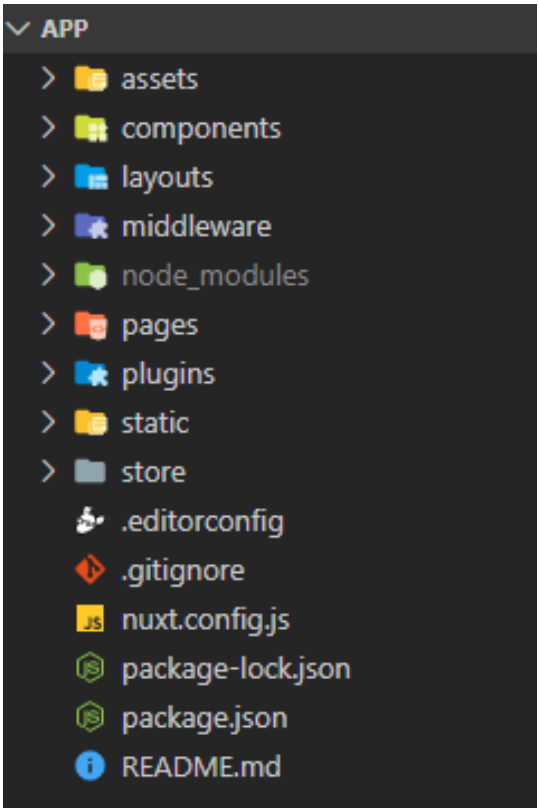
Het configuration of configuratie (Nuxt.js, 2019a) bestand (nuxt.config.js) is waar alle instellingen van Nuxt geconfigureerd worden.

```
export default {
  mode: 'universal',
/*
** Headers of the page
*/
  head: {
  title: process.env.npm_package_name || '',
  meta: [
    { charset: 'utf-8' },
    { name: 'viewport', content: 'width=device-width, initial-scale
   =1' },
    { hid: 'description', name: 'description', content: process.env
   .npm_package_description || '' }
  ],
  link: [
    { rel: 'icon', type: 'image/png', href: '/icon.png' }
  ]
  },
/*
** Customize the progress-bar color
*/
  loading: { color: '#fff' },
/*
** Global CSS
*/
  css: [],
/*
** Plugins to load before mounting the App
*/
  plugins: [],
/*
** Nuxt.js dev-modules
*/
  buildModules: [],
/*
** Nuxt.js modules
*/
  modules: [],
/*
** Build configuration
*/
  build: {
/*
** You can extend webpack config here
*/
    extend (config, ctx) {
    }
  }
}
```

Codevoorbeeld 5.1: Configuration

5.1.4 Directory Structure

Standaard heeft Nuxt.js al een mappen structuur (Nuxt.js, 2020a). Deze mappenstructuur is een goed beginpunt voor verschillende applicaties om een overzicht binnenin het project te bewaren. Indien de gebruiker deze structuur niet wilt gebruiken is deze vrij om de mappenstructuur aan te passen.



Figuur 5.1: Voorbeeld mappen structuur nuxt

5.1.5 Routing

Nuxt.js maakt het heel gemakkelijk om de verschillende routes (Nuxt.js, 2020c) in de site te configureren. Het enige dat de gebruiker hoeft te doen is een map genereren met de naam van de route en een index.vue bestand aanmaken waar nuxt standaard naartoe zal navigeren indien er geen parameters aan te pas komen.

```
pages/
--| user/
-----| index.vue
-----| one.vue
--| index.vue
```

Codevoorbeeld 5.2: Routering mappenstuctuur

Bovenstaande mappenstructuur zal deze routes genereren:

```
router: {
  routes: [
    {
      name: 'index',
      path: '/',
      component: 'pages/index.vue'
    },

    {
      name: 'user',
      path: '/user',
      component: 'pages/user/index.vue'
    },

    {
      name: 'user-one',
      path: '/user/one',
      component: 'pages/user/one.vue'
    }
  ]
}
```

Codevoorbeeld 5.3: Routering generatie

Indien er parameters aan te pas komen moet de gebruiker in plaats van een index.vue te creëren een bestand maken met als naam '_ naam van de parameter'. Hierdoor wordt naar deze pagina genavigeerd indien die parameter aanwezig is.

```
pages/
--| users/
-----| _id.vue
--| index.vue
```

