Professionele Bachelor Toegepaste Informatica

ARGEÜS

eye for your financial software

Implementatie Word-documentgenerator

Jonas Gerits

Promotoren:

Reinaut Krekels
Lowie Vangaal

Argeüs bvba
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Implementatie Word-documentgenerator

Jonas Gerits

Promotoren:

Reinaut Krekels — Argeüs bvba
Lowie Vangaal — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Na een intensieve twaalf weken komt mijn stage ten einde. Met dit dankwoord leg ik de laatste loodjes aan mijn eindwerk. Tijdens deze periode heb ik zeer veel bijgeleerd op verschillende vlakken. Op vlak van applicatieontwikkeling en sociale vaardigheden ben ik het hardst gegroeid. Maar dit kon ik natuurlijk niet zonder de hulp van een aantal mensen. Deze mensen zou ik graag even willen bedanken met behulp van dit dankwoord.

Ten eerste wil ik graag al mijn collega's binnen Argeüs bedanken voor de fijne samenwerking. Er was altijd een goede sfeer aanwezig bij jullie en dat zorgde er ook voor dat ik me snel thuis voelde binnen Argeüs. In het bijzonder zou ik graag Reinaut Krekels, mijn bedrijfspromotor, willen bedanken voor alle feedback en begeleiding die ik gekregen heb tijdens deze twaalf weken. Hij heeft me altijd goed op pad gebracht op momenten wanneer ik niet wist hoe ik een probleem moest aanpakken.

Verder wil ik ook hogeschool PXL bedanken. Zonder hun samenwerking was ik natuurlijk nooit op dit punt gekomen. Tijdens mijn opleiding bij hogeschool PXL heb ik veel bijgeleerd wat ik tijdens mijn stage kon toepassen. Binnen hogeschool PXL wil ik iemand in het bijzonder bedanken namelijk Lowie Vangaal, mijn hogeschoolpromotor. Hij gaf mij elke week duidelijke feedback die ik direct kon gaan verwerken. Ook stond hij altijd klaar voor mij ingeval er zich problemen voordeden tijdens mijn stage.

Vervolgens zou ik ook graag mijn ouders en mijn vriendin bedanken omdat ze altijd goed voor mij gezorgd hebben tijdens deze stage periode. Ze stonden altijd klaar voor mij op momenten dat ik het even niet meer zag zitten. Ze luisterden naar de problemen waar ik mee zat en gaven me wijze raad.

Ten slotte wil ik ook mijn vrienden bedanken voor de momenten waar ik mijn hoofd even op nul kon zetten. Deze momenten zorgden ervoor dat ik er elke week weer volop tegenaan kon gaan.

Abstract

Argeüs biedt verschillende softwarepakketten aan waaronder de successiecalculator. Via een website kan alle informatie van de klant ingegeven worden die nodig is om een aangifte op te stellen. De huidige implementatie gebruikt een *templating*-systeem, waarbij de ingegeven data de *placeholders* binnen een Word-document vervangt. Alle data die de klant heeft ingegeven wordt dan ingevuld in de *inputboxen* van de template. Dit systeem is niet goed onderhoudbaar en niet erg analyseerbaar.

Om dit proces te verbeteren wordt het volledige aangiftedocument, vanuit Java-code, opgebouwd met behulp van een *library* die al eerder was opgebouwd binnen Argeüs. Deze nieuwe implementatie bouwt het hele document op vanaf nul, met alle secties met de daarbij horende *headings*, titels, tabellen, voetnoten, enzovoort. Doordat alle secties vanuit code gegenereerd worden, kunnen de *developers* binnen Argeüs deze makkelijk aanpassen door enkele regels code aan te passen.

De frontend van de successiecalculator wordt opgebouwd via Eclipse Scout 5. Deze versie van Eclipse Scout bereikt stilaan haar einde qua ondersteuning en haar technische opbouw sluit minder goed aan bij de structuur die wenselijk is in de nieuwe projecten binnen Argeüs.

Doordat er zo'n grote diversiteit is aan frontendframeworks, wordt er onderzocht welk framework het beste bij Argeüs past, omdat een goed framework heel wat voordelen heeft, zowel technisch als economisch.

Het onderzoek doorloopt verschillende fasen. In de eerste fase worden er een tiental frameworks vergeleken. Dit gebeurt aan de hand van verschillende aspecten zoals populariteit en testbaarheid.

