Professionele Bachelor Toegepaste Informatica

Intelliway

Steffen Lenaerts

Promotoren:

Dhr. Bert Swinnen — IDA Mediafoundry
Dhr. Tom Schuyten — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2018

Professionele Bachelor Toegepaste Informatica

Intelliway

Steffen Lenaerts

Promotoren:

Dhr. Bert Swinnen — IDA Mediafoundry
Dhr. Tom Schuyten — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Na drie jaar studeren kon ik mijn stage eindelijk uitvoeren bij IDA Mediafoundry. Tijdens deze drie jaar heb ik enorm veel bijgeleerd op technisch vlak en heb ik een betere kijk gekregen op de effectieve werking binnen het werkveld. Hiervoor zou ik graag een aantal mensen willen bedanken.

In de eerste plaats wil ik graag IDA Mediafoundry bedanken dat ze mij de kans hebben gegeven om bij hun een leerrijke stage te mogen uitvoeren. Meneer Bert Swinnen wil ik bedanken om mij te ondersteunen tijdens mijn stage. Door hem heb ik de kans gekregen om onder professioneel toezicht praktische kennis op te doen.

Ik zou ook alle medewerkers van IDA Mediafoundry willen bedanken. Zij hebben er allemaal voor gezorgd dat ik mij meteen thuis voelde en ze gaven mij het gevoel dat ik deel uitmaakte van hun team. Ook zou ik graag meneer Thijs Lanssens bedanken omdat ik bij hem met al mijn technische vragen terecht kon.

Ook een speciaal dankwoord voor meneer Tom Schuyten. Hij heeft mij zeer goed begeleid tijdens het hele verloop van mijn stage. Bij hem kon ik steeds terecht voor alle mogelijke vragen in verband met mijn eindwerk.

Tot slot wil ik mijn ouders en vrienden bedanken voor hun aanmoediging en ondersteuning tijdens mijn stage.

Abstract

Meer en meer toeristen gebruiken tegenwoordig een applicatie om hun vakanties te plannen. In dit type applicatie krijgen toeristen een overzicht van heel wat activiteiten op hun vakantiebestemming. De bedoeling van deze stageopdracht is om een webapplicatie te maken waarbij toeristen een overzicht krijgen van allerlei vakantieactiviteiten gecombineerd met *augmented reality*.

Wanneer de toerist aan een wegwijzer staat, kan hij naar de website surfen. Hierdoor wordt de camera geopend en kan de gebruiker de *marker* op de wegwijzer scannen. Aan elke wegwijzer is een activiteit gekoppeld. Wanneer de gebruiker de *marker* scant, wordt er in *augmented reality* informatie over de activiteit getoond. Ook kan de gebruiker de informatie via de website bekijken. Wanneer de gebruiker op de website komt, heeft hij de mogelijkheid om zelf zijn locatie te selecteren of kan hij ervoor kiezen om zijn locatie te laten bepalen door zijn gps. Vanaf het moment dat de locatie van de gebruiker bepaald is, wordt er een lijst getoond van de verschillende activiteiten op deze locatie. Nu heeft de gebruiker de mogelijkheid om een bepaalde activiteit te selecteren. Nadat de gebruiker een activiteit heeft geselecteerd, krijgt hij meer details over deze activiteit te zien.

Zoals de meeste projecten, bestaat deze applicatie uit een frontend en backend. De frontend is geschreven in JavaScript met behulp van Vue.js en de backend is geschreven in Java met behulp van Spring Boot. De frontend communiceert met de backend om de data op te halen en deze te visualiseren voor de gebruiker. Dit keer wordt de data uit de backend niet in een standaard database opgeslagen, maar in Contentful. Contentful is een content-management-systeem. Om gebruik te maken van *augmented reality* wordt er in de frontend gebruik gemaakt van AR.js. Dit is een *marker based augmented reality framework* voor webapplicaties.

De onderzoeksopdracht komt voort uit de stageopdracht. In de stageopdracht is het de bedoeling om een applicatie te maken die informatie toont via *augmented reality*. Om deze informatie te tonen wordt er gebruik gemaakt van een *marker*. Een *marker* dient als herkenningspunt voor de applicatie. Door deze *marker* weet de applicatie waarop hij de *augmented reality content* moet tonen. Binnen andere *frameworks* is het mogelijk om zonder *markers* de virtuele content te tonen. De bedoeling van dit onderzoek is om na te gaan op welke manieren dit gerealiseerd wordt. Verder wordt er ook gecontroleerd of het mogelijk is om in een webapplicatie *markerless augmented reality* toe te passen.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte afkortingen

6DOF	6 degrees of freedom
AR	Augmented reality
API	Application programming interface
CMS	Content-management-system
COM	Concurrent odometry and mapping
CSS	Cascading style sheet
DTO	Data transfer object
FPS	Frames per second
HTML	Hyper text markup language
JSON	JavaScript object notation
P2P	Peer-to-peer
POI	Point of interest
REST	Representational state transfer
SDK	Software development kit
SLAM	Simultaneous localization and mapping
VIO	Visual-inertial odometry

Inleiding

IDA Mediafoundry heeft voor de toeristische sector een idee voor een stageopdracht uitgewerkt. Zij vinden dat toeristen vaak op zoek moeten gaan naar hun huidige locatie en hun volgende bestemming. Vervolgens kwam IDA met het idee om deze toeristen te helpen. Wanneer de toeristen aan een wegwijzer staan, kunnen ze een marker scannen. Wanneer ze deze *marker* scannen, wordt er in *augmented reality* informatie getoond over een bepaalde bestemming.

In deze opdracht dienen de toeristen steeds aan een wegwijzer te staan om een *marker* te scannen zodat hierop de informatie getoond kan worden. In de onderzoeksopdracht worden verschillende *augmented reality* platformen met elkaar vergeleken om zo na te gaan of *markerless augmented reality* mogelijk is in een webapplicatie.

Deze stageopdracht werd uitgevoerd bij IDA Mediafoundry. IDA maakt deel uit van de Xplore group en is gelegen op de Corda campus. IDA focust zich vooral op het bouwen van digitale ervaringen met een focus op web en e-commerce.

Dit project bestaat uit een frontend en backend. De frontend is geschreven in Vue.js en de backend met Java en het Spring boot *framework*. In de frontend wordt voor elk scherm de nodige componenten aangemaakt. Ook wordt er in de frontend gebruik gemaakt van AR.js. Met AR.js wordt de *augmented reality* ervaring gerealiseerd. Aan elke *marker* wordt de juiste virtuele content gekoppeld.

In de backend wordt er gewerkt met een drie lagen structuur. Zo zijn er de nodige *controllers*, *services* en *repositories* voorzien. Alle data die de backend gaat ophalen, komt vanuit het CMS Contentful.

In dit eindwerk wordt er dieper ingegaan op de structuur van zowel de backend en de frontend. Ook wordt het onderzoek verder uitgewerkt en wordt er een conclusie rond *markerless augmented reality* in een webapplicatie gevormd.

I. Stageverslag

1 Bedrijfsvoorstelling

1.1 Situering bedrijf

IDA Mediafoundry is een bedrijf dat deel uitmaakt van de Xplore group. De Xplore group maakt op zijn beurt deel uit van de Cronos group. In 2005 is IDA opgericht als het Adobe competence center binnen de Xplore Group. Een jaar later is hieruit IDA N.V. opgericht. Het doel van IDA N.V. was om de expert te worden binnen Adobe Enterprise solutions. Door de jaren heen hebben ze eraan gewerkt om hun expertise uit te breiden. Bovendien is IDA vorig jaar, in 2018 Adobe Experience Cloud partner van het jaar geworden.

Bij IDA Mediafoundry maken ze digitale applicaties met een sterke focus op web en e-commerce. Dit wil zeggen dat ze volledige websites bouwen die te integreren zijn met volledige backend systemen. Bovendien zorgen ze er ook voor dat hun klanten de volledige *customer journey* kunnen volgen. Dit wil zeggen dat er op basis van de persoon die de website bezoekt de meest geschikte *content* getoond kan worden.

Het hoofdkantoor van IDA Mediafoundry is gelegen in Kontich. IDA heeft ook kantoren in Merelbeke, Utrecht en Hasselt. Mijn stage gaat door op de Corda campus in Hasselt. Op dit moment werken er ongeveer 50 werknemers bij IDA Mediafoundry.

1.2 Methodologie

Binnen IDA Mediafoundry wordt er gebruik gemaakt van agile methodologieën. Elke dag starten ze gewoonlijk met een scrumsessie. Hierin wordt besproken wie wat gedaan heeft, wie wat gaat doen en of ze problemen hebben ondervonden. Verder maken ze ook gebruik van verschillende Atlassian-tools. Er wordt gewerkt via tweewekelijkse sprints. Voor elke sprint worden alle doelen bepaald. Op Jira worden deze doelen bijgehouden. Ook wordt er gebruik gemaakt van Confluence. Hierop wordt alles gedocumenteerd om zo een duidelijk beeld te krijgen binnen het project.

Verder gebruiken ze ook nog Bitbucket pipelines of Jenkins om continous integration te voorzien.

1.3 Expertise

IDA helpt zijn klanten om Adobe Marketing Cloud-oplossingen te integreren in bestaande e-commerce systemen. Ook worden deze marketing platformen aangepast aan de noden van de klant.