Codevoorbeeld 5.4: Routering mappenstructuur met id parameter

Bovenstaande mappenstructuur zal onderstaande routes genereren:

```
router: {
  routes: [
    {
      name: 'index',
      path: '/',
      component: 'pages/index.vue'
    },
    {
      name: 'users-id',
      path: '/users/:id?',
      component: 'pages/users/_id.vue'
    }
  ]
}
```

Codevoorbeeld 5.5: Routering generatie met id parameter

5.1.6 Vuex Store

Vuex Store (Nuxt.js, 2020d) wordt standaard met Nuxt.js meegeleverd. Dit wordt gebruikt om verschillende gegevens te beheren. De Vuex store wordt opgebouwd uit verschillende modules, tijdens het bouwen van de website worden deze modules samengevoegd en geconfigureerd als Vuex store.

```
export const state = () => ({
  counter: 0
})

export const mutations = {
  increment (state) {
    state.counter++
  }
}
```

Codevoorbeeld 5.6: Vuex store index.js

```
export const state = () => ({
  list: []
})

export const mutations = {
  add (state, text) {
    state.list.push({
      text,
      done: false
    })
  },
  remove (state, { todo }) {
    state.list.splice(state.list.indexOf(todo), 1)
  },
  toggle (state, todo) {
    todo.done = !todo.done
  }
}
```

Codevoorbeeld 5.7: Vuex store todos.js

Bovenstaande bestanden worden geconfigureerd als volgend vuex store:

```
new Vuex.Store({
  state: () => ({
    counter: 0
  }),
  mutations: {
    increment (state) {
      state.counter++
    }
  },
  modules: {
    todos: {
      namespaced: true,
      state: () => ({
```

```
14          list: []
15        }),
16        mutations: {
17          add (state, { text }) {
18            state.list.push({
19              text,
20              done: false
21            })
22          },
23          remove (state, { todo }) {
24            state.list.splice(state.list.indexOf(todo), 1)
25          },
26          toggle (state, { todo }) {
27            todo.done = !todo.done
28          }
29        }
30      }
31    })
```

Codevoorbeeld 5.8: Vuex store

Een andere mogelijkheid om de store op te bouwen is om de modules over verschillende bestanden te spreiden. In de store map staan er dan vier bestanden: state.js, actions.js, mutations.js en getters.js.

5.1.7 Modules

Modules (Nuxt.js, 2020b) zijn Nuxt.js extensies waarmee de Nuxt functionaliteit uitgebreid kan worden met de volgende functionaliteiten: @nuxt/http, @nuxt/axios, @nuxt/pwa en @nuxt/auth. Deze modules kunnen verder uitgebreid worden met eigen geschreven modules.

@nuxt/http

Dit is een universele manier om HTTP-aanvragen te doen.

@nuxt/axios

De nuxt axios module gebruikt het axios pakket om HTTP-aanvragen te doen.

@nuxt/pwa

De pwa module zorgt ervoor dat je applicatie een progressive web app wordt. Dit zorgt ervoor dat de webapplicatie geïnstalleerd kan worden als een native applicatie op het apparaat.

@nuxt/auth

Deze module zorgt voor de authenticatie van de gebruikers.

5.1.8 Plugins

Plugins (Nuxt.js, 2019b) worden gebruikt om externe pakketten toe te voegen aan de applicatie. In de plugin wordt de configuratie van het gekozen pakket gedefinieerd. Na het configureren van de plugin moet deze toegevoegd worden aan de plugins in het configuratiebestand van Nuxt. In onderstaand voorbeeld wordt Vue material geconfigureerd.

```
import Vue from 'vue'
import VueMaterial from 'vue-material'
import 'vue-material/dist/vue-material.min.css'
import '~/assets/scss/material.scss' // css presets

Vue.use(VueMaterial)
```

Codevoorbeeld 5.9: Vue material plugin

Toe te voegen aan nuxt.config.js:

```
plugins: ['~/plugins/vue-notifications']
```

Codevoorbeeld 5.10: Plugin toevoegen aan nuxt.config.js

Indien de plugin een ES6 module exporteert moet deze ook toegevoegd worden aan de transpile optie binnen in het configuratiebestand.

```
build: {
  transpile: ['vue-notifications']
}
```