In de tweede fase worden een vijftal frameworks geselecteerd die het meest relevant zijn voor Argeüs, dus rekening houdend met de aspecten die zij belangrijk vinden binnen een framework. Deze vijf frameworks worden vervolgens in detail onderzocht.

Tenslotte wordt er onderzocht welke twee frontendframeworks het meest geschikt zijn voor Argeüs. Van deze twee frameworks wordt er een prototype uitgewerkt zodat alle aspecten visueel toegelicht kunnen worden aan de collega's binnen Argeüs.

Inhoudsopgave

Dankwoord ii
Abstract iii
Inhoudsopgave iv
Lijst van gebruikte figuren vi
Lijst van gebruikte tabellen vii
Lijst van gebruikte afkortingen en begrippen viii
Inleiding 1
I. Stageverslag 2
1 Bedrijfsvoorstelling 2
1.1 Profiel 2
1.2 Ligging 3
1.3 Organigram 4
1.4 Kadering stageopdracht binnen het organigram 5
2 Voorstelling stageopdracht 6
2.1 Probleemstelling 6
2.2 Doelstellingen 6
2.3 Gebruikte technologieën 6
3 Uitwerking stageopdracht 7
3.1 Aanpak 7
3.2 Het huidige systeem 7
3.2.1 Inleiding 7
3.2.2 Werking 7
3.2.3 Voor- en nadelen 9
3.3 Het nieuwe systeem 10
3.3.1 Inleiding 10
3.3.2 Werking 10
3.3.3 Voor- en nadelen 19
3.4 Resultaten 20
4 Besluit 21
4.1 Programmeervaardigheden 21
4.2 Persoonlijk vlak 21
4.3 Sociaal vlak 21
4.4 Andere 21

II. Onderzoek topic 22
1 Vraagstelling 22
2 Methode van onderzoek 22
3 Resultaten 23
3.1 Literatuurstudie 23
3.1.1 Introductie 23
3.1.2 Bron 1: Angular vs React vs Vue: Which Framework to Choose in 2019 [2] 23
3.1.3 Bron 2: React vs. Angular vs. Vue.js: A Complete Comparison Guide [3] 24
3.1.4 Bron 3: React vs Angular vs Vue.js — What to choose in 2019? [4] 25
3.1.5 Inhoudelijke vergelijking van de bronnen 26
3.1.6 Conclusie 27
3.1.7 Reflectie 27
3.2 Fase 1 - De diverse frontendframeworks 28
3.2.1 Inleiding 28
3.2.2 Vergelijking van de frameworks 29
3.3 Fase 2 – De filtratie 33
3.3.1 Wat vindt Argeüs belangrijk? 33
3.3.2 Welke frameworks zijn het meest geschikt? 36
3.4 Fase 3 – De potentiële frameworks voor Argeüs 45
3.4.1 Inleiding 45
3.4.2 Angular prototypes 46
3.4.3 VueJs prototype 51
4 Discussie, aanbevelingen en conclusie 55
4.1 Welk framework wordt Argeüs aangeraden? 55
4.2 Aanbevelingen 55
4.3 Reflectie 56
Bibliografie 57

