# Afstudeerscriptie

Diract-it

Sieger Falkena

500602792

Inhoud

[Afstudeerscriptie 1](#_Toc450681687)

[Inleiding 5](#_Toc450681688)

[Diract-it 5](#_Toc450681689)

[Concentrator 5](#_Toc450681690)

[WMS 5](#_Toc450681691)

[Probleemstelling 6](#_Toc450681692)

[Service Oriented Architecture 6](#_Toc450681693)

[Toegankelijk voor mobiele applicaties 6](#_Toc450681694)

[Bruikbaar voor meerdere teams 6](#_Toc450681695)

[Onderzoeksmethoden 6](#_Toc450681696)

[Doelstelling 6](#_Toc450681697)

[Wat is een goede front-end opzet voor diract-it? 7](#_Toc450681698)

[Wie zijn de gebruikers en waar hechten zij waarde aan? 7](#_Toc450681699)

[PACMAD 7](#_Toc450681700)

[Deelconclusie 8](#_Toc450681701)

[Wat zijn mobiele limitatie en hoe kunnen deze worden gecompenseerd? 9](#_Toc450681702)

[Limitaties 9](#_Toc450681703)

[Visueel 9](#_Toc450681704)

[Invoer 9](#_Toc450681705)

[Netwerk 10](#_Toc450681706)

[Hardware 11](#_Toc450681707)

[Voordelen 11](#_Toc450681708)

[Touchscreen 11](#_Toc450681709)

[Microfoon 11](#_Toc450681710)

[Camera 11](#_Toc450681711)

[Tactiel / locatie gebaseerde informatie 12](#_Toc450681712)

[Deelconclusie 12](#_Toc450681713)

[Hoe kunnen we de webapplicatie structureren om de mobiele toegankelijkheid te vergroten? 12](#_Toc450681714)

[Business rules 13](#_Toc450681715)

[Export- en importregels 13](#_Toc450681716)

[Producten en attributen 13](#_Toc450681717)

[Kanalen 13](#_Toc450681718)

[Details abstraheren 13](#_Toc450681719)

[Wat is nodeJs en waarom gebruikt iedereen het? 13](#_Toc450681720)

[Node.JS 13](#_Toc450681721)

[Dependency management 13](#_Toc450681722)

[Build-management 14](#_Toc450681723)

[Testen en ontwikkelondersteuning 14](#_Toc450681724)

[Integratie met visual studio 2015 14](#_Toc450681725)

[Conclusie 14](#_Toc450681726)

[Welke javascript frameworks sluiten het beste aan bij de vereisten? 15](#_Toc450681727)

[AngularJS 16](#_Toc450681728)

[Voordelen 16](#_Toc450681729)

[Nadelen 16](#_Toc450681730)

[Sencha ExtJS 17](#_Toc450681731)

[Voordelen 17](#_Toc450681732)

[Nadelen 17](#_Toc450681733)

[ReactJS 18](#_Toc450681734)

[Voordelen 18](#_Toc450681735)

[Neutraal 18](#_Toc450681736)

[Nadelen 18](#_Toc450681737)

[Deelconclusie 18](#_Toc450681738)

[Welke CSS framework past het beste bij concentrator? 19](#_Toc450681739)

[Waarom gebruiken ontwikkelaars een front-end framework? 19](#_Toc450681740)

[Selectie van frameworks 19](#_Toc450681741)

[Material Design 19](#_Toc450681742)

[Voordelen 19](#_Toc450681743)

[Speciaal voor mobiele webapplicaties 19](#_Toc450681744)

[Nadelen 19](#_Toc450681745)

[Bootstrap 19](#_Toc450681746)

[Voordelen 20](#_Toc450681747)

[Nadelen 20](#_Toc450681748)

[Metro-ui 20](#_Toc450681749)

[Zurb foundation 20](#_Toc450681750)

[Deelconclusie 20](#_Toc450681751)

[Hoe werkt een service oriented architecture? 21](#_Toc450681752)

[REST 21](#_Toc450681753)

[SOAP 22](#_Toc450681754)

[Backend webserver 22](#_Toc450681755)

[Deelconclusie 23](#_Toc450681756)

[Hoe werkt de authenticatie van het systeem? 24](#_Toc450681757)

[Conclusie 24](#_Toc450681758)

[Verklarende woordenlijst 24](#_Toc450681759)

[Bibliografie 24](#_Toc450681760)

# Inleiding

Dit onderzoeksrapport is geschreven naar aanleiding van mijn afstudeeronderzoek bij Diract-it.

## Diract-it

Diract-it is een software huis dat actief is in de branche van detailhandel. De producten van diract-it ondersteunen bijna elk onderdeel gerelateerd aan de inkoop, distributie en verkoop van producten. Diract heeft daarmee een webshop groep, een warehouse mangement systeem, een business-intelligence afdeling en een product information management systeem.

Diract-it was dicht gekoppeld aan de BAS-groep, een verkoopketen gericht op consumenten-electronica. De BAS-groep was de primaire afnemer van de systemen van Diract, tot het punt dat beide bedrijven in hetzelfde gebouw gehuisvest waren. Diract heeft meerdere innovatieve systemen aan BAS geleverd, onder andere het aanbieden van een mobiele kassa en omni-channel verkoop.

Binnen de organisatie van de BAS groep is er nu een ruimte om zich op een bredere klantenbasis te oriënteren. Meerdere producten zijn toe aan een nieuwe versie, en er is een druk om met opgedane ervaring een nieuw systeem te bouwen.