2 Voorstelling Stageopdracht

2.1 Probleemstelling

Heel wat toeristen maken gebruik van een applicatie om hun reis of uitstap te plannen. In dit type applicatie wordt er heel wat informatie getoond over verschillende bezienswaardigheden in de omgeving. Tijdens de stage is het de bedoeling om een gelijkaardige applicatie te maken en ook nog uit te breiden met innovatieve technologieën zoals *augmented reality*. Wanneer een toerist bij een wegwijzer staat, heeft hij de mogelijkheid om een AR *marker* te scannen. Wanneer de *marker* gescand wordt, krijgt de toerist in *augmented reality* informatie te zien over een activiteit. Deze activiteit is afhankelijk van de wegwijzer waar de toerist aan staat.

Tijdens deze stage wordt er met verschillende tools en programmeertalen gewerkt. Voor de frontend wordt er gebruik gemaakt van Vue.js en de backend is geschreven in Java met het Spring Boot *framework*. De backend communiceert deze keer niet met een klassieke database, maar met de tool Contentful. Contentful is een infrastructuur om data op te slaan onafhankelijk van een platform. Voor het *augmented reality* gedeelte wordt er gebruik gemaakt van AR.js.

2.2 Stakeholders

Binnen deze stageopdracht zijn er maar een beperkt aantal stakeholders. De eerste stakeholder is IDA Mediafoundry zelf. Deze stageopdracht is een project voor IDA en zij willen uiteraard dat deze opdracht een meerwaarde geeft aan hun bedrijf.

De tweede stakeholder is de toerist zelf. Het is de bedoeling dat de toeristen zoveel mogelijk gebruik gaan maken van de applicatie.

De activiteiten of bezoekplaatsen zijn de derde stakeholder. Omdat er in de applicatie, informatie over hun getoond wordt, moeten zij hun goedkeuring geven om deze informatie te gebruiken.

Ook dient er binnen dit systeem een administrator te zijn om alle gegevens te beheren. Wanneer er bijvoorbeeld nieuwe activiteiten bijkomen of activiteiten trekken hun goedkeuring in, dan moet de administrator de nodige aanpassingen kunnen maken in het *content-management-system*.

2.3 Tools

2.3.1 Vue.js

Vue.js is een *progressive JavaScript framework* om *user interfaces* en *single-page applications* mee te bouwen. *Single-page applications* zijn websites waarbij telkens bepaalde componenten van een pagina worden geladen en weergegeven. Hierdoor moet niet steeds de volledige pagina herladen worden. Bij niet *single-page applications* dienen alle componenten herladen te worden. Ook de componenten waar geen wijzigingen in gebeuren.

Omdat er met Vue.js *Single-page applications* gebouwd worden, wordt er binnen Vue.js gewerkt met componenten. Een component dient ervoor om herbruikbare code samen te krijgen in een file. Elke file bevat drie onderdelen namelijk HTML, CSS en JavaScript. [1]

2.3.2 Spring boot

Voor de backend wordt er gebruik gemaakt van Spring boot. Spring boot is een *framework* dat gebouwd is op het Spring *framework*. Spring boot zorgt ervoor dat er binnen Java gemakkelijk REST applicaties gebouwd kunnen worden.

2.3.3 Contentful

Contentful is een *content-management-sytem*. Alle data wordt hier aangemaakt en opgeslagen. Contentful maakt gebruik van *content models.* Deze modellen komen overeen met de structuur van de datamodellen in de backend. Binnen Contentful bevatten de modellen ook een aantal velden. Aan deze velden kan het juiste datatype toegekend worden. Op basis van de verschillende modellen kan er vervolgens data worden toegevoegd.

Om de data op te halen heeft Contentful een aantal API's voorzien. De 2 belangrijkste zijn de *Content Delivery API* en de *Content Management API*. Via de *Content Delivery API* wordt de data opgehaald in JSON formaat. De *Content Management API* dient om data aan te maken en te updaten. [2]

2.3.4 AR.js

AR.js is een *JavaScript framework* om heel gemakkelijk *augmented reality* toe te voegen aan webapplicaties. Met slechts een tiental regels HTML code is het al mogelijk om een *augmented reality* ervaring aan te bieden in een webapplicatie. De code hiervan wordt weergegeven in figuur 1.

```
<!doctype HTML>
<html>
<script src="https://aframe.io/releases/0.9.0/aframe.min.js"></script>
<script src="https://cdn.rawgit.com/jeromeetienne/AR.js/1.6.2/aframe/build/aframe-ar.js"></script>
  <body style='margin : 0px; overflow: hidden;'>
    <a-scene embedded arjs>
      <a-marker preset="hiro">
          <a-box position='0 0.5 0' material='color: yellow;'></a-box>
      </a-marker>
      <a-entity camera></a-entity>
    </a-scene>
  </body>
</html>
```

Figuur 1: Voorbeeldcode AR.js

Deze code zorgt ervoor dat op de standaard *marker* van AR.js zelf een gele kubus getoond wordt.

Op regel 3 en 4 worden AR.js en a-frame geïmporteerd. A-frame wordt geïmporteerd omdat AR.js hier gebruik van maakt. Voor beide *frameworks* worden de nodige *script tags* voorzien.

Nadien wordt er een 3D *scene* voorzien. In deze *scene* wordt de *augmented reality content* voorzien. In dit voorbeeld wordt er een simpele kubus getoond. Natuurlijk moet AR.js ook weten waarop de *content* getoond moet worden. Om dit te realiseren wordt de standaard Hiro *marker* van AR.js gebruikt. Hieronder in figuur 2 wordt de Hiro *marker* getoond.

Figuur 2: Hiro marker

Als laatste moet AR.js nog de toestemming krijgen om de camera te gebruiken. Nu kan de applicatie gestart worden en wanneer de Hiro *marker* in beeld is, wordt er een gele kubus op getoond. [3]

2.3.4.1 Markers

Buiten de standaard Hiro *marker*, biedt AR.js ook nog andere *markers* aan. Er zijn binnen AR.js twee verschillende soorten *markers*. De eerste soort wordt *Pattern markers* genoemd.

Dit soort *marker* is gebaseerd op een afbeelding. Deze afbeelding wordt omringd door een zwarte rand. Standaard zit er maar één *pattern marker* in AR.js, namelijk de Hiro *marker*.

Het is mogelijk om eigen *pattern markers* te maken. Om gebruik te maken van deze *markers* moet er wel rekening gehouden worden met een aantal vereisten. Zo moeten de *markers* een vierkant zijn en niet te complex, zodat AR.js nog in staat is om de *marker* te detecteren. Als er in de applicatie meerdere *markers* gebruikt worden, moet er genoeg onderscheid zijn zodat AR.js de *markers* nog kan detecteren. Anders zou het kunnen dat AR.js de verkeerde *marker* detecteert en zo de verkeerde *content* gaat tonen. Ook mogen ze geen kleuren bevatten. Alleen zwart, wit en grijs zijn toegelaten.

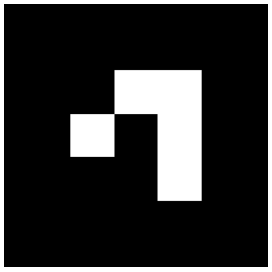
Het voordeel van eigen *markers* te gebruiken is dat de *markers* zo gemaakt kunnen worden dat deze in de context van de applicatie passen. Jammer genoeg is er ook een nadeel aan het gebruiken van eigen *markers*. AR.js is iets trager in het herkennen van zelfgemaakte *markers*. Hierdoor duurt het langer om de virtuele *content* te tonen.

De tweede soort *marker* is een barcode. Bij barcodes worden er geen patronen gebruikt die de gebruiker zelf kan kiezen. In plaats daarvan is een barcode een vooraf bepaald patroon van witte en zwarte vierkanten. Deze vierkanten zijn ingedeeld in een matrix patroon. Het is mogelijk om de grootte van de matrix in te stellen. Afhankelijk van de grootte zijn er een vast aantal barcodes beschikbaar. Als de matrix te groot wordt ingesteld, kan het zijn dat AR.js de verkeerde barcode herkent en hierdoor de verkeerde *content* toont. Elke barcode is achterliggend gekoppeld aan een getal. Door dit getal kan de juiste barcode geïdentificeerd worden.

Ook het gebruik van barcodes heeft een aantal voor- en nadelen. Het grootste voordeel van barcodes te gebruiken is dat AR.js deze sneller kan detecteren. Hierdoor gaat de *content* sneller getoond worden op de barcode.

Het nadeel van barcodes is dat er een vast aantal barcodes beschikbaar zijn. Als de matrix op de grootst mogelijke grootte wordt ingesteld zijn er wel meer dan 8 miljard *markers* beschikbaar. Maar hierdoor is de kans wel weer groter dat AR.js de verkeerde barcode gaat detecteren.