Lijst van gebruikte figuren

Figuur 1: Softwarepakketten van Argeüs 2
Figuur 2: Locatie Argeüs Lummen 3
Figuur 3: Organigram Argeüs 4
Figuur 5: Template: Gegevens van de overledene 8
Figuur 6: Functie voor het invullen van template "Gegevens van de overledene" 9
Figuur 7: Template van sectie "Gegevens van de erfgenaam, legataris of begiftigde die de erfenis verworpen hebben" 11
Figuur 8: BasicIdentityDataDto-klasse 12
Figuur 9: DeceasedIdentityDto-klasse 13
Figuur 10: BasicIdentityDataDtoFactory-klasse 14
Figuur 11: DeceasedIdentityDtoFactory-klasse 15
Figuur 12: DeceasedIdentityDocumentPart-klasse 17
Figuur 13: Optie om extra info te tonen 17
Figuur 14: Downloadformulier aangifte 18
Figuur 15: 'Identiteit van de overledene' gegenereerde sectie (Nederlands) 18
Figuur 16: 'Identiteit van de overledene' gegenereerde sectie (Frans + extra info) 19
Figuur 17: Bron 1 - community vergelijking 23
Figuur 18: Bron 2 - Populariteit grafiek 24
Figuur 19: Bron 3 - jobaanbiedingen per framework 25
Figuur 20: Bron 1 - jobaanbieding analyse 26
Figuur 21: Stack Overflow-vraag 32
Figuur 22: Grafiek frameworkscores 32
Figuur 23: Angular Material Input voorbeeld 36
Figuur 24: Grafiek van gevraagde frameworks in onlinevacatures 37
Figuur 25: Voorbeeld Ember Unit Test 39
Figuur 26: Unit test in VueJs 40
Figuur 27: Groottes van webframeworks 41
Figuur 28: npm-package downloads React vs. Angular 42
Figuur 29: Voorbeeld test in Jest 43
Figuur 30: Eigen geschreven componenttest in Aurelia 44
Figuur 31: Angular-prototype: Dashboard 46
Figuur 32: Angular-prototype: Admin Layout Component 47
Figuur 33: Angular-prototype: Angular Material 47
Figuur 34: Angular-prototype: Bootstrap voorbeeld 48
Figuur 35: Angular-prototype: applicatieoverzichtscherm 49
Figuur 36: Angular-prototype: Typescript validatiemogelijkheden 50
Figuur 37: Angular-prototype: HTML-validatieregels 50
Figuur 38: Vue parent component voorbeeld 52
Figuur 39: Vue Material button component 53
Figuur 40: Vue Material button 53
Figuur 41: Unit test in Jest 54

Lijst van gebruikte tabellen

Tabel 1: Overzicht gebruikte technologieën 6
Tabel 2: Voor- en nadelen van het huidig systeem 9
Tabel 3: Grootte- en snelheidsresultaten 20
Tabel 4: Overzicht van de frameworks 28
Tabel 5: Vergelijkend overzicht van de frameworks 29

Lijst van gebruikte afkortingen en begrippen

Begrip of afkorting	Uitleg
Back end	Het gedeelte van de applicatie dat niet zichtbaar is voor de gebruiker.
Branch	Een locatie waar een bepaalde versie van een applicatie opgeslagen staat.
Bug	Een fout die zich voordoet in een programma.
Commit message	Het bericht dat wordt meegegeven wanneer er code toegevoegd wordt aan een *branch*.
Cascading Style Sheets	Een bestand dat code bevat om de lay-out van een website te beheren.
Data Transfer Object (DTO)	Een object dat data bevat en gebruikt kan worden in verschillende processen.
Factory	Een klasse die instaat voor het aanmaken van objecten.
Framework	Een set van *tools* bovenop een programmeertaal om het werk van een *developer* makkelijker te maken.
Frontend	Het gedeelte van de applicatie dat zichtbaar is voor de gebruiker.
Hypertext Markup Language (HTML)	Een programmeertaal waar websites mee kunnen ontwikkeld worden.
Integrated development environment (IDE)	De omgeving waar een *developer* in ontwikkelt.
Mergen	Het samensmelten van twee *branches* binnen een *repository*.
Model-view-controller (MVC)	Een ontwerppatroon binnen applicatieontwikkeling. Via deze manier zullen complexe toepassingen in drie delen opgedeeld worden.
Model-view-viewmodel (MVVM)	Is een soort softwarearchitectuur die de verantwoordelijkheden verdeeld over drie soorten componenten.
Null	Een onbestaande objectverwijzing.
Nullpointer exceptions	Fouten in een programma veroorzaakt door een *null*.
Pipeline	Een ketting van processen die doorlopen wordt bij het *builden* van een applicatie op een server.
Renderen	Het laden van een applicatie.
Repository	Een locatie waar code van een applicatie wordt opgeslagen.

Inleiding

In deze paper wordt de implementatie van de Word-documentgenerator beschreven. Deze implementatie gaat ervoor zorgen dat er vanuit de Java-code een volledig Word-document wordt opgesteld. In dit geval wordt er een aangifte gegenereerd.

Eerst wordt het huidige systeem dat Argeüs gebruikt beschreven. Verder wordt deze ook helemaal toegelicht aan de hand van een kleine tutorial. Na het toelichten van het systeem worden ook de voor- en nadelen ervan uitgelegd. Deze zullen een beeld geven op de redenen waarom Argeüs graag van dit systeem zou afstappen.