## Concentrator

De concentrator is een product dat gebruikt word voor het importeren en exporteren van productinformatie naar fysieke winkels en webshops.

Dit importeren en exporteren gaat aan de hand van business-rules die voor de klant worden geprogrammeerd.

Producten bevatten veel informatie zoals maat, naam en kleur. Deze informatie is beschikbaar in verschillende contexten zoals taal, import- of exportlocatie.

## WMS

WMS is een ander product van Diract-it. Het Warehouse Management Systeem houdt inventaris van een warehouse bij, en komt met verschillende functionaliteiten.

# Probleemstelling

Voor de volgende versie van concentrator willen de ontwikkelaars de functionaliteit van de webapplicatie ook op mobiele appraten goed aanbieden. In de huidige versie is de webapplicatie maar matig toegankelijk voor mobiele gebruiker.

De huidige versie (concentrator 2) was al een single-page applicatie, en dat wil de opdrachtgever graag zo houden.

## Service Oriented Architecture

Binnen Diract is er een wens alle producten makkelijker aan elkaar te koppelen. Dit verloopt tot nu toe een beetje moeilijk. Bedrijfsbreed is er besloten alle nieuwe producten en productversies een servicelaag te geven die beide door eindgebruikers als andere systemen als aanspreekpunt kan worden gebruikt.

## Toegankelijk voor mobiele applicaties

De webapplicatie moet bruikbaar zijn op kleinere apparaten en displays. De concentrator is daarnaast een relatief complexe applicatie die veel functionaliteit bevat en veel interactie van de gebruiker vereist. Relatief gezien is het een complexere applicatie dan bijvoorbeeld een webshop.

## Bruikbaar voor meerdere teams

De wens vanuit Diract is dat niet-product-specifieke functionaliteit bedrijfsbreed kan worden gebruikt in meerdere producten. Hiermee kunnen capaciteiten flexibeler worden ingezet zonder lange inwerktijd.

## Onderzoeksmethoden

Er is gekozen voor een combinatie van deskresearch en praktische applicatie door middel van een proof-of-concept.

# Doelstelling

Doel is het onderzoeken van opties voor de nieuwe Diract frontend.

De werking wordt aangetoond door middel van een proof-of-concept met bijbehorende documentatie.

Er wordt specifiek niet gezocht naar een nieuw platform voor de applicatielogica.

# Wat is een goede front-end opzet voor diract-it?

# Wie zijn de gebruikers en waar hechten zij waarde aan?

## PACMAD

People at the center of mobile application development

Mobiele gebruiksvriendelijkheid is anders dan gebruiksvriendelijkheid voor desktopsystemen, en veel problemen worden vaak over het hoofd gezien. Harrison et. Al. Hebben een metaonderzoek gedaan naar andere onderzoeken die eigenschappen voor mobiele gebruiksvriendelijkheid classificeerden. (Harrison, et al., 2013)

Volgens Harrison zijn de volgende factoren van invloed op mobiele gebruiksvriendelijkheid.

* De gebruiker
* De taken die de gebruiker wil ondernemen
* De context in welke de gebruiker deze taken wil uitvoeren

In het geval van Diract-IT is de gebruiker een *product manager* of *order manager*. Zij zijn respectievelijk verantwoordelijk voor het bijhouden van productinformatie en order afhandeling. Van beide kan een basiskunde van computers worden verwacht. Tussen *product managers* en *order managers* is een overlap in het takenpakket, en de functies worden in sommige organisaties door dezelfde werknemer gedaan.

De taken voor beide gebruikers ligt in het onderhouden en beheren van het informatiesysteem, ondersteund door professionals van Diract-IT. Voorbeelden zijn het inzien en aanpassen van productinformatie, het activeren en deactiveren van producten en het te hulp schieten van mislukte orders.

De nieuwe concentrator kent verschillende contexten; de applicatie kan door de huidige opzet momenteel alleen gemakkelijk met een desktop worden gebruikt. De organisatie wil dat de concentrator ook dynamisch op de werkvloer kan worden gebruikt met een tablet of smartphone.

Een ander context is dat Diract-it als bedrijf internationaal wil kunnen opereren en services aan wil bieden. De huidige webapplicatie is Engelstalig, maar het is niet vanzelfsprekend dat buitenlandse gebruikers de Engelse taal machtig zijn.

In het PACMAD model hebben deze contexten invloed op de volgende attributen;

* Effectiviteit
  + Het kunnen uitvoeren van taken. Dit heeft veel te maken met het accuraat implementeren van user stories.
* Efficiëntie
  + De snelheid en accuraatheid van taken
* Bevrediging
  + De subjectieve gebruikerservaring
* Leerbaarheid
  + De snelheid waarin de gebruiker leert zijn taken in het systeem te vervullen.
* Onthoudbaarheid
  + De snelheid waarin een gebruiker vergeet hoe hij zijn taken binnen het systeem kan doen
* Fouten
  + De fouten die de gebruiker veroorzaakt, en de bugs die de ontwikkelaar introduceert. Fouten zijn onvermijdelijk, maar de mate waarin feedback wordt gegeven en de snelheid waarmee bugs worden verholpen zijn cruciaal voor een goed gebruikersgemak
* Cognitieve Ballast
  + De concentratie die de applicatie vereist.