Codevoorbeeld 5.11: Plugin toevoegen aan nuxt.config.js transpile

6. Methodologie

7. Conclusie

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

De native app kende de afgelopen jaren een enorme groei in aantal gebruikers en werd de voornaamste manier om apps te maken. Er was oorspronkelijk wel een groot nadeel, namelijk dat de app apart moest worden geprogrammeerd voor android en voor ios. Hierdoor waren er verschillende code bases waardoor het onderhoud van de app moeilijker werd. Dit werd later opgelost door cross-platform development. Het nadeel hiervan is dat de categorieën zich vandaag niet meer beperken tot smartphone en tablets. Daarnaast blijft de app ook kampen met onderhoudsproblemen, vanwege bepaalde features die enkel beschikbaar zijn op ios. Aldus werd er verder gezocht naar een oplossing. Die is er nu, namelijk de progressive web app (PWA). Dit is een app die je gemakkelijk kan installeren op zowel android als op ios. Hierdoor los je het probleem op van de native apps. Namelijk dat een PWA maar 1 code base heeft voor alle platformen(Experius, 2019). Deze bachlorproef beseert zich op de volgende onderzoeksvragen.

- Wat zijn de voordelen van PWA vs Cross-Platform Native Apps?
- Welke frameworks komen hiervoor in aanmerking?
- Wat is de impact op de gebruikerservaring en toegankelijkheid ?
- Zal de PWA de native app vervangen?

A.2 Stand van zaken

Er zijn reeds onderzoeken uitgevoerd die de verschillende voor- en nadelen van een PWA en van een Cross-Platform app met elkaar vergelijken. Aldus onderzoekt men de mogelijkheid om native apps te vervangen door progressive web apps. Deze onderzoeken staan beschreven in artikels zoals die van Marjchrzak, 2018, Osmani, 2017 en Steiner, 2018

Uit onderzoek van Osmani, 2017 blijkt dat een PWA de performantie toch wel kan verhogen in vergelijking met een native app. Dit omdat de ruimte voor de app te installeren veel kleiner is dan een native app. Daarentegen vond ik terug in artikel Marjchrzak, 2018 dat er nog geen zekerheid is of een native app kan vervangen worden door een PWA. Daarnaast stelt Marjchzak dat PWA's nog niet supported zijn binnen het apple ecosysteem. Dit omdat safari nog geen ondersteuning biedt om een service worker te draaien op het systeem. Aangezien een service worker ervoor zorgt dat een PWA alle data cached, is deze essentieel voor het bouwen van een progressive web application. Als deze data terug moet opgehaald worden als er geen internet verbinding is, haalt hij deze van de service worker die het op zijn beurt ophaalt van de cash van het systeem. Hierdoor kan de PWA offline blijven werken eens alle data geladen is.

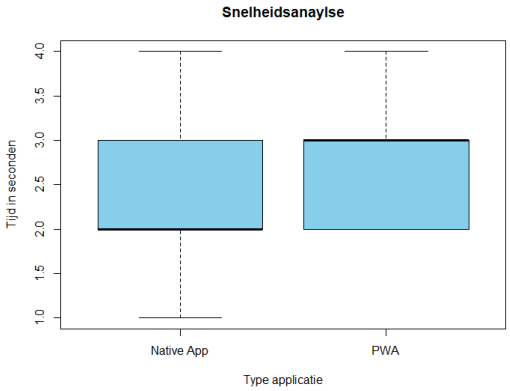
A.3 Methodologie

Om de performantie van de verschillende apps te vergelijken zal er twee maal eenzelfde applicatie gebouwd worden. Op android wordt deze gemaakt met framework kotlin, de PWA zal gemaakt worden met behulp van het framework vue.js in combinatie met nuxt. Op deze apps zullen verschillende soorten operaties uitgevoerd kunnen worden. Elk van deze operaties zal meermaals worden uitgevoerd en ondertussen zal de snelheid getest worden. Deze verkregen resultaten zullen vervolgens met elkaar vergeleken worden om te bepalen welke applicatie het meest performant is. Naast het onderzoek naar de performantie zal er ook onderzoek worden gedaan naar hoeveel ruimte deze app inneemt op een apparaat. Ook deze resultaten zullen worden vergeleken.