Ook zien barcodes er in het algemeen minder aantrekkelijk uit en zijn ze moeilijker te onderscheiden. Barcodes zijn een aantal zwarte en witte kubussen. Hierdoor zou de gebruiker bijvoorbeeld de verkeerde *marker* kunnen scannen waardoor de verkeerde *content* getoond gaat worden. In figuur 3 kan je een voorbeeld zien van een barcode. [4]



Figuur 3: Barcode met waarde 5

3 Uitwerking Stageopdracht

3.1 Verschillende onderdelen

De structuur van deze applicatie is helemaal niet complex. Zoals de meeste applicaties is er een frontend die communiceert met de backend om de juiste data op te halen. Buiten de normale onderdelen zoals de frontend en de backend wordt er ook een *content-management-sytem* en een *framework* voor *augmented reality* gebruikt.

3.1.1 Frontend

De frontend bestaat uit 2 belangrijke onderdelen. Namelijk de website zelf en het *augmented reality* gedeelte. Voor het *augmented reality* wordt AR.js gebruikt. In de *browser* van de gebruiker wordt de camera geopend om de *marker* te scannen. Wanneer de gebruiker aan de wegwijzer staat, wordt via de locatie van de wegwijzer bepaald welke virtuele *content* op de *marker* getoond moet worden.

Nadien kan de gebruiker ook nog de website bezoeken. De website dient als *user interface* voor de gebruiker. Als eerste moet de locatie van de gebruiker bepaald worden. Dit kan hij doen op twee manieren. De eerste manier is door de locatie handmatig uit een lijst te selecteren. De tweede manier is om de locatie te laten bepalen door de gps van de gebruiker.

Nadat de locatie bepaald is, wordt er genavigeerd naar het volgende scherm. Op dit scherm wordt een lijst van activiteiten op basis van de gekozen locatie getoond. Per activiteit wordt er een kleine foto, de titel en een korte beschrijving getoond. Indien de gebruiker meer informatie wil over een activiteit, dient hij op de activiteit te klikken.

Vervolgens wordt hij weer naar een nieuw scherm genavigeerd. Op dit scherm krijgt de gebruiker meer informatie over de gekozen activiteit. Voorbeelden zijn: openingsuren, adresgegevens en de prijs.

3.1.2 Componenten frontend

Zoals hierboven beschreven, zijn er een aantal schermen in de frontend. Voor elk scherm is er een component aangemaakt die de nodige functionaliteit bevat.

Het eerste component is de *header*. Deze component is op elk scherm te zien en bevat ook geen functionaliteit. Het enige doel van deze component is puur lay-out gericht.

Binnen het eerste scherm, waar de gebruiker zijn locatie moet bepalen, worden er twee componenten gebruikt. De eerste is de component om de locatie te selecteren uit de lijst. De tweede component dient ervoor om de locatie van de gebruiker te bepalen op basis van zijn huidige locatie. Wanneer de gebruiker op de knop klikt, navigeert hij naar het volgende scherm en wordt zijn locatie meegestuurd.

Het derde component is een overzicht van alle activiteiten van de gekozen locatie. Binnen deze component wordt er gebruik gemaakt van meerdere componenten om elke activiteit op dezelfde manier te tonen. Wanneer de gebruiker op een activiteit klikt, navigeert hij naar het laatste scherm.

Op dit laatste scherm is er een component voorzien om meer informatie over de gekozen activiteit te zien. Ook is er nog een component voorzien voor de camera. Hierin wordt de camera geopend en de juiste *marker* aan de juiste *content* gekoppeld.

Om te navigeren tussen de verschillende componenten wordt Vue.js router gebruikt.

3.1.3 Backend

De backend is eigenlijk alles wat op de server draait. De backend gaat alle data ophalen in het *content-management-system* en vervolgens wordt de data doorgegeven aan de frontend, zodat deze gevisualiseerd kan worden voor de gebruiker.

Binnen de backend wordt er gebruik gemaakt van een drie lagen structuur. Voor elk object is er een *controller, service* en *repository* voorzien.

De verantwoordelijkheid van de *repository* is om een connectie te maken naar het *content-management-system* en vervolgens wordt de juiste data opgehaald en bewaard in objecten van het juiste type.

Binnen de *service* wordt de *repository* aangesproken. Ook wordt er een *entity* omgezet naar een *data transfer object*. Vaak is het zo dat de structuur van de data die intern in het systeem zit en de data die naar buiten wordt gestuurd niet volledig overeenkomt. Om zo een *entity* om te zetten naar een *data transfer object* wordt er gebruik gemaakt van *mappers* van spring zelf. Deze gaan op *runtime* bepalen hoe de omzetting tussen *entity* en dto dient te gebeuren.

Als laatste hebben we dan nog de *controllers*. Deze gaan de juiste service aanspreken en gaan vervolgens als de *request* van de gebruiker lukt, de juiste data en een *response code* teruggeven.

Ook is er beveiliging voorzien op de *endpoints*. Hiervoor wordt er gebruik gemaakt van *Basic Spring authentication*. Dit wil zeggen dat de gebruiker zich moet aanmelden met behulp van een gebruikersnaam en wachtwoord. Er is beveiliging voorzien op alle methoden van de backend. Als de gebruiker data wil opvragen, toevoegen, verwijderen of wijzigen, dient hij zich te authentiseren.

3.1.4 Contentful

Binnen Contentful wordt alle data opgeslagen. Om de data op te delen, wordt er gebruik gemaakt van verschillende *content models*. Deze modellen zijn gebaseerd op het *domain model* dat ook in de *backend* gebruikt wordt. Dus dit wil ook zeggen dat er tussen de verschillende modellen relaties liggen.

In figuur 4 worden alle huidige *content models* getoond.

Name	Description
locations	A list of all the available locations
marker	marker: number of barcode
poi	A list of all the points of interests
waypoint	List of all waypoints

Figuur 4: Contentful - Content models

Op dit moment zijn er binnen Contentful slechts vier modellen gedefinieerd.

Het eerste model is 'locations'. Dit model stelt een stad voor. Van een stad zelf worden er niet zoveel gegevens bijgehouden. Op dit moment wordt alleen de naam van de stad bijgehouden. Verder is er nog een *reference field* naar 'poi'. Dit wil zeggen dat een locatie een lijst van activiteiten bijhoudt.

Vervolgens is er het model 'poi' of *point of interest.* Een 'poi' is een activiteit die kan plaatsvinden in een stad. Aangezien de hele stageopdracht draait rond de verschillende activiteiten, worden er heel wat gegevens bijgehouden van een activiteit. Deze gegevens bevatten vooral informatie van de activiteit zoals de naam, adresgegevens, openingsuren en de prijs.

De laatste twee modellen zijn belangrijk voor het *augmented reality* gedeelte van de applicatie. Als eerste is er het model 'waypoint'. Dit stelt een wegwijzer voor. Van een wegwijzer zelf wordt er niet zoveel informatie bijgehouden. Enkel de locatie en een naam van de wegwijzer worden bijgehouden. De enige functie van de naam is voor identificatie van de verschillende wegwijzers.

Het laatste model is de *marker* zelf. Binnen de applicatie worden er barcodes gebruikt. Deze barcodes bevatten achterliggend een nummer. Binnen contentful wordt dit nummer opgeslagen. Verder bevat een *marker* het model dat er getoond moet worden. Dit is een *reference* naar het type 'poi'. Het laatste veld is een *reference* naar het type 'waypoint'. Op deze manier is het duidelijk aan welke wegwijzer de *marker* gekoppeld is.

3.1.5 AR.js

Voor het *augmented reality* gedeelte wordt er gebruik gemaakt van AR.js. AR.js is een *marker based framework.* Binnen AR.js moet de juiste *content* aan de juiste *marker* gekoppeld worden. Dit kan met slechts enkele regels code. In figuur 5 wordt aan de *marker* met barcode 5 de waarde 'Corda Campus' gekoppeld.

```
<a-scene arjs="detectionMode: mono_and_matrix; matrixCodeType: 3x3;">
  <a-marker type="barcode" value="5">
    <a-entity position="0 0 0" text="value: Corda Campus"></a-entity>
  </a-marker>
  <a-entity camera></a-entity>
</a-scene>
```

Figuur 5: Koppelen content aan marker

Zoals eerder al aangehaald zijn er binnen AR.js twee verschillende soorten *markers.* Namelijk *pattern markers* en barcodes. Binnen de stage wordt er gewerkt met barcodes. AR.js kan deze sneller herkennen en daardoor de virtuele *content* sneller tonen. Ook is het gemakkelijker om barcodes te gebruiken. Aan elke barcode is achterliggend een getal gekoppeld. Binnen Contentful wordt dit getal gebruikt als *marker.* Dit getal opvragen is gemakkelijker en efficiënter dan het opvragen van een afbeelding.

Binnen AR.js moet ervoor gezorgd worden dat aan de verschillende barcodes de juiste objecten worden gekoppeld. Er is een systeem voorzien om alleen de wegwijzers op te halen die bij de gebruiker in de buurt staan. De positie van elke wegwijzer wordt in het systeem bijgehouden. Door de afstand te berekenen tussen de positie van de gebruiker en de wegwijzer, kunnen de tien dichtstbijzijnde wegwijzers bepaald worden.

Vanaf het moment dat de tien dichtstbijzijnde wegwijzers bepaald zijn, kunnen de *makers* die op deze wegwijzers hangen, opgevraagd worden. Aan elke *marker* is de juiste activiteit gekoppeld. Vanaf het moment dat in de frontend de juiste data aanwezig is, gaat AR.js ervoor zorgen dat de *marker* gekoppeld wordt aan de juiste data.