De eerste drie (Effectiviteit, Efficiëntie en bevrediging ) zijn het meest onderzocht en het meest van invloed op gebruikerservaring. Het gaat hier vooral om de tijd die een gebruiker nodig heeft om een taak succesvol af te ronden.

Prefetching is een van de technieken die de efficientie ten goede komt. Door veelgebruikte functionaliteit automatisch op de achtergrond te laden, is de laadtijd achteraf nul. Dit vinden gebruikers fijn.

De andere attributen; (leerbaarheid, onthoudbaarheid, fouten en cognitieve ballast) zijn in de meeste studies alleen subjectief gemeten. Ondanks het gebrek aan informatie over deze attributen is het wel mogelijk om hier rekening mee te houden.

* Een gebruikershandleiding komt de leerbaarheid van een applicatie ten goede.
* Goede feedback en confirmatie van handelingen komt de foutgevoeligheid van de applicatie ten goede.

Het meest subjectieve punt van aandacht is de cognitieve ballast van de applicatie. Deze is alleen uit te drukken in een context van multitasking – een context die eigenlijk niet van toepassing is voor de gebruiker van de applicatie.

## Deelconclusie

In tegenstelling tot de frontend van de huidige concentrator (2.5) zijn er nieuwe contexten aanwezig. De applicatie toegankelijk zijn voor gebruikers die in het buitenland opereren. De nieuwe webapplicatie moet ook op een mobiele applicatie goed mobiel toegankelijk worden. Deze mobiele context wordt in het volgende hoofdstuk beter behandeld.

In het proof-of-concept is er een veilige aanname gedaan dat er in ieder geval CRUD functionaliteit gaat worden gebruikt - Simpele taken zoals in ieder geval het lezen en aanmaken van producten. De rest van de taken zijn breed, verschillen per klant en liggen daarmee buiten de haalbaarheid van het proof-of-concept.

Het zoeken en filteren op producten is daarmee een praktische eis, omdat het vinden en aanpassen van sets producten een kerntaak van product-managers is.

Prefetching

Foutafhandeling kan met validatie worden bewerkstelligd. Ongeacht gebruiker- of systeemfout moet het systeem duidelijk aangeven wat er verkeerd is gegaan en liefst waarom.

Voor de leerbaarheid en onthoudbaarheid van de applicatie is het nodig een goede handleiding aan te bieden, aan de hand van de taken van de gebruiker. Deze kan digitaal worden verstrekt in de webapplicatie zelf en/of via documentatie en trainingen.

# Wat zijn mobiele limitatie en hoe kunnen deze worden gecompenseerd?

De limitaties van mobiele apparaten tegenover het traditionele desktop met toetsenbord en muis zijn significant, en moeten worden onderzocht.

## Limitaties

### Visueel

Visueel hebben mobiele apparaten een over het algemeen kleiner beeldscherm en een hogere resolutie. Dit maakt dat elementen (knoppen en tekst) die als goed bruikbaar op desktop worden ervaren, als te klein worden ervaren op mobile. Vice-versa kunnen tekst en knoppen voor mobile als te groot worden ervaren in een desktop context.

De oplossing is het gebruik van Responsive Web Design. RWD geeft de ontwikkelaar de mogelijkheid elementen in verschillende maten aan te bieden, afhankelijk van de afmetingen van het beeldscherm. Het responsive framework maakt dan de inhoud.

Er zijn meerdere frameworks waarmee RWD gemakkelijk te implementeren is. Voornaamste spelers zijn Twitter bootstrap, Foundation, Metro-UI en Material Design.

Daarnaast zijn er elementen die weinig waarde toevoegen

### Invoer

Mobiele apparaten beschikken over een touchscreen als primair invoerapparaat. Het touchscreen functioneert beide als muis en toetsenbord. Dit hoofdstuk behandeld alleen de zwaktes van het touchscreen t.o.v. muis en toetsenbord; de sterktes worden in een volgend hoofdstuk behandeld.

|  |  |  |
| --- | --- | --- |
|  | **Mouse** | **Fingers** |
| **Precision** | High | Low |
| **Number of points specified** | 1 | usually 1 2–3 with multi-touch |
| **Number of controls** | 3: left/right button, scroll wheel | 1 |
| **Homing time?** | Yes | No |
| **Signal states** | Hover, mouse-down, mouse-up | Finger-down, finger-up |
| **Accelerated movements** | Yes | No |
| **Suitable for use with big desktop monitors (30-inch or more)** | Yes, because of acceleration | No: arm fatigue |
| **Visible pointer/cursor** | Yes | No |
| **Obscures view of screen** | No, thus allowing for continuous visual feedback | Yes |
| **Suitable for mobile** | No | Yes: nothing extra to carry around |
| **Ease of learning** | Fairly easy | Virtually no learning time |
| **Direct engagement with screen and "fun" to use** | No: an indirect pointing device | Yes |
| **Accessibility support** | Yes | No |

Bron: <https://www.nngroup.com/articles/mouse-vs-fingers-input-device/>, 10-4-2012

**Touchscreen als muis**

Het touchscreen doet niet onder de muis, maar is wel sterk anders in gebruik. Vingers zijn minder precies dan een muiscursor, en nemen daarnaast het scherm in beslag. Daartegen is de touchscreen wel intuïtief in gebruik, omdat ze de meest natuurlijke impuls weet te gebruiken.