A.4 Verwachte resultaten

Om de resultaten voor te stellen wordt er gebruik gemaakt van een boxplot, zoals te zien is op figuur 1. Er zal er ook nog een tabel komen die de boxplot weergeeft. De getallen die hierin terug te vinden zijn: het gemiddelde, het maximum, het minimum, eerste kwadrant en derde kwadrant. Hierdoor krijgen we een goed overzicht om de resultaten te vergelijken tussen de PWA en native app.

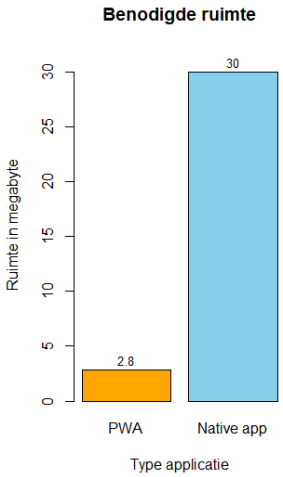
Voor de resultaten van de benodigde ruimte weer te geven zal gebruik gemaakt worden van een historiek, zoals te zien is op figuur 3. Er zal zoals bij de vergelijking van de tijd ook een tabel zijn met alle info die de historiek weergeeft.



Figuur A.1: Voorbeeld van boxplot die de snelheid vergelijkt

	min.	1ste kwa-drant	Gem.	3de kwa-drant	max.
PWA	2	2	2.714	3	4
native app	1	1	2.429	3	4

Figuur A.2: Voorbeeld van een tabel die de snelheid vergelijkt



Figuur A.3: Voorbeeld van historgram die de benodigde ruimte vergelijkt

Type applicatie	Benodigde ruimte
progressive web app	2.8 MB
native app	30

Figuur A.4: Voorbeel van tabel die de benodigde ruimte vergelijkt

A.5 Verwachte conclusies

Er wordt verwacht dat de snelheid niet significant verschillend zal zijn tussen de PWA en de native app. De native app haalt enkel de content op waardoor deze net iets sneller zal zijn. De PWA moet alle data ophalen samen met de layout , dit is de data moet weergegeven worden. Hierdoor zal de PWA een x tal milliseconden trager zijn dan de native applicatie

Verder wordt er verwacht dat de benodigde ruimte op een apparaat significant minder zal zijn dan de native app. Dit komt doordat de native app op het apparaat zelf geïnstalleerd is. Hierdoor staat er een standaard layout geïnstalleerd op het apparaat, wat redelijk veel ruimte inneemt. De PWA haalt zowel zijn inhoud als zijn layout van een server. Hierdoor kan een PWA de benodigde ruimte op het apparaat beperken.

Bibliografie

Babiuk, T. (2018, oktober 23). Choosing the best cross-platform framework for sharing code between iOS and Android. Verkregen van https://medium.com/@tomasz.babiuk/choosing-the-best-cross-platform-framework-for-sharing-code-between-ios-and-android-fa45b8162c81

Developers, A. (2019, december 27). Data Binding Library. Verkregen van https://developer.android.com/topic/libraries/data-binding

Developers, A. (2020, januari 22). LiveData Overview. Verkregen van https://developer.android.com/topic/libraries/architecture/livedata

East, D. (2017, september 14). What is Server-Side Rendering? (Server-side Rendering with JavaScript Frameworks). Verkregen van https://www.youtube.com/watch?v=GQzn7XRdzxY

Experius. (2019). Progressive Web Apps: de toekomst voor webshops is hier. Verkregen van https://www.experius.nl/pwa

Firtman, M. (2018, maart 30). Progressive Web Apps on iOS are here. Verkregen van https://medium.com/@firt/progressive-web-apps-on-ios-are-here-d00430dee3a7

Firtman, M. (2019, januari 31). Google Play Store now open for Progressive Web Apps. Verkregen van https://medium.com/@firt/google-play-store-now-open-for-progressive-web-apps-ec6f3c6ff3cc

Google. (2020, april 7). Workbox Strategies. Verkregen van https://developers.google.com/web/tools/workbox/modules/workbox-strategies

JetBrains. (2020, april 21). Null Safety. Verkregen van https://kotlinlang.org/docs/reference/null-safety.html

Koin. (g.d.). What is Koin? Verkregen van https://insert-koin.io/

Kotlin. (2020, februari 14). Coroutine Basics. Verkregen van https://kotlinlang.org/docs/reference/coroutines/basics.html

Marjchrzak, T. A. (2018, januari 6). Progressive Web Apps: the Definite Approach to Cross-Platform Development?