4 Reflectie

Binnen deze stageopdracht worden er heel wat technologieën gebruikt. Zowel technologieën waar ik via de school mee in aanraking ben gekomen, als technologieën die ik nog niet eerder gebruikte. Dit is exact wat ik zocht in een professionele stage. Door de hulp en professionele kennis van de werknemers van IDA heb ik hierover heel wat bijgeleerd. Ik kreeg regelmatig tips over hoe ik zaken beter kon aanpakken en hoe het in de praktijk toegepast wordt. Hierbij viel het op dat er op school een grondige basiskennis wordt aangebracht, maar dat je toch pas echt leert op het werkveld zelf.

Naast de technologieën die we in school aangeleerd krijgen, ben ik ook in contact gekomen met nieuwe technologieën: AR.js, Contentful en Vue Material. Dit zorgde voor een extra uitdaging aan de stage. Vooraleer ik deze technologieën kon gebruiken, moest ik eerst heel wat opzoeken rond de mogelijkheden van deze technologieën.

Deze technologieën gebruiken was niet altijd even gemakkelijk. De documentatie van Contentful voor Java was niet altijd correct. Hierdoor moest ik soms lang zoeken naar de oplossing van een fout. Door de hulp van het personeel van IDA ging dit een stuk sneller.

Ook waren er een aantal problemen met AR.js. Zo stond de camera maar half in beeld en bleef de camera op de achtergrond staan wanneer er genavigeerd werd van de camera naar de website. Voor deze problemen kon ik geen oplossingen vinden op het internet. Hierdoor heb ik zelf lang moeten zoeken naar de oorzaak en mogelijke oplossingen. Uiteindelijk heb ik voor deze problemen wel oplossingen kunnen vinden. Uit deze ervaring heb ik geleerd dat ik sneller hulp moet vragen. Met nieuwe technologieën werken, vraagt tijd en extra inspanningen. Maar hoe meer ik ermee werkte, hoe beter alles begon te lukken.

De behulpzaamheid en vriendelijkheid van de werknemers van IDA hebben ervoor gezorgd dat ik mij snel thuis voelde. Hierdoor durfde ik sneller feedback en hulp vragen. Zo twijfelde ik over hoe het *augmented reality* gedeelte toegepast moest worden. Doordat ik op tijd hulp heb gevraagd, heeft Bert mij geholpen en tijdig in de juiste richting geduwd.

Persoonlijk vind ik dat ik er een goede stageperiode heb opzitten. Ik heb veel ervaring kunnen opdoen. Zowel op professioneel als op persoonlijk vlak ben ik gegroeid. Ze hebben mij heel wat kansen gegeven om er een mooi eindresultaat van te maken.

II. Onderzoekstopic

1 Onderzoeksvraag

Augmented reality is een technologie die steeds meer en meer gebruikt wordt. Bekende *augmented reality* toepassingen zijn de filters van snapchat of de hype van 2016, Pokémon Go. Niemand kan ontkennen dat *augmented reality* aan een opmars bezig is. In de meeste gevallen dient de gebruiker een app te downloaden. Dit komt voornamelijk omdat *augmented reality* op iOS of Android veel verder ontwikkeld is en hierdoor meer mogelijkheden heeft.

Via *augmented reality* wordt er een virtueel object getoond in de echte wereld via de camera van een smartphone of laptop. Om te weten waar dit virtuele object getoond kan worden, zijn er verschillende mogelijkheden. In de *augmented reality frameworks* voor webapplicaties worden *markers* gebruikt. *Markers* zijn eigenlijk de herkenningspunten voor de applicatie. De applicatie gaat de virtuele objecten op deze *marker* tonen.

Op de *augmented reality frameworks* voor iOS en Android is het mogelijk om zonder deze *markers* te werken. In deze *frameworks* wordt er op andere manieren bepaald waar het virtuele object getoond moet worden. De bedoeling van dit onderzoek is om na te gaan hoe *markerless augmented reality* wordt toegepast binnen verschillende *frameworks.* Bovendien wordt er ook gekeken of het mogelijk is om *markerless augmented reality* toe te passen in een webapplicatie. In dit onderzoek wordt er een vergelijking gemaakt tussen ARKit (iOS), ARCore (Android), Ar.js (web).

2 Onderzoeksmethode

Het onderwerp van dit onderzoek komt voort uit de stageopdracht. In de stageopdracht is het de bedoeling om een webapplicatie te maken die informatie toont via *augmented reality*. Om deze informatie te tonen worden er *markers* gebruikt. Hierdoor weet de applicatie waarop de informatie getoond moet worden. De bedoeling van dit onderzoek is om na te gaan of het mogelijk is om een *markerless augmented reality* toepassing te bouwen in een webapplicatie.

De eerste stap binnen dit onderzoek bestaat uit een literatuurstudie. Binnen deze literatuurstudie worden de eerder vermelde *frameworks* bestudeerd. Er wordt vooral gekeken naar de verschillende *features* van de drie *frameworks.* Nadien wordt er gekeken of er een web *framework* is dat dezelfde *features* gebruikt om *markerless augmented reality* in een webapplicatie mogelijk te maken. Tot slot wordt er gekeken welk *framework* het meest geschikt is om een *markerless* applicatie in te bouwen.

Het onderzoek is geslaagd wanneer er duidelijkheid is over de status van *markerless augmented reality* binnen een webapplicatie.

3 Literatuurstudie

3.1 Artikel 1: ARCore vs ARKit

ARCore en ARKit zijn platformen om *augmented reality* applicaties mee te ontwikkelen. ARCore is ontwikkeld door Google en ARKit is ontwikkeld door Apple. Google en Apple geloven in de toekomst van *augmented reality*. Hierdoor zijn ze constant hun eigen platformen aan het verbeteren om zo de grootste speler op de markt te worden.

Apple heeft in de zomer van 2017 ARKit uitgebracht. Google heeft pas een half jaar later ARCore op de markt gebracht waardoor Apple als eerste een *augmented reality tool* op de markt bracht en zo een voorsprong opbouwde. Ontwikkelaars konden vanaf nu *augmented reality* applicaties bouwen voor iPhones met een A9 processor of hoger.

Sinds februari 2018 is Apple niet meer de enige aanbieder van een *augmented reality* platform. Op dit moment is ARCore beschikbaar op high-end smartphones met Android versie 7.0 of 8.0.

Met beide *augmented reality* platformen worden *native* applicaties gebouwd. Dit maakt het voor ontwikkelaars een stuk gemakkelijker om mobiele *augmented reality* applicaties te bouwen.

ARCore en ARKit zijn in de basis heel gelijkaardig. Ze maken beide gebruik van de drie belangrijkste onderdelen om aan *augmented reality* te doen, namelijk *environmental understanding, motion tracking en light estimation.* Deze technieken dienen er vooral voor om de echte wereld te begrijpen om zo de virtuele objecten realistisch in de wereld te plaatsen. Beide platformen hebben deze geïmplementeerd, maar toch zijn er een aantal kleine verschillen.

Om aan *augmented reality* te doen, moet het toestel zijn positie en rotatie kunnen bepalen. Binnen ARCore en ARKit wordt dit op dezelfde manier gerealiseerd, namelijk door *visual inertial odometry (VIO)*. VIO gaat gebruik maken van de camera en sensoren van het toestel om beweging waar te nemen. Door deze technologie kunnen de virtuele objecten preciezer gepositioneerd worden in de wereld.

Plane detection dient ervoor om te weten waarop de virtuele objecten getoond moeten worden. ARCore en ARKit kunnen beide horizontale en verticale vlaktes detecteren. Hierdoor kunnen de objecten op een realistische manier op de oppervlakte geplaatst worden.

Beide platformen maken gebruik van *plane detection*, alleen geven ze hier een andere naam aan. Binnen ARCore wordt dit *environmental understanding* genoemd en binnen ARKit *scene understanding.*

Beide platformen hebben een vorm van light estimation geïmplementeerd. Hierbij wordt er via de sensoren van de camera de lichthoeveelheid in de echte wereld bepaald. Vervolgens wordt dezelfde lichthoeveelheid op het virtuele object toegepast, waardoor het virtuele object realistischer lijkt.

Beide platformen werken op een heel gelijkaardige manier. Toch zijn er een aantal kleine verschillen. ARCore is iets beter in het in kaart brengen van de wereld. Dit komt omdat in ARKit alleen de meeste recente data wordt opgeslagen. Hierdoor duurt het iets langer om een omgeving de herlokaliseren.

Ook trackt ARCore meer *feature points* tijdens het in kaart brengen van een omgeving. Dit wil zeggen dat de *gemapte* omgeving sneller wordt uitgebreid in ARCore. Maar ARKit is wel beter in het verschil te herkennen tussen horizontale en verticale vlakten.

Beide platformen staan op hetzelfde niveau op basis van het naast elkaar plaatsen van virtuele objecten. Soms komen de objecten niet natuurlijk over. Het lijkt zo alsof de objecten in de ruimte zweven.

Beide platformen zijn heel gelijkaardig aan elkaar en de keuze hangt vaak af van de ontwikkelaar. Als er een applicatie voor Android gebouwd wordt, is ARCore de beste keuze en anders kies je voor ARKit. [5]

3.2 Artikel 2: Native vs Web AR

Om een *augmented reality* ervaring aan te bieden zijn er op dit moment twee mogelijkheden. Zo kan *augmented reality* gebruikt worden in een mobiele applicatie of in een webapplicatie. Aan beide mogelijkheden zijn een aantal voor- en nadelen verbonden.