**Touchscreen als toetsenbord**

Een mobiele telefoon is minder geschikt voor de invoer van lange stukken tekst, voornamelijk omdat de tactiele sensatie van een toetsendruk ontbreekt en omdat de knoppen over het algemeen kleiner zijn dan die van een fysiek toetsenbord. Beginnende touchscreen typisten halen een gemiddelde van 10 wpm op een (klein) touchscreen, vergeleken met 41,2 wpm voor een normaal toetsenbord.

Typefouten zijn daarnaast makkelijk te maken op een touchscreen, voornamelijk vanwege de imprecisie van de vinger.

De oplossing voor slechte tekstinvoer is het automatisch aanvullen van gedeeltelijke informatie. Door de gebruiker automatisch aanvullingsopties te geven hoeven er minder toetsen te worden aangeslagen.

De tweede oplossing is het gebruik van de sterke eigenschappen van de telefoon, en wordt in volgende hoofdstukken behandeld.

### Netwerk

Een snelle internetconnectie is voor mobiele apparaten minder vanzelfsprekend dan een desktop. Telefoons zijn afhankelijk van snelle draadloze netwerken die niet altijd en overal beschikbaar zijn. Zelfs wifi kan overbezet worden door de bandbreedte van de radiofrequenties.

Een lage bandbreedte in combinatie met grote applicatiegrootte kan tot een trage gebruikerservaring leiden. Wachten op het laden van de applicatie kan voor frustraties zorgen. Dit is een grote frustraties voor *productmanagers* die efficiënt en effectief een aantal operaties willen uitvoeren.

Het dynamisch inladen van dependencies kan laadtijden verkorten. Voornamelijk dependencies met een significante grootte en situationeel gebruik (Quagga.js later meer) hoeven niet vanaf het begin ingeladen te worden.

Door dependencies vooraf te laden kunnen wachttijden worden verminderd naar 0. Door vanuit de navigatieboom alle dependencies van connected edges te laden hoeven te laden hoeft er tijdens een gebruikersactie niets te worden geladen omdat alles al geladen is. Ook bij zoekresultaten met verschillende pagina’s kan alvast de volgende pagina worden geladen.

Bepaalde onderdelen kunnen ook worden gecached in de browser, server en localstorage. De browser doet cached bepaalde objecten automatisch, maar bepaalde onderdelen kunnen niet makkelijk gecached worden.

Zoekqueries met pagination kunnen alvast de tweede pagina laden.

### 

### Hardware

Mobiele telefoons beschikken over minder geheugen dan een computer voor dezelfde prijs. Onzorgvuldig gebruik van dit geheugen kan leiden tot frustratie van de gebruiker. Javascript is garbage-collected, maar een combinatie van lange sessies en slecht objectbeheer kunnen op langere duur het geheugen van elk apparaat volstouwen. Het is belangrijk een goed objectmanagement uit te voeren, en niet-relevante objecten uit het geheugen te laden.

Een goed framework zoals AngularJS of React kunnen een applicatie architectuur te forceren, en door een architectuur te forceren (MVVC) is het duidelijk welke dependencies (views, controllers en bijbehorende objecten) wel-of niet relevant zijn.

## Voordelen

De meeste smartphones en tablets komen met capaciteiten die niet standaard beschikbaar zijn op een normale desktop.

### Touchscreen

Touchscreens zijn intuitief in gebruik. Bijna alle touchscreens komen met de capaciteit voor multi-touch. Touchscreens zijn zeer goed in herkennen van gestures (gebaren). Dit zijn relatief grove bewegingen die dingen bewerkstelligen. Denk bijvoorbeeld aan het openen van een menu.

### Microfoon

De meeste tablets en smartphones zijn uitgerust met een microfoon. Deze kan gebruik maken van nieuwe ontwikkelingen in het gebied van spraakherkenning. De toepassing van spraakherkenning is niet triviaal, en de toepasbaarheid op de werkvloer is niet ideaal, maar het is in de praktijk mogelijk.

### Camera

Een camera met goede resolutie kan worden gebruikt om teksten te herkennen. De informatie in een stuk tekst kan met een computer worden ingelezen en als invoer worden gebruikt. Dit zou de moeizame invoer van tekst op een touchscreen gedeeltelijk kunnen vervangen.

Momenteel zijn er geen browserstandaarden om mobile camera

### Tactiel / locatie gebaseerde informatie

Ik heb geen toepassing voor de tactiele en locatie-gebaseerde informatie kunnen bedenken in enige context van de opdracht.

## Deelconclusie

Een mobile platform heeft een flink aantal nadelen tegenover een desktop, voornamelijk qua overzichtelijkheid en invoer van informatie. Het is minder effectief en efficiënt om met een telefoon of tablet de applicatie te beheren. Dit leidt tot frustraties bij de eindgebruikers (zie hoofdstuk 1).

Een Responsive Web Design framework kan de applicatie overzichtelijk maken door de inhoud van de applicatie te schalen naar het formaat van het platform. Er zijn meerdere platformen getest, en zijn qua ondersteuning van RWD allemaal ongeveer het zelfde. Er is geen sluitende beslissing gemaakt, omdat dit meer in de handen van een eventuele webdesigner ligt. Voor de praktische; korte term is er gekozen voor bootstrap vanwege de hoge native ondersteuning van javascript-frameworks.

De invoer van data is een van de meest frustrerende aspecten van mobiele applicaties. Bepaalde functionaliteit, zoals zoeken en automatisch aanvullen van velden, kan er met minder invoer meer worden gedaan. Dit is uitgewerkt in het proof-of-concept. Meer esoterische functionaliteit, zoals het gebruik van Optical Character REcognition, is door diract niet als waardevol gezien.