Na het beschrijven van het huidige systeem en de nadelen ervan wordt het nieuwe systeem toegelicht. Eerst zal er een korte inleiding zijn om de basis van de Word-documentgenerator toe te lichten. Daarna zal deze meer in detail uitgelegd worden aan de hand van een tutorial. Op deze manier is het goed zichtbaar hoe de generator te werk gaat. Vervolgens worden ook van het nieuwe systeem de voor-en nadelen toegelicht.

Verder wordt in deze paper ook een onderzoek uitgeschreven. Dit onderzoek gaat over het selecteren van de meest geschikte frontendframeworks. Deze frameworks moeten zich aansluiten bij de punten die Argeüs belangrijk vindt binnen zo een framework. Het doel van dit onderzoek is dat Argeüs de conclusie kan meenemen naar de toekomst. Graag zouden ze het meest geschikte framework gaan gebruiken in de nieuwe projecten die er aankomen.

Het onderzoek wordt opgedeeld in drie verschillende fases. In de eerste fase wordt er onderzoek uitgevoerd om te weten te komen welke diverse frontendframeworks zich allemaal bevinden op de markt. In deze fase worden ze ook al direct vergeleken op verschillende vlakken.

Vervolgens wordt er in de tweede fase onderzocht welke aspecten Argeüs belangrijk vindt binnen een frontendframework. Verder worden er een aantal frameworks geselecteerd en worden deze verder uitgewerkt. In deze uitwerking worden de verschillende aspecten, die Argeüs belangrijk vindt, binnen dat framework toegelicht.

Ten slotte worden er in fase drie, twee frameworks geselecteerd als potentiële frontendframeworks. Er wordt toegelicht waarom deze twee frameworks de beste keuze zijn voor Argeüs. Ook wordt er van deze frameworks een prototype uitgewerkt. Aan de hand van deze prototypes worden de verschillende aspecten toegelicht.

I. Stageverslag

1 Bedrijfsvoorstelling

1.1 Profiel

In 2011 werd Argeüs initieel opgericht als een *financial planning* consultingbureau. Doorheen de jaren is haar activiteit uitgebreid naar het domein van softwareontwikkeling.

Argeüs houdt van innoveren om te evolueren. Samen met partners uit de financiële, notariële en juridische sectors innoveert ze de bestaande systemen, processen en werkwijzen om de adviesverlening verder te kunnen laten evolueren. De klanten zijn voor Argeüs de grootste drijfveer en inspiratie tot deze innovaties.

Zelf is Argeüs gericht op de creatie van maximale efficiëntie, verhoging van de rentabiliteit en kwaliteit. De detectie van behoeftes en optimalisatie komen vanuit de interactie van Argeüs met de gebruikersomgeving.

Er zijn al verschillende softwareoplossingen op de markt gebracht door Argeüs. Deze softwareoplossingen worden in samenwerking met partners, die jarenlange praktijkervaring en expertise in de desbetreffende materie hebben, opgebouwd. Op deze manier voldoen de oplossingen aan de praktijknoden en bieden ze een maximale meerwaarde in de praktijk.

Argeüs heeft een heel aantal klanten. Enkele voorbeelden zijn: Belfius, ING, Vandelanotte. Ze focust zich dus vooral op klanten zoals banken, notarissen en accountants.

Momenteel heeft Argeüs twee grote softwarepakketten op de markt gebracht. De '*succession software*' en de '*financial planning software*'.

Figuur 1: Softwarepakketten van Argeüs

In 2015 bracht Argeüs de successiecalculator op de markt. Dit is een berekeningsprogramma om successierechten te berekenen en te optimaliseren.

Binnen deze successiecalculator zijn er twee varianten beschikbaar. De gewone '*Light*' versie en de '*Pro*' versie. Bij de '*light'* versie wordt de focus gelegd op het snel adviseren. De '*Pro'* versie heeft uitgebreidere functionaliteiten die uitgebreid adviseren en optimaliseren mogelijk maakt.

Ook heeft Argeüs een softwarepakket genaamd 'Financial Planning Software' ontwikkeld om diepgaande en onderbouwde analyse te voeren op het fiscaalaspect van een persoon. Deze software bezit over een zeer uitgebreide rekenmotor en is geïntegreerd met de successiecalculator.