Voor 2017 was *augmented reality* op smartphones niet zo belangrijk. Sinds ARKit en ARCore op de markt zijn, is het voor ontwikkelaars veel gemakkelijker om *augmented reality* applicaties te bouwen. Deze *frameworks* maken gebruik van *computer vision* en *machine learning.* Dit wordt ook vaak SLAM genoemd. SLAM staat voor *simultaneous localization and mapping,* dit wil zeggen dat de applicatie de omgeving in kaart gaat brengen.

ARKit en ARCore hebben *augmented reality* voor mobiele applicaties enorm verder geholpen. *Augmented reality* voor web heeft nog niet de volledige mogelijkheden van SLAM benut. Web *augmented reality* heeft wel het voordeel dat het voor iedereen die een smartphone heeft beschikbaar is.

Augmented reality voor web zijn eigenlijk websites waar *augmented reality* aan toegevoegd is. Binnen deze *augmented reality* websites wordt er gebruik gemaakt van JavaScript om *computer vision libraries* te *deployen*. Dit is nog maar net mogelijk gemaakt door updates aan WebRTC. WebRTC is een protocol dat toestemming vraagt aan de gebruiker om de camera te gebruiken.

Een ander voordeel van web *augmented reality* is dat het platform onafhankelijk is. Dit gaat werken op de meeste Android & iOS toestellen. De toestellen moeten wel de laatste versie van WebRTC hebben. Anders kan er geen toegang gevraagd worden tot de camera en gaat hij geen *augmented reality* ervaring kunnen beleven.

Toch zijn er voor *augmented reality* in web nog steeds wat beperkingen. Web *Augmented reality* staat nog niet op hetzelfde niveau als *augmented reality* in mobiele applicaties. ARKit en ARCore staan veel verder in het herkennen van de wereld en kunnen hierdoor de virtuele objecten realistischer plaatsen.

Als we *augmented reality* voor webapplicaties gaan vergelijken met mobile applicaties zien we dat beide voor- en nadelen hebben. Zo is web *augmented reality* gemakkelijker toegankelijk maar is het nog niet zo uitgebreid als *augmented reality* in een mobiele applicatie. [6]

3.3 Artikel 3: Web AR – AR.js

In de beginfase van *augmented reality* hadden de gebruikers zeer krachtige smartphones nodig of moest er een applicatie gedownload worden om *augmented reality* ervaringen te beleven. Door deze beperkingen was het moeilijk om *augmented reality* ervaringen aan een groot publiek aan te bieden. Alleen gebruikers met de nieuwste smartphones en de laatste versie van hun *operating system* konden *augmented reality* ervaren.

Tot nu toe was *augmented reality* ook nog niet ver ontwikkeld voor webapplicaties. *Augmented reality* was alleen beschikbaar op smartphones door een app te downloaden. Hierdoor konden de mensen die gebruik maken van een desktop geen *augmented reality* ervaren.

AR.js is een *open-source framework* dat *augmented reality* voor iedereen mogelijk maakt. AR.js is een *framework* gebaseerd op WebGL en WebRTC, waardoor *augmented reality* op elk platform beschikbaar wordt.

AR.js maakt gebruik van verschillende *augmented reality libraries* zoals ARToolkit en JSARToolkit. Deze *libraries* helpen om afbeeldingen en *markers* te plaatsen in de echte wereld. Het grootste voordeel van AR.js is dat er heel gemakkelijk een *augmented reality* ervaring gebouwd kan worden. Met slechts tien lijnen HTML code kan er al een eerste *augmented reality* applicatie gebouwd worden.

De HTML code maakt gebruik van A-frame. Dit is een web *framework* om *virtual reality* applicaties mee te bouwen voor alle platformen. In de code dient er een *marker* gedefinieerd te worden. Deze *marker* dient als herkenningspunt voor de camera. AR.js beheert het verplaatsen van de camera, het is A-frame dat het virtuele object kan aanpassen. [7]

3.4 Conclusie literatuurstudie

In het eerste artikel worden ARKit en ARCore met elkaar vergeleken. Volgens dit artikel zijn ARKit en ARCore heel gelijkaardig aan elkaar. Ze gebruiken beide ongeveer dezelfde technologieën om te bepalen waar de virtuele *content* getoond moet worden. Dit artikel besluit dat de keuze tussen deze twee platformen afhangt van het *operating system* (Android of iOS) waarvoor er ontwikkeld wordt.

Ook in het tweede artikel wordt kort aangehaald dat ARCore en ARKit dezelfde technologie gebruiken. Vervolgens worden deze *frameworks* vergeleken met web *augmented reality* in het algemeen. Ook zou volgens dit artikel *markerless augmented reality* in webapplicaties mogelijk zijn.

In het derde artikel wordt kort uitgelegd wat de voordelen van web *augmented reality* zijn en kort beschreven hoe AR.js werkt. Volgens dit artikel zou *markerless augmented reality* in AR.js niet mogelijk zijn.

In het onderzoek wordt er gekeken of ARKit en ARCore ook effectief veel gelijkenissen hebben en wordt er verder gezocht naar oplossingen voor *augmented reality* in webapplicaties.

4 Uitwerking onderzoek

4.1 Augmented reality

Augmented reality is eigenlijk de echte wereld verrijken door er digitale *content* aan toe te voegen. Dit gebeurt meestal door de camera van een smartphone of laptop. Binnen *augmented reality* zijn er verschillende technieken om te bepalen waar de virtuele *content* getoond moet worden. De twee belangrijkste manieren om de *content* te tonen zijn *marker based* en *markerless augmented reality*.

4.1.1 Marker based augmented reality

Marker based augmented reality maakt gebruik van een soort visuele code of *marker*. Dit is in de meeste gevallen een QR-code of een code verwerkt in een afbeelding. Deze *markers* dienen als een soort herkenningspunt voor de camera. Wanneer de camera een *marker* gedetecteerd heeft, wordt de virtuele *content* getoond op de *marker*.

Om deze *markers* te herkennen, wordt er gebruik gemaakt van image recognition. De *markers* dienen op voorhand kenbaar gemaakt te worden aan het systeem, zodat het systeem weet waarop de virtuele *content* getoond moet worden.

4.1.2 Markerless augmented reality

Bij *markerless augmented reality* worden er geen *markers* gebruikt. Dit wil zeggen dat de camera geen herkenningspunt heeft om de virtuele *content* op te tonen. *Markerless augmented reality* maakt gebruik van de verschillende meetapparatuur van het toestel. Dit om te bepalen waar de *content* getoond moet worden. Voorbeelden zijn: GPS, kompas en een snelheidsmeter. Deze apparatuur zitten tegenwoordig in de meeste smartphones. [8]

4.2 ARCore (Android)

ARCore is een *software development kit* ontwikkeld door Google. Met deze SDK kunnen *markerless augmented reality* applicaties gebouwd worden voor Android toestellen. Binnen ARCore wordt er gebruik gemaakt van verschillende API's die het mogelijk maken voor de smartphone om de wereld beter te begrijpen. Ook wordt er binnen ARCore gebruik gemaakt van verschillende technologieën om de virtuele *content* te integreren in de echte wereld.

Deze technologieën zijn:

- Motion tracking
- Environmental understanding
- Light estimation

4.2.1 Motion tracking

Als het toestel door de ruimte beweegt, voert ARCore een proces uit genaamd *concurrent odometry and mapping* (com). Dit proces dient ervoor om te weten waar het toestel is ten opzichte van de echte wereld. ARCore past dit toe door een aantal gemakkelijk te onderscheiden punten in het beeld van de camera te vinden. Deze worden ook wel *feature points* genoemd. Deze *feature points* worden gebruikt om de verplaatsing van het toestel te berekenen. Deze visuele informatie wordt gecombineerd met de *inertial measurement unit.* Vervolgens wordt er een schatting gemaakt om de positie en rotatie van de camera relatief ten opzichte van de wereld te bepalen.

4.2.2 Environmental understanding

ARCore zoekt naar een verzameling van *feature points* die zich op dezelfde horizontale of verticale vlakte bevinden, zoals tafels of muren. Binnen ARCore zijn deze oppervlaktes beschikbaar als *planes*. Ook kan ARCore de grens bepalen van een *plane*.

Vanaf het moment dat de applicatie weet waar de *planes* zich bevinden, kan er een *hit-cast* uitgevoerd worden. Met een *hit-cast* weet de applicatie op welke *plane* de user heeft gedrukt. Hierdoor kan de applicatie de objecten op de juiste *plane* plaatsen. Doordat de objecten op een *plane* getoond worden, lijkt het alsof ze op de grond of op een tafel staan.

4.2.3 Light estimation

ARCore kan informatie verzamelen rond de lichtinval van de omgeving. Ook kan het informatie rond de gemiddelde intensiteit en kleurcorrectie van een bepaald camerabeeld aan de applicatie bezorgen. Hierdoor is het mogelijk om het virtuele object met dezelfde lichtwaarden te tonen als in de echte wereld. Dit zorgt ervoor dat het beeld realistischer lijkt.