Om de telefoon of tablet wél doeltreffend platform te gebruiken, moeten de krachten van mobiele platformen worden uitgebuit. Deze krachten liggen voornamelijk bij intuitief gebruik van touchscreens en de camera.

# Hoe kunnen we de webapplicatie structureren om de mobiele toegankelijkheid te vergroten?

De context waarin de gebruiker zich bevind moet grafisch duidelijk worden weergegeven. Dit kan op veel verschillende manieren; het tonen van een klein icoontje is er een van.

Momenteel is de meeste informatie getabt aanwezig, maar dit is niet per sé de meest gebruiksvriendelijke manier (onderzoek terugvinden). Onderzoek toont aan dat er een subjectieve voorkeur is voor een ‘plat’ geheel, terwijl gebruikers .3 seconden effectiever zijn in een gelaagd ontwerp. Gebruikers krijgen een subjectief gevoel van overzicht in een platte applicatiestructuur, maar kunnen hun taken net iets sneller uitvoeren in een gelaagd ontwerp. Ik ga er van uit dat de complexiteit van de applicatie toeneemt met een hogere gelaagdheid.

*Bronnen van onoverzichtelijkheid*

*Tonen van relaties*

*Platte structuur*

*Hoge doorklikbaarheid*

Typische CRUD functionaliteit kan dezelfde views delen, en dit bespaard aan complexiteit voor de eindgebruiker.

Relaties tussen verschillende objecten

Binnen de concentrator zijn de volgende contexten aanwezig. De context van

### Business rules

### Export- en importregels

### Producten en attributen

### Kanalen

## Details abstraheren

# Wat is nodeJs en waarom gebruikt iedereen het?

Elke technologie komt met een bepaalde ontwikkelomgeving, javascript komt met Node.

De complexiteit van front-end applicaties is sterk toegenomen. Een explosie aan functionele eisen aan front-end applicaties heeft voor een oerwoud aan front-end libraries gezorgd. NodeJS en NPM zijn de voornaamste distributiekanalen voor front-end libraries.

## Node.JS

Node.js is een javascript engine die niet op de browser draait. Het bevat native OS integraties en kan zo systeemresources directer benaderen. Er zijn bijvoorbeeld webservers die op Node kunnen draaien, met hun eigen voordelen. NodeJs is niet meer weg te denken uit het front-end ecosysteem.

Node.js is ook snel.

## Dependency management

NPM is een repository die veel functionaliteit voor front-end ontwikkeling bevat. Het zijn over het algemeen kleinere packages die kleine functionaliteit bevatten.

Bower bevat naast javascript ook veel front-end functionaliteit.

Het ophalen van dependencies gebeurt via de commandline.

## Build-management

Alhoewel er geen code wordt gecompileerd, moeten dependencies wel op een gedefinieerde plaats. Handmatig kopieren is tijdrovend en foutgevoelig.

Gulp is een buildmanager voor nodejs. Het bestaat uit taken die automatisch kunnen worden uitgevoerd. Buildstappen zoals “clean”, ”build” of ”install” kunnen met gulp worden geautomatiseerd. Ook kan gulp de ontwikkelaar ondersteunen door bijvoorbeeld gedeeltelijke “builds” te doen als de ontwikkelaar een bestand opslaat.

## Testen en ontwikkelondersteuning

NPM en Bower repositories bevatten meerdere testservers en verschillende testformaten. Karma is een van de test-engines waarbij er een sterke code.

## Integratie met visual studio 2015

Visual studio komt met integraties voor NodeJS. Gulp taken kunnen automatisch worden gedraaid tijdens het builden van de backend.

## Transpilers

Hoewel javascript niet gecompileerd hoeft te worden. is javascript zo dynamisch en anderzijds gelimiteerd dat het niet veel

Transpilers interpreteren een superset van javascript, en cconverteren deze naar javascript.

Javascript is een zeer dynamische taal en functies van objecten kunnen in runtime worden veranderd. De voornaamste keuze

Typescript is een transpiler en superset van javascript die een .ts files naar javascript transpiled. De voordelen van typescript zijn Typing, en integratie met intellisense. Intellisense.

## Conclusie

Er is gekozen om de ontwikkeling van de front-end met Node te ondersteunen. Voornamelijk omdat de hoeveelheid dependencies enorm is toegenomen sinds de laatste concentrator front-end. Het PoC maakt gebruik van beide NPM en Bower.

Node maakt het ook mogelijk de pakketten de functionaliteit beter op te splitsen door.

Er zijn meerdere gulp taken en geintegreerd in visual studio. Het gulp-script heeft de volgende capaciteiten;

* dependencies ophalen
* dependencies en code kopieren naar distributiefolder
* Distributiefolder opschonen
* Andere assets zoals plaatjes beheren
* Automatisch kopiëren van asset als deze wordt gewijzigd
* Geïntegreerd in VS15 solution

Door de complexiteit van de applicatie is het aan te raden Typescript te gebruiken. Het gebruik van intellisense voorkomt kleine foutjes die programmeurs maken, en maakt het makkelijk componenten te onderzoeken. Daarnaast helpt het gebruik van types het correct programmeren van de applicatie met gedefinieerde methodes.

Er is echter bewust voor gekozen om typescript niet in het PoC te integreren, aangezien ik geen ervaring had met javascript, en al helemaal niet met typescript.