Argeüs houdt continu diverse nichemarkten in adviesverlening in de gaten. Zo kunnen ze te weten komen wat de behoeftes zijn en wat de mogelijke optimalisaties zijn. Dit kan leiden tot het lanceren van een nieuw product dat continu geoptimaliseerd en bijgestuurd wordt.

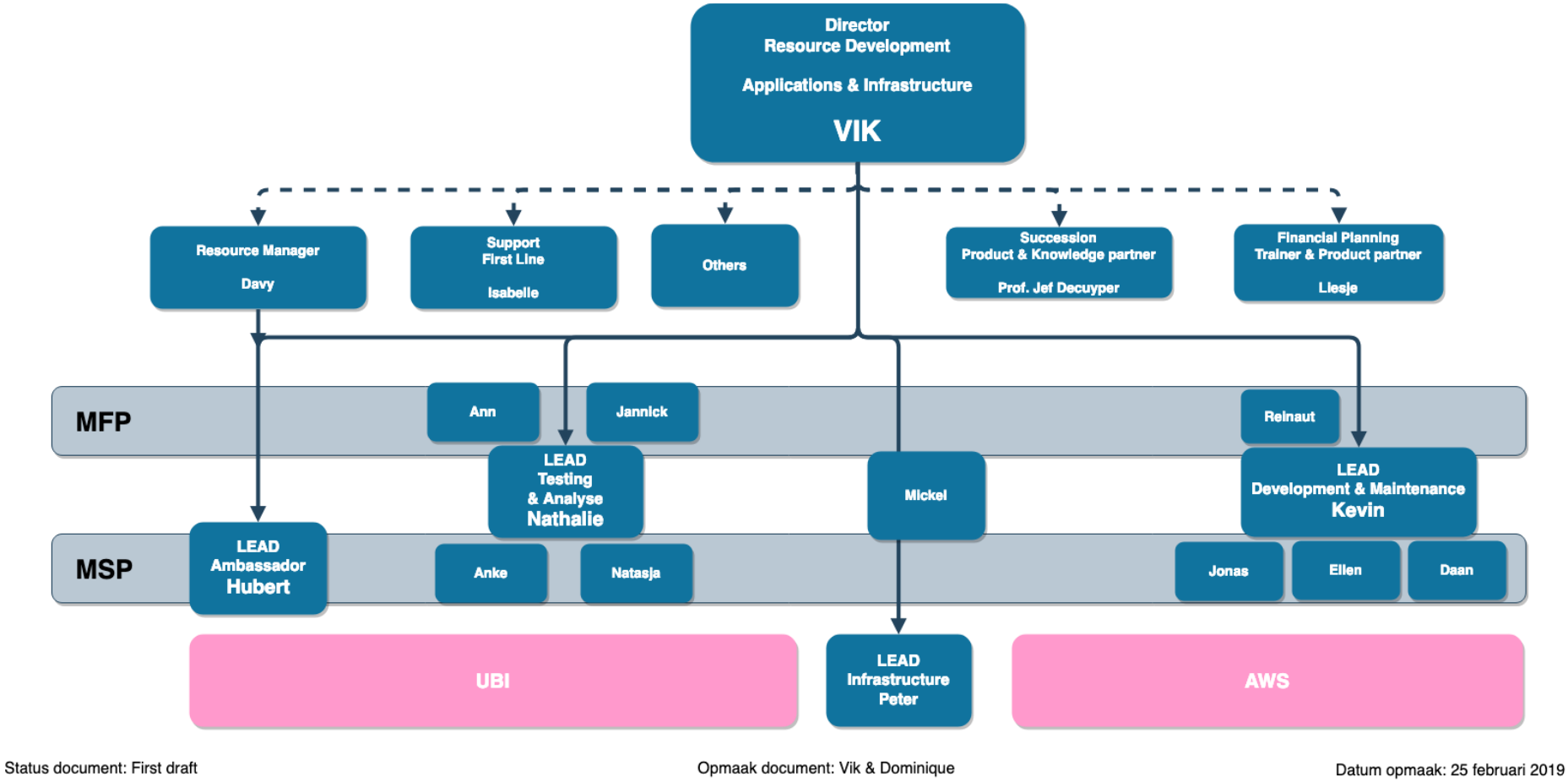
1.2 Ligging

In juni 2016 kocht Argeüs een kerk, gelegen in Lummen. Deze kerk is strategisch gelegen omdat het dichtbij het Klaverblad (Waar de E313 en de E314 samenkomen) ligt. Dit is ideaal voor het personeel en de klanten van Argeüs. [1]
Deze kerk zijn ze nog volop aan het verbouwen maar ze hopen er in januari 2020 zeker in te zitten. Momenteel zit Argeüs dus nog in moderne containers.