4.2.4 Augmented images

Ook binnen ARCore kunnen er *marker based augmented reality* toepassingen gebouwd worden. Binnen ARCore moeten de *markers* geen vast patroon hebben. Er kan gebruik gemaakt worden van 2D afbeeldingen zoals logo's en posters. Deze afbeeldingen worden ook wel *augmented images genoemd.*

ARCore is in staat om 20 afbeeldingen tegelijkertijd te herkennen, maar ARCore kan dezelfde afbeelding niet meerdere keren herkennen. Ook moeten de afbeeldingen minstens een oppervlakte hebben van 15cm bij 15cm en moeten ze vlak zijn.

4.2.5 Ondersteunde toestellen

Google gaat voor elke smartphone een proces doorlopen om te controleren of de smartphone krachtig genoeg is om ARCore te ondersteunen. Elke smartphone wordt gecontroleerd op verschillende aspecten. Zoals de kwaliteit van de camera, de bewegingssensoren en de architectuur van het toestel zelf.

Ook wordt er gekeken naar de kracht van de smartphone zelf. De processor van het toestel dient krachtig genoeg te zijn om een goede performantie te krijgen. Binnen het ruime assortiment van Android toestellen zijn er heel wat toestellen met verschillende hardware en software specificaties. Google probeert om constant ARCore te verbeteren zodat zoveel mogelijk modellen ondersteunt kunnen worden.

Om op dit moment ARCore te gebruiken op Android toestellen moet het toestel ten minste Android versie 7.0 beschikken. Ook is het mogelijk om ARCore te gebruiken op iOS toestellen. De iOS toestellen die ARCore ondersteunen, dienen ook ARKit te ondersteunen. Ook moeten ze tenminste over iOS versie 11.0 beschikken. Op de website van ARCore zelf is de volledige lijst van ondersteunde toestellen terug te vinden. Deze lijst wordt continue geüpdatet met nieuwe toestellen die ARCore ondersteunen. [9]

4.3 ARKit (Apple)

ARKit is de *software development kit* van Apple om *markerless augmented reality* applicaties te bouwen voor iOS toestellen. ARKit is de tegenhanger van Google zijn *augmented reality* platform, ARCore.

Ook ARKit maakt gebruik van een aantal gelijkaardige technieken om te bepalen waar de virtuele *content* getoond moet worden.

4.3.1 World tracking

Om de echte wereld in kaart te brengen, wordt er binnen ARKit gebruik gemaakt van een techniek genaamd *visual-inertial odometry*. Binnen dit proces wordt er informatie van de sensoren van het toestel, zoals de snelheidsmeter en de gyroscoop, gecombineerd met een *computer vison* analyse van de *scene*. ARKit kan ook verschillende punten in de *scene* herkennen en de verschillen waarnemen wanneer de camera beweegt.

Bij *world tracking* wordt ook de inhoud van de *scene* geanalyseerd. Hierdoor gaat ARKit de wereld beter begrijpen. Net zoals bij ARCore maakt ARKit gebruik van *hit-testing.* Door de *hit-testing* worden oppervlaktes die overeenkomen met een punt in de camera *scene* gedetecteerd. Ook is het mogelijk om met *plane detection* te werken. Net zoals in ARCore worden de positie en grootte van horizontale en verticale vlaktes gedetecteerd. Op deze vlaktes kan met behulp van een *hit-test* de virtuele *content* geplaatst worden.

Net zoals in ARCore wordt er binnen ARKit gebruik gemaakt van light estimation. ARKit gaat informatie verzamelen rond de lichtinval en gaat deze proberen toe te passen op het virtuele object.

4.3.2 World tracking beperkingen

Door *world tracking* zien *augmented reality* applicaties er tegenwoordig heel realistisch uit. Maar toch heeft *world tracking* een aantal beperkingen. Het hangt enorm veel af van de omgeving waarin *augmented reality* gebruikt wordt.

Bij *world tracking* wordt het beeld van de camera geanalyseerd. Om dit zo optimaal mogelijk te doen, moet het beeld zo duidelijk mogelijk zijn. Als er weinig details in het beeld zitten, zoals een witte muur, is het heel moeilijk om aan *world tracking* te doen.

Ook maakt *world tracking* gebruik van de sensoren van de smartphone zelf. Ideaal zou zijn als de camera een beetje beweegt zodat ARKit de wereld beter gaat begrijpen. Ook mag er niet overdreven worden bij het bewegen van de camera. Als de camera te snel beweegt, gaat ARKit de wereld minder goed in kaart kunnen brengen.

Wanneer er gebruik gemaakt wordt van *plane detection,* kan het zijn dat de positie en grootte niet volledig juist is. Als de gedetecteerde *plane* langer in beeld blijft, gaat ARKit de positie en grootte beter kunnen bepalen. [10]

4.3.3 Image detection

Ook binnen ARKit wordt *marker based augmented reality* ondersteund. Hier moeten de *markers* geen vast patroon hebben. Er kan gebruik gemaakt worden van 2D afbeeldingen, logo's of schilderijen. Elke *marker* moet gekend zijn door ARKit. Wanneer er een *marker* gedetecteerd wordt, kan ARKit de *content* op de afbeelding tonen.

Om gebruik te maken van *image detection* moet de iPhone wel minstens iOS versie 11.3 bezitten en moet binnen ARKit *image detection* expliciet worden aangezet.

4.4 Ondersteunde toestellen

ARKit is de *software development kit* van Apple. Apple ontwikkelt alleen maar voor Apple apparaten. Hierdoor is ARKit exclusief beschikbaar op iOS toestellen. De toestellen die ondersteund worden, moeten tenminste beschikken over een A9 Bionic processor. Elke iPhone vanaf de iPhone 6s bevat tenminste een A9 Bionic processor. Buiten iPhones worden ook alle iPad Pro modellen ondersteund.

Ook dient op de toestellen minsten iOS 11 geïnstalleerd te zijn. Om ARKit zelf te gebruiken, dient er bijkomend niets geïnstalleerd te worden. ARKit maakt gebruik van de hardware van het toestel. Namelijk de camerasensoren en de bewegingssensoren. [11]

4.5 ARKit 2

In 2018 werd door Apple ARKit 2 aangekondigd. Dit is de opvolger van ARKit. ARKit 2 is een onderdeel van iOS12 en brengt een aantal nieuwe functies met zich mee.

- Shared experiences (Multiplayer *augmented reality*)
- Persistent augmented reality
- Beelddetectie en -tracking
- USDZ

Shared experiences wil eigenlijk zeggen dat je vanaf nu ook een soort multiplayer *augmented reality* hebt. Je kan samen met iemand anders, die ook een eigen iPhone of iPad heeft, dezelfde augmented reality *content* zien en eventueel samen een spel spelen.

Persistent *augmented reality* zorgt ervoor dat de gebruiker kan verdergaan waar hij gebleven was. Was hij in *augmented reality* zijn huis aan het inrichten en wil hij een dag later verder werken, dan staan de objecten nog steeds op dezelfde plaats.

Met beelddetectie en -tracking is het voor ARKit mogelijk om 3D-objecten te detecteren. Samen met de detectie van horizontale en verticale vlakken moet dit voor een nog realistischer effect zorgen.

USDZ is het nieuwe bestandsformaat dat Apple samen heeft ontwikkeld met Pixar. Dit formaat moet het voor ontwerpers mogelijk maken om 3D-afbeeldingen te maken die dan in *augmented reality* tot leven komen. [12]

4.6 AR.js

In tegenstelling tot ARCore en ARKit is AR.js niet ontwikkeld door een grote speler op de markt, maar door een van de meest actieve gebruikers op GitHub. Namelijk Jerome Etienne, hij is de achtste van de meest actieve gebruikers op GitHub. Hij heeft AR.js ontwikkeld om *augmented reality* ervaringen te bouwen in webapplicaties. AR.js is een *open source* project. Dus alle code die hij ontwikkeld heeft is beschikbaar op GitHub. Binnen AR.js wordt er gewerkt met verschillende *frameworks*, namelijk three.js en ARToolKit.

4.6.1 Three.js

Three.js is een 3D JavaScript *framework* dat gebruik maakt van WebGL om gemakkelijk 3D objecten te tonen in een web browser.

4.6.2 ARToolKit

AR.js maakt gebruik van ARToolKit, hierdoor is AR.js een *marker based framework*. ARToolKit is een *open source* computer *tracking library* om *augmented reality* applicaties mee te bouwen.

ARToolKit kan ervoor zorgen dat virtuele *content* boven op een afbeelding in de echte wereld geplaatst wordt. ARToolKit gaat via de camera de echte wereld analyseren. In elke frame wordt er gezocht naar *markers*. Wanneer een *marker* gevonden wordt, gaat ARToolKit een wiskundige berekening maken om te positie van de camera te bepalen ten opzichte van de *marker*. Nadien wordt er gecontroleerd welke *marker* in beeld is en wordt deze vergeleken met de *markers* in het geheugen. Wanneer de juiste *marker* gevonden is, wordt er bepaald welke virtuele *content* erop getoond moet worden. Ook wordt ervoor gezorgd dat je positie en oriëntatie van de virtuele *content* overeen komt met die van de *marker*. Tenslotte kan de gebruiker via zijn smartphone de virtuele *content* bekijken. [13]

4.6.3 Ondersteunde toestellen

AR.js is een *framework* dat volledig gefocust is op webapplicaties. Hierdoor is er geen installatie vereist. Er wordt ook gebruik gemaakt van verschillende webstandaarden, namelijk WebRTC en WebGL. Doordat deze standaarden gebruikt worden, is AR.js beschikbaar op de meeste browsers en smartphones.