# Welke javascript frameworks sluiten het beste aan bij de vereisten?

Er valt niet aan een framework te ontkomen. Om goed om te gaan met verschillende functionaliteit moet er een keuze worden gemaakt naar de verschillende frameworks. Het gaat niet zozeer om de basisfunctionaliteit.

Enkel CSS is niet genoeg om de website responsive te maken, een javascript library is verantwoordelijk voor het mogelijk maken van de interactiviteit en staat veranderingen. Deze library moet binnen een bastaand ecosysteem passen, welke soms behoorlijk expansief kan zijn.

Bovenop het basisframework draait dan een library en een CSS die verantwoordelijk voor het uiterlijk van de webapplicatie. Er zijn verschillende opties, maar er is een voorselectie gemaakt.

Vanuit de business-case is er een sterke neiging naar een bedrijfbreed gedragen oplossing voor de frontend, en er is al een PoC voor angular opgezet.

Er zijn vanuit het team en eigen inzicht verschillende criteria aan het framework gesteld;

* Het moet een GRID (uitgebreide tabel) ondersteunen
* De ontwikkeling van het framework moet stabiel zijn
* Het moet responsive libraries ondersteunen
* Definieert een architectuur
  + Bied een alternatief aan DOM manipulaties.
* Komt met ingebouwde functionaliteit
* Is goed te gebruiken voor back-end developers

De wens is naar een MVC of MVVC architectuur te gaan.

## AngularJS

Groot ecosysteem van google. Angular1 was een van de eerste gratis javascript frameworks. De ontwikkelars van Angular zijn in 2010 opgekocht door Google.

Angular1 komt met een groot scala aan functionaliteit; onder andere het beheren van cookies, filters, en object-bindingen.

Angular2 is een herschreven versie van Angular1, en is momenteel in beta.

### Voordelen

Angular is gratis

On-officieel het SPA-framework van google, en word gefinancierd uit de budgetten van google.

Angular is het meest toegepaste framework, en heeft de meeste libraries van derde partijen. Functionaliteit zoals GRIDS, touch en integraties.

Angular is binnen Diract-IT al eerder successvol gebruikt voor een proof of concept.

Angular is een echt framework, met veel beschikbare functionaliteit.

### Nadelen

Angular 2 is een herschreven versie van angular 1, en bijna alle functionaliteit van Angular 1 gaat veranderen. Angular2 is op het moment van schijven net in beta, en de meeste Angular1 community packages zijn nog niet overgeport naar Angular2.

Angular1 is zeer breed opgezet, en bevat overal functionaliteit voor. Sommige modules zijn echter sterk ondermaats, zoals de routing module. Er zijn wel alternatieven voor deze modules.

## Sencha ExtJS

De eerste versie van Ext.js kwam in 2007 uit en was daarmee de eerste. Het is een zeer sterk platform voor het ontwikkelen van desktop-achtige webapplicaties. Slepen van elementen is makkelijk definieerbaar. De voorbeelden met wat

### Voordelen

Meest uitgebreide GRID, met grafiekjes in de vakken.

Professionele ondersteuning bij aankoop.

Heel goed in het maken van webapplicaties die op een desktop-applicatie lijken.

### Nadelen

Van nature niet responsive. Er is een splitsing in Ext.JS 6 waarbij alle geavanceerde elementen (GRID, layout manager, mooie elementen) op een niet-responsive manier zijn gemaakt.

De voorbeelden zijn onleesbaar zonder extensieve kennis van Ext.JS .

Kost 9000 tot 14000 euro.

Heeft bijna geen community met libraries en toevoegingen.

Voorbeelden zijn niet bepaald verhelderend.

## ReactJS

Onderdeel van het facebook FLUX ecosysteem. Flux kwam in 2014 op de markt, maar is pas recentelijk opgepikt door nieuwe projecten.

### Voordelen

Kleine losstaande onderdelen maken het mogelijk een kleine, snel laadbare applicatie op te leveren.

Veel functionaliteit van derde partijen.

Ook gratis.

Onofficieel het framework van en voor facebook.

### Neutraal

Onderhouden door facebook.

### Nadelen

Tijdens het onderzoek brak reactJS door een van de dependencies. Left-pad was een stukje javascript dat een string naar een bepaalde grootte bracht door teken aan de linkerhand in te voeren. Het was een indicatie van de kwaliteit van facebook ontwikkelaars.

React betreft enkel views, andere functionaliteit (controllers, models) moeten via andere FLUX onderdelen worden geregeld.

De voorbeelden zijn simpel en helder, maar niet optimaal. Templates staan in de javascript controllers, en er weinig sprake van scheiding tussen views en controllers.

## Deelconclusie

Javascript is een zeer dynamische taal, en voor niet front-enders niet direct inzichtelijk. De keuze is gevallen voor Angular1 omdat deze een duidelijke architectuur oplegt. De ontwikkelaars zijn geen javascript specialisten, en zijn primair geinteresseeerd in het ontwikkelen van backend functionaliteit.

Daarnaast is er out-of-the-box veel functionaliteit in Angular beschikbaar, in tegenstelling tot andere frameworks die meer op integraties van andere producten berusten (react). Dit is niet makkelijk inzichtelijk te maken, en een back-end ontwikkelaar wil niet zoeken welke dependency welke functionaliteit bevat.