Nuxt.js. (2019a, november 20). Configuration. Verkregen van https://nuxtjs.org/guide/configuration

Nuxt.js. (2019b, oktober 24). Plugins. Verkregen van https://nuxtjs.org/guide/plugins

Nuxt.js. (2020a, maart 31). Directory Structure. Verkregen van https://nuxtjs.org/guide/directory-structure

Nuxt.js. (2020b, maart 20). Modules. Verkregen van https://nuxtjs.org/guide/modules

Nuxt.js. (2020c, januari 23). Routing. Verkregen van https://nuxtjs.org/guide/routing

Nuxt.js. (2020d, maart 29). Vuex Store. Verkregen van https://nuxtjs.org/guide/vuex-store

Osmani, A. (2017, december 24). A Tinder Progressive Web App Performance Case Study. Verkregen van https://medium.com/@addyosmani/a-tinder-progressive-web-app-performance-case-study-78919d98ece0

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Gent: Academia Press.

Puysseleyr, P. D. (2018). VERGELIJKENDE STUDIE ANDROID: PERFORMANTIE VAN KOTLIN VS.JAVA BIJ DATA INTENSIEVE APPLICATIES.

Rajput, M. (2019, augustus 1). Best Frameworks for Building Progressive Web Apps. Verkregen van https://www.mindinventory.com/blog/best-progressive-web-apps-frameworks/

Saltis, S. (2020, april 6). What is a Progressive Web App? (And Do You Need One). Verkregen van https://www.coredna.com/blogs/progressive-web-app

SimiCard. (2020, februari 13). Publishing PWAs to Major App Stores: The Whys and Hows. Verkregen van https://www.simicart.com/blog/pwa-app-stores/#Apple%20App%20Store

Simon, G. (2018, februari 19). Vue SEO Tutorial with Prerendering. Verkregen van https://www.youtube.com/watch?v=pwHdFPEX4NA

Snipcart. (2019, september 6). Vue PWA: A Progressive Web Application Example With Nuxt. Verkregen van https://snipcart.com/blog/vue-pwa#repo

Steiner, T. (2018, april 27). What is in a Web View? An Analysis of Progressive Web App Features When the Means of Web Access is not a Web Browser.

Verwey, J. (2018, mei 31). Verschillen Web-app, Native-app, Hybride-app en Progressive-app. Verkregen van https://www.dailycreations.nl/posts/verschil-webapp-native-app-en-hybride-app

Vue.js. (2020a, januari 26). Components Basics. Verkregen van https://vuejs.org/v2/guide/components.html

Vue.js. (2020b, februari 24). Conditionals and Loops. Verkregen van https://vuejs.org/v2/guide/index.html#Conditionals-and-Loops

Vue.js. (2020c, februari 24). Declarative Rendering. Verkregen van https://vuejs.org/v2/guide/index.html#Declarative-Rendering

Vue.js. (2020d, februari 24). What is Vue.js? Verkregen van https://vuejs.org/v2/guide/

VueSchool. (g.d.). What is Nuxt.js? Verkregen van https://vueschool.io/lessons/what-is-nuxtjs

web docs, M. (2020, april 1). Web app manifests. Verkregen van https://developer.mozilla.org/en-US/docs/Web/Manifest

Wikipedia. (2019, september 5). Framework. Verkregen van https://nl.wikipedia.org/wiki/Framework

Wikipedia. (2020a, april 17). Dependency injection. Verkregen van https://nl.wikipedia.org/wiki/Dependency_injection

Wikipedia. (2020b, maart 10). Kotlin (programmeertaal). Verkregen van https://nl.wikipedia.org/wiki/Kotlin_(programmeertaal)

Wikipedia. (2020c, mei 3). Progressive web application. Verkregen van https://en.wikipedia.org/wiki/Progressive_web_application