WebRTC is een *open source* protocol dat ervoor zorgt dat webapplicaties gebouwd kunnen worden voor spraak- en videogesprekken en p2p-bestandsuitwisseling zonder gebruik te maken van extra plugins.

WebGL is een JavaScript API dat ervoor zorgt dat er 2D en 3D objecten getoond kunnen worden in een web browser zonder gebruik te maken van extra plugins. [3]

4.7 Benodigdheden markerless augmented reality

Zoals op de vorige pagina's aangehaald, wordt er bij *markerless augmented reality* gebruik gemaakt van allerlei verschillende manieren om te bepalen waar de virtuele *content* getoond moet worden. Binnen ARCore en ARKit wordt de wereld in kaart gebracht door verschillende technieken zoals *world tracking* te gebruiken. Beide *frameworks* proberen om vlaktes in de echte wereld te herkennen. Om nadien op deze vlaktes de virtuele *content* te plaatsen. Om de vlaktes te bepalen, wordt er gebruik gemaakt van allerlei verschillende sensoren. Voorbeelden hiervan zijn de snelheidsmeter en de gyroscoop. Doordat de gebruiker kan bewegen en het toestel kan draaien, zou het kunnen dat de virtuele *content* anders getoond moet worden om toch nog realistisch over te komen. Door gebruik te maken van deze sensoren kan het toestel bepalen hoe het zich bevindt ten opzichte van de echte wereld. Het kan hierdoor de virtuele *content* terug realistischer plaatsen.

Deze technieken worden toegepast binnen *markerless augmented reality frameworks*. Binnen *marker based augmented reality frameworks*, zoals AR.js, wordt er geen gebruik gemaakt van deze sensoren of van een principe om de wereld in kaart te brengen.

AR.js steunt op andere *frameworks* en technologieën. Binnen deze *frameworks* wordt er geen gebruik gemaakt van sensoren van de toestellen of van technieken zoals *world tracking*. AR.js maakt gebruik van ARToolKit. Zoals eerder aangehaald gaat ARToolKit via de camera de wereld analyseren en gaat het op zoek naar *markers.* Wanneer de *markers* gedetecteerd worden, gaat hierop de virtuele *content* getoond worden.

Doordat AR.js gebruik maakt van ARToolKit is het niet mogelijk om binnen AR.js een *markerless augmented reality* applicatie te bouwen. Maar hieruit kan er nog niet besloten worden dat *markerless augmented reality* niet mogelijk is in een webapplicatie.

Er bestaat ook een redelijk nieuw *framework* voor webapplicaties, genaamd 8th Wall. Dit *framework* maakt gebruik van gelijkaardige technieken als ARCore en ARKit om *markerless augmented reality* mogelijk te maken.

4.8 8th Wall

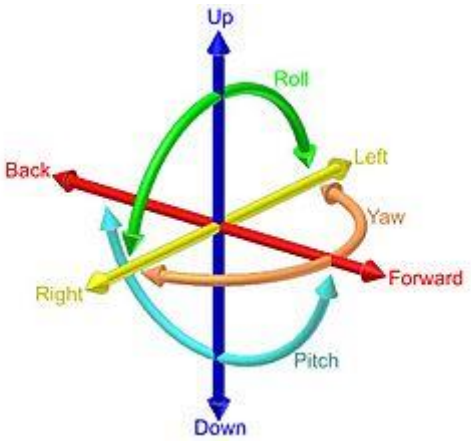
8th Wall is een *augmented reality framework* voor webapplicaties. 8th Wall zorgt ervoor dat *augmented reality* geïntegreerd kan worden op elke website. Om dit mogelijk te maken, steunt 8th Wall op standaarden zoals JavaScript en WebGL. Ook wordt er binnen 8th Wall gebruik gemaakt van SLAM. Dit staat voor *Simultaneous Localization and Mapping.* SLAM wil eigenlijk zeggen dat het toestel een kaart probeert op te stellen van de omgeving en tegelijkertijd zichzelf op deze kaart probeert terug te vinden.

Ook binnen 8th Wall worden er verschillende manieren gebruikt om te bepalen waar de virtuele *content* getoond moet worden.

4.8.1 6DoF positional tracking

6DoF staat voor 6 *Degrees of Freedom*. Dit verwijst naar het aantal manieren dat een object door de ruimte kan bewegen. 8th Wall is in staat om de positie van de smartphone op zes manieren te volgen.

Figuur 6 visualiseert de verschillende *Degrees of Freedom*.



Figuur 6: 6 Degrees of Freedom

4.8.2 Surface estimation

Door Surface Estimation is 8th Wall in staat om de grond en vlakke oppervlaktes te herkennen. Wanneer de vlaktes herkend worden, kan 8th Wall ervoor zorgen dat de virtuele *content* op deze vlaktes getoond worden. Op deze manier kunnen de objecten realistischer geplaatst worden.

4.8.3 World points

Buiten vlaktes kunnen ook punten in de echte wereld herkend worden. Hierdoor is het mogelijk om deze punten te gebruiken voor visualisatie.

4.8.4 Hit-tests

Door het detecteren van oppervlaktes en punten in de echte wereld zijn er genoeg plaatsen waar de objecten op getoond kunnen worden. *Hit-tests* maakt het mogelijk om interactie te hebben met deze oppervlaktes en punten. Wanneer de gebruiker op een oppervlakte klikt, wordt er een hit-test uitgevoerd om nadien op deze plaats de virtuele objecten te plaatsen.

4.8.5 Light estimation

Net zoals ARCore en ARKit kan 8th Wall ervoor zorgen dat de objecten er realistischer uitzien. Dit door de lichtinval van de echte wereld te analyseren en nadien dezelfde lichtinval op de virtuele objecten toe te passen.

4.8.6 Ondersteunde toestellen

Aangezien 8th Wall een web *framework* is, kunnen de meeste smartphones met een camera en een web browser 8th Wall applicaties draaien. [14]

4.9 Prototype 8th Wall

In het prototype wordt er een kleine voorbeeldapplicatie gemaakt dat aantoont dat *markerless augmented reality* in een webapplicatie mogelijk is. Er wordt in dit prototype een applicatie gemaakt waarin er doormiddel van een *raycast* een kubus in de ruimte wordt geplaatst.

De eerste stap is om een account aan te maken op de website van 8th Wall zelf. Nadat een account is aangemaakt, wordt de *app key* weergegeven op het *console* scherm.

Nadat de *key* gegenereerd is, kan er gestart worden met het bouwen van de applicatie. Om gebruik te maken van 8th Wall moeten er een aantal *script tags* voorzien worden.

Figuur 7 toont de verschillende *script tags*. De eerste twee dienen voor het importeren van een aangepaste versie van a-frame. De derde dient voor het plaatsen van de *app key.*

```
<script src="//cdn.8thwall.com/web/aframe/8frame-0.8.2.min.js"></script>
<script src="//cdn.8thwall.com/web/aframe/aframe-animation-component-5.1.2.min.js"></script>
<script async src="//apps.8thwall.com/xrweb?appKey=bmcjNQkypwst3UAo8kolG1lHjGGmTyoUOeYikaLUzLc1BTocWxweTTAhvOsPipuPmCJiMF"></script>
```

Figuur 7: Script tags

Nadien wordt er aan het *body* element van de webpagina een aantal a-frame elementen toegevoegd. Het eerste element is het 'a-scene' element. Dit element is het globale element van a-frame. Hierin worden alle objecten geplaatst. Ook heeft dit element een aantal attributen. De meeste zijn niet zo belangrijk en dienen vooral voor error afhandeling en om extra laadschermen te voorzien. De twee belangrijkste zijn 'xrweb' en 'tap-place'. De eerste dient om gebruik te maken van SLAM. Het tweede attribuut wordt dadelijk via JavaScript voorzien. Deze dient voor het plaatsen van het virtuele object op de plaats waar de gebruiker heeft geklikt.

Binnen het ‘a-scene’ element zijn nog een aantal a-frame elementen voorzien. Het eerste element is een ‘a-camera’. Deze camera bepaalt wat de gebruiker te zien krijgt. In dit voorbeeld bevat de camera een attribuut ‘raycaster’. Dit attribuut heeft de waarde ‘objects: .cantap’. Dit wil zeggen dat er een *raycast* wordt uitgevoerd wanneer er op een object met het id ‘cantap’ wordt geklikt.

Vervolgens is er nog een ‘a-entity’ element. Dit is een doorzichtige vlakte die we laten overeenkomen met de grond. Ook heeft dit element het id ‘cantap’, zodat we op de grond kunnen klikken om nadien de objecten op de grond te plaatsen.

Als laatste zijn er nog twee elementen. Deze dienen slechts voor het bepalen van de juiste lichtinval.