Ext.JS is een zeer mooi framework, maar de kracht van Ext ligt overduidelijjk niet bij responsive design. Het gebruikt Bootstrap voor de responsive gedeeltes, en de paradepaardjes uit de voorbeelden zijn niet responsive. Ext is wel prachtig mooi, maar de meerwaarde voor Ext is niet toepasbaar op mobiele platformen.

Daarnaast ligt het al bijna vast dat andere teams binnen diract Angular gaan gebruiken. De kennisdrempel zal hiermee verlaagd worden, waardoor bijvoorbeeld inhouse ontwerpers ook makkelijk aan de front-end van de concentrator kunnen sleutelen.

# Welke CSS framework past het beste bij concentrator?

CSS frameworks zijn verantwoordelijk voor de uitstraling van de applicatie.

Front-end frameworks bestaan uit een grote collectie herbruikbare elementen zoals knopjes, titels en etc. Deze worden door middel van klassen geïmplementeerd.

## Waarom gebruiken ontwikkelaars een front-end framework?

Er wordt een stuk meer verwacht van een front-end. De uitstraling van een webapplicatie is voor sales net zo belangrijk als de technologie die erachter draait. Door een framework te gebruiken is het mogelijk de kunde van een front-end professional te implementeren in plaats van zelf iets matigs te gebruiken.

De aanbieders van browsers nemen het niet zo nauw met de standaarden. Er zijn nogal veel verschillen in welke browsers wat voor een functionaliteit bieden, en op welke manier. Sommige functionaliteit moet daarom meerdere keren worden geïmplementeerd zodat deze op alle grote browsers zou werken. Front-end frameworks bestaan uit een collectie van veelgebruikte onderdelen.

## 

## Selectie van frameworks

Het front-end framework moet zo begrijpelijk mogelijk blijven. De meeste Concentrator ontwikkelaars werken incidenteel met de front-end, en het is niet hun favoriete bezigheid.

Over het algemeen zijn

## Material Design

Iedereen met de Google Gmail app kent dit ontwerp. MD is ontwikkeld door google voor in-house projecten. Het is gratis beschikbaar, maar komt niet met harde eisen.

### Voordelen

### Goed voor mobiele applicaties.

### Nadelen

Minder goed voor niet mobiele applicaties

Zwaardere hardware-eisen

Minder goede ondersteuning

## Bootstrap

Bootstrap is ontwikkeld bij twitter. Het was een van de eerste frameworks die alle uiterlijke functionaliteit probeerde te standaardiseren. Daarnaast was bootstrap het eerste framework dat op beide desktop en mobile goed toonbaar was.

### Voordelen

In actieve ontwikkeling met ondersteuning van twitter

Goede integratie met Angular.

Veel functionaliteit uit de doos.

Veel libraries die visuele functionaliteit aanbieden zijn op bootstrap gebaseerd en vereisen bootstrap als dependency.

Uitgebreide documentatie met voorbeelden.

### Nadelen

Het uiterlijk van bootstrap is zeer herkenbaar. Er zijn mensen die daar een hekel aan hebben, maar er zijn .css scripts die het uiterlijk veranderen.

## 

## Metro-ui

Metro-ui is ontwikkeld door een team uit wit-rusland. Het is een interpretatie van de windows metro huisstijl.

### Voordelen

Mooi

### Nadelen

Voorbeeldjes zijn maar matig responsive.

Kleine community

## MetroCSS

De

## Zurb foundation

Voor ontwerpers

### Voordelen

Veel functionaliteit die belangrijk is voor ontwerpers.

Grote community.

### Nadelen

Veel functionaliteit die niet relevant is aan de opdracht.

## Deelconclusie

Er zijn relatief weinig verschillen tussen de frameworks te bekennen. Er zijn weinig relevante (RWD) verschillen tussen de frameworks te bekennen. Er is daarom gekozen voor bootstrap, omdat deze out-of-the-box het meeste functionaliteit bevat en door de community het best ondersteund wordt.

# Hoe werkt een service oriented architecture?

Concentrator 2 was ontwikkeld naar een Model View Controller architectuur. Dit is goed voor gebruikers, maar minder goed voor communicatie tussen systemen.

Het voornaamste idee is dat de interacties tussen server en gebruiker, en server-server dezelfde interface gebruiken. Dezelfde functionaliteit die voor menselijke gebruikers beschikbaar is, moet ook beschikbaar zijn voor andere systemen.

Dit in tegenstelling tot bijvoorbeeld MVC, dat specifiek bedoelt is voor menselijke eindgebruikers, en meestal moeilijk voor andere systemen bereikbaar is. De volgende versie van de concentrator wil de software als een service aanbieden, waardoor communicatie tussen systemen een grotere prioriteit krijgt.

Omdat de webapplicatie gewoon op de browser beschikbaar moet zijn, is het verstandig een specifiekere architectuur te kiezen die over http beschikbaar is.

## REST

REST staat voor representational state transfer en is bedacht door Roy Fielding, een software doctorandus. Roy Fielding ontworp REST naar de verschillende problemen die organisaties ondervonden bij hun IT activiteiten.