Figuur 2: Locatie Argeüs Lummen

Verder zijn er nog externen, die Argeüs ondersteunen, verspreid over heel België. Zij ondersteunen Argeüs door bijvoorbeeld bij notarissenkantoren langs te gaan en proberen de producten te verkopen. Verder leveren ze ook *support*. Dus als er klanten zijn die graag hebben dat er iemand langskomt kan het zijn dat het consultant van Argeüs is.

1.3 Organigram



Figuur 3: Organigram Argeüs

1.4 Kadering stageopdracht binnen het organigram

Binnen Argeüs zijn er twee grote projecten. Voor elk project is er één team. Ikzelf zal deel uitmaken van het MSP-team (het team van de successiecalculator). Dit omdat mijn opdracht onder het project van de successiecalculator valt.

Hoewel we eigenlijk in teams zijn opgedeeld, wordt er toch samengewerkt met andere mensen van het andere team. Deze staan altijd klaar om hulp te bieden waar het nodig is. Ann zal bijvoorbeeld ook instaan voor het testen van de Word-documentgenerator.

Vik is de *product owner* en bezit over de kennis die in elke applicatie wordt toegepast. Hij zal dan ook een aanspreekpunt zijn voor eventuele problemen in verband met aangiftes.

2 Voorstelling stageopdracht

2.1 Probleemstelling

Argeüs biedt verschillende producten aan. Eén van die producten is de successiecalculator. Dit is een grote applicatie met verschillende mogelijkheden. Eén van de mogelijkheden is om een aangiftedocument te genereren. Hiervoor dient de klant enkele gegevens in te geven via een website. Daarna kan hij ervoor kiezen om een Word-document te downloaden.

Het systeem dat ervoor zorgt dat er een ingevuld document uit komt, werkt met een oud systeem dat verschillende problemen met zich meebrengt. Het is moeilijk te onderhouden en het analyseren valt ook tegen. De redenen waarom deze problemen er zijn worden later in dit eindwerk toegelicht.

De reden waarom Argeüs deze opdracht heeft opgesteld is omdat de ontwikkelaars binnen Argeüs het systeem beu zijn. Ze geraken gefrustreerd als ze aanpassingen moeten doen aan de documenten omwille van de complexiteit.

2.2 Doelstellingen

Argeüs wil graag een nieuw systeem dat goed te analyseren en onderhouden valt. Zo zal er minder stress zijn bij de ontwikkelaars en zullen ze beter en sneller aanpassingen kunnen leveren. Ook zullen klanten minder lang moeten wachten op aanpassingen en *bugfixes*.

Op het vlak van analyseren is het doel om sneller terug te vinden waar er zich fouten voordoen. Met het huidige systeem is analyseren nogal omslachtig en onduidelijk. Met het nieuwe systeem wil Argeüs graag een duidelijke flow zien van hoe de data opgehaald wordt, hoe deze verwerkt wordt en hoe deze uiteindelijk in het document terechtkomt. Zo weten de ontwikkelaars, mocht er zich een fout voordoen, exact waar een fout zit en kunnen ze meteen beginnen met het oplossen ervan.

Verder moet het systeem ook goed te onderhouden zijn. Het huidige systeem is niet goed onderhoudbaar omwille van verschillende redenen. Deze redenen worden hieronder in detail beschreven. Argeüs wil nu dat het systeem de ontwikkelaars toelaat om gemakkelijk documenten aan te passen. Het is immers mogelijk dat een document bijvoorbeeld aangepast moet worden omwille van een wetswijziging.

2.3 Gebruikte technologieën

Tabel 1: Overzicht gebruikte technologieën

Besturingssysteem	Programmeertaal	IDE & text editors	Databases
• Windows 10	• Java 8	• Eclipse • Word	• MySQL

Framework & libraries	Versiebeheer	Projectmanagementtools	Opensourcesoftware
• Eclipse Scout 5	• SmartGit • Bitbucket	• Jira • Confluence	• OpenVPN