In figuur 8 wordt het ‘body’ element weergegeven.

```
<a-scene xrweb tap-place xrextras-almost-there xrextras-loading xrextras-runtime-error>
    <a-camera position="0 8 0" raycaster="objects: .cantap" cursor="
      fuse: false;
      rayOrigin: mouse;">
    </a-camera>

    <a-entity id="ground" class="cantap" geometry="primitive: box"
        material="color: #ffffff; transparent: true; opacity: 0.0" scale="1000 2 1000" position="0 -1 0">
    </a-entity>

    <a-entity light="type: directional;
          intensity: 0.8;" position="1 4.3 2.5">
    </a-entity>
    <a-light type="ambient" intensity="1"></a-light>
</a-scene>
```

Figuur 8: Inhoud body element

Tenslotte moet er nog voor gezorgd worden dat er een kubus tevoorschijn komt wanneer de gebruiker op het scherm klikt. De kubus moet tevoorschijn komen op de plaats waar de gebruiker geklikt heeft.

Als eerste dient het 'tap-place' attribuut geregistreerd te worden bij a-frame. Vervolgens wordt het object met het id 'ground' opgehaald. Op dit object wordt een *click event* gekoppeld. Binnen dit *event* dient er een nieuwe kubus aangemaakt te worden en wordt de positie van de kubus ingesteld. De positie moet overeenkomen met het punt waar de gebruiker geklikt heeft. Deze positie kan achterhaald worden door aan het *click event* het geklikte punt te vragen. Vervolgens kan er op de kubus dezelfde positie ingesteld worden. Tenslotte moet de kubus nog aan de 'a-scene' toegevoegd worden. Wanneer de kubus is toegevoegd, wordt deze zichtbaar op het scherm.

Als deze code is uitgevoerd, wordt er een kubus getoond op de plaats waar de gebruiker geklikt heeft. De gebruiker kan zo vaak klikken als hij wil en er gaat telkens een nieuwe kubus tevoorschijn komen.

Figuur 9 toont de JavaScript code om een nieuwe kubus te plaatsen op de plaats waar de gebruiker geklikt heeft. [15]

```
AFRAME.registerComponent('tap-place', {
    init: function () {
        const ground = document.getElementById('ground')
        ground.addEventListener('click', event => {
            const newElement = document.createElement('a-entity')
            const touchPoint = event.detail.intersection.point

            newElement.setAttribute('position', touchPoint)
            newElement.setAttribute('geometry', {
                primitive: 'box',
            });
            this.el.sceneEl.appendChild(newElement)
        })
    }
})
```

Figuur 9: JavaScript code

Conclusie

In het onderzoek zijn er drie verschillende *frameworks* met elkaar vergeleken. Namelijk ARCore, ARKit en AR.js. Door deze *frameworks* te onderzoeken, werd er gekeken welke technieken er gebruikt worden om de virtuele objecten in de echte wereld te plaatsen. Nadien werd er gekeken of er een *framework* voor een webapplicatie bestaat waar dezelfde technieken gebruikt worden. Uit deze vergelijking kunnen een aantal conclusies gehaald worden.

Ten eerste, ARCore en ARKit zijn heel gelijkaardig aan elkaar. Ze gebruiken beide ongeveer dezelfde technieken om de virtuele objecten in de echte wereld te plaatsen. Beide platformen gaan de echte wereld proberen te begrijpen en gaan vlaktes detecteren om de objecten zo realistisch mogelijk in de wereld te plaatsen.

Als tweede is er gekeken naar AR.js. Dit om na te gaan of de technieken van ARCore en ARKit ook toegepast worden in AR.js. Hieruit is gebleken dat AR.js deze technieken niet toepast. Hierdoor is er enkel *marker based augmented reality* mogelijk binnen AR.js.

Tenslotte is er gekeken of er andere web *frameworks* bestaan die wel gelijkaardige technieken toepassen. Hieruit bleek dat er een *framework* bestaat dat *markerless augmented reality* in een webapplicatie mogelijk maakt. Dit *framework* is 8th Wall.

Binnen het 8th Wall *framework* worden dezelfde technieken gebruikt als binnen ARCore en ARKit. 8th Wall maakt gebruik van *Surface Estimation,* het herkennen van *world points* en het bepalen van de positie van de smartphone in de ruimte. Verder kan het ook *hit tests* uitvoeren om een virtueel object op een bepaalde positie te plaatsen.

Om aan te tonen dat *markerless augmented reality* in 8th Wall ook effectief werkt, is er een prototype gebouwd. In dit prototype wordt er een kubus getoond op de plaats waar de gebruiker klikt. Hieruit kan geconcludeerd worden dat *markerless augmented reality* in een webapplicatie mogelijk is.

Als deze resultaten vergeleken worden met de conclusie van de literatuurstudie, blijkt dat deze volledig overeenkomen. Verder bewijst de literatuurstudie dat ARCore en ARKit heel gelijkaardig aan elkaar zijn. Het onderzoek heeft uitgewezen dat dit ook effectief geval is. Beide platformen zijn gebaseerd op dezelfde technieken om de virtuele objecten te tonen in de echte wereld.

Als slot toont de literatuurstudie ook dat *markerless augmented reality* in een webapplicatie mogelijk is. Alleen niet met het AR.js *framework.* Door AR.js te onderzoeken, bleek ook dit te kloppen. Met AR.js kunnen er alleen maar *marker based* toepassingen gebouwd worden. Ook bestaan er *frameworks* waarin *markerless augmented reality* applicaties gebouwd kunnen worden.

Uit het onderzoek blijkt dat *markerless augmented reality* in een webapplicatie mogelijk is. Dit door middel van het 8th Wall *framework*. Maar hieruit kan nog niet besloten worden dat *augmented reality* in een webapplicatie al even ver staat als *augmented reality* in een mobiele applicatie.

Tijdens het opstellen van het prototype viel het op dat het 8th Wall project niet zo soepel voelt als andere *augmented reality* toepassingen. Dit zou kunnen betekenen dat *augmented reality* in een webapplicatie nog niet zo ver ontwikkeld is als *augmented reality* in een mobiele applicatie. Een logisch volgend onderzoek is dat de verschillende *frameworks* met elkaar worden vergeleken op basis van performantie. Hieruit kan het *framework* gehaald worden dat de beste gebruikservaring heeft.

Het belangrijkste wat er kan besloten worden uit dit onderzoek is dat *markerless augmented reality* in een webapplicatie mogelijk is. Hierdoor is *markerless augmented reality* op heel wat platformen beschikbaar. Elk *framework* gebruikt gelijkaardige technieken om de virtuele objecten in de wereld te plaatsen. Het *framework* dat gebruikt wordt, hangt af van het platform waarvoor er ontwikkeld wordt.

5 Bibliografie

[1] Flaviocopes, „Flaviocopes,” FlavioCopes, 11 november 2018. [Online]. Available: https://flaviocopes.com/single-page-application/. [Geopend 22 april 2019].

[2] Contentful, „Contentful,” Contentful, [Online]. Available: https://www.contentful.com/faq/about-contentful/. [Geopend 5 maart 2019].

[3] J. Etienne, „github,” github, [Online]. Available: https://github.com/jeromeetienne/AR.js/blob/master/README.md. [Geopend 05 maart 2019].

[4] J. Etienne, „aframe,” aframe, 11 juli 2017. [Online]. Available: https://aframe.io/blog/arjs/. [Geopend 30 april 2019].

[5] A. Blum, „iFlexion,” 09 maart 2018. [Online]. Available: https://www.iflexion.com/blog/arcore-vs-arkit-better-building-augmented-reality-apps. [Geopend 05 april 2019].

[6] S. motte, „blog.vertebrae,” 01 juni 2018. [Online]. Available: https://blog.vertebrae.com/app-vs-web-ar-advertising. [Geopend 05 april 2019].

[7] T. Bhatt, „dzone.com,” 11 juli 2018. [Online]. Available: https://dzone.com/articles/the-fundamentals-of-developing-augmented-reality-. [Geopend 05 april 2019].

[8] realitytechnologies, „realitytechnologies,” realitytechnologies, [Online]. Available: https://www.realitytechnologies.com/augmented-reality/. [Geopend 08 maart 2019].

[9] Google, „Google,” Google, 28 februari 2019. [Online]. Available: https://developers.google.com/ar/discover/. [Geopend 08 maart 2019].

[10] Apple, „Apple,” Apple, [Online]. Available: https://developer.apple.com/documentation/arkit/understanding_world_tracking_in_arkit. [Geopend 29 maart 2019].

[11] S. Axon, „arstechnica,” 16 juni 2018. [Online]. Available: https://arstechnica.com/gadgets/2018/06/arkit-2-why-apple-keeps-pushing-ar-and-how-it-works-in-ios-12/. [Geopend 30 maart 2019].

[12] Apple, „Apple,” Apple, 4 juni 2018. [Online]. Available: https://www.apple.com/benl/newsroom/2018/06/apple-unveils-arkit-2/. [Geopend 2 april 2019].

[13] Hitl, „hitl.washington,” hitl.washington, [Online]. Available: http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm. [Geopend 10 mei 2019].

[14] 8ThWall, „8ThWall,” 8ThWall, [Online]. Available: https://www.8thwall.com/faq.html. [Geopend 10 mei 2019].

[15] 8ThWall, „Github,” 8ThWall, [Online]. Available: https://github.com/8thwall/web/tree/master/examples/aframe. [Geopend 10 mei 2019].