Er zijn keuzes te maken

De doelen die Fielding voor ogen had was een architectuur die de volgende eigenschappen had;

* Performance
* Schaalbaarheid, hotswappen van de codebasis
* Simpliciteit van interfaces, om de
* Aanpasbaarheid van onderdelen terwijl de applicatie draait.
* Zichtbaar voor agents (caching, load-balancing)
* Overdraagbare functionaliteit
* Betrouwbaarheid, zelfs bij falen vna de back-end systemen

Aan de hand deze doelen heeft Fielding de volgende beperkingen opgelegd:

* **Client-Server** Communicatie verloopt tussen client-server
* **Stateless –** De server houdt geen sessies bij, en elke request bevat alle informatie die nodig is om de request succesvol af te ronden.
* **Cache –** De server kan responses cachen, en is verantwoordelijk voor de invalidatie.
* **Interface / uniform contract –** De communicatie tussen systemen is expliciet en ge
* **Layered system –** De API is enkel een lag in het system, en is niet sterk gekoppeld met andere lagen.
* **(optioneel) code on demand -**  De API kan de client van functionaliteit voorzien; bijvoorbeeld in het interpreteren van rare datatypes. De server kan functies en modules aan de client verstrekken.

#### HATEOAS

HATEOAS staat voor Hypermedia as the Engine of Application State. Het voornaamste idee is dat alle functionaliteit van de server bereikbaar is vanaf een centrale root “/api/”. De RESTful service geeft dan de verschillende functies in een context weer. Het belangrijke van HATEOAS is dat het de service op het gebruikersniveau inzichtelijk maakt. Het is daarmee makkelijk voor de programmeur om de iets tegen de API aan te bouwen. Dit is een vorm van zelf-documenterende code.

De server kan ook nieuwe, of niet gestandaardiseerde mediatypen aangeven. Ook functionaliteit om nieuwe media kan door middel van code-on-demand aan de client worden verstrekt.

## Lagen Structuur

De front-end is een abstractie van de processen die op de achtergrond draaien, en is relatief ontkoppeld van de business logica en heeft zeker niets te maken met de database van de applicatie.

Verschillende back-end applicaties kunnen dezelfde front-end delen.

## SOAP

Soap staat voor Simple Object Access Protocol, dat minder simpel is dan REST. Een SOAP bericht is een XML document dat aan bepaalde standaarden voldoet. De specificatie vereist een speciale XML opbouw waarin in ieder geval een envelop met een body aanwezig moet zijn.

De interpretatie en afhandeling van SOAP berichten zijn gestandaardiseerd. SOAP is daarmee een protocol, en REST is geen protocol. Het protocolzijnde van SOAP maakt het mogelijk om heel duidelijke afspraken te maken tussen de client en server.

SOAP ondersteund transacties en messaging. SOAP daardoor meer geschikt voor betrouwbare informatie-uitwisseling tussen systemen.

## Deelconclusie

Eigenlijk is er geen overlap tussen de gebieden waar SOAP voor bedoelt is, en REST voor bedoelt is. SOAP is een protocol dat voornamelijk bedoeld is om ACID transacties tussen systemen uit te voeren, terwijl REST erop gericht is services aan te kunnen bieden.

ACID kritieke functionaliteit kan met SOAP worden gedaan, op zich met het verstrekken van een javascript SOAP client met Code-on-demand. Dit is echter niet praktisch uitgewerkt vanwege een gebrek aan tijd.

REST is een architecturale stijl gericht op ontkoppelen van webservice, applicatie en client, terwijl SOAP een echt protocol behelst.

Er is gekozen om in het PoC alleen de REST laag te implementeren, voornamelijk omdat het implementeren van een SOAP laag niet in de tijd zou kunnen.

Vanwege de eerdere ervaringen en kennis binnen Diract-it is er een keuze gemaakt voor het ontwikkelen van een webservice gebaseerd op IIS server met ASP Core. Hoewel ASP Core momenteel nog in alpha ontwikkeling staat, zijn er werkende release candidates met al relatief veel functionaliteit.

Om de werking van de applicatie aan te tonen is er wel een back-end aanwezig, maar deze is enkel illustratief. Deze is ontwikkeld met EntityFramework 7, dat ook in staat van alfa is.

Het Poc beschikt qua functionaliteit over het draaien van queries, updaten en toevoegen en verwijderen van producten.

# Hoe implementeer ik de authenticatie van het systeem?

Authenticatie en autorisatie zijn integraal voor het werken van het systeem.

Omdat een REST server geen sessies bijhoud is authenticatie niet zo vanzelfsprekend als bij een MVC applicatie.

Om integratie de software voor derde partijen te versimpelen is er een keuze voor single-sign-on gemaakt door de lead-developer van concentrator. Dit houdt in dat de identiteit van een gebruiker door verschillende (externe) partijen kan worden beoordeeld.

Binnen AspCore is er een module beschikbaar (Identity) die mogelijk maakt.

## Deelconclusie

Authenticatie verloopt nu via een basic request, waarbij de server een cryptografische authenticatie cookie meestuurt waarin de rollen van de gebruiker staan. Deze cookie word bij elk request meestuurd, en kan worden ge invalideert door Angular als . Het legen van alle cookies volstaat als logout aangezien de gebruiker.

Omdat authenticatie niet mag worden gecached en het significant is moet het met een POST worden gedaan.

Er is een korte impressie van login gemaakt, maar voorlopig nog geen daadwerkelijke implementatie van enige identiteit

# Conclusie

# Verklarende woordenlijst

Responsive Web Design

REST

Front-end

Back-end

HATEOAS; Hypermedia as the Engine of Application State

Service oriented architecture

# Bibliografie

**Harrison, Rachel, Flood, Derek en Duce, David. 2013.** *Usability of mobile applications.* sl : SpringerOpen, 2013.

**http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\_arch\_style.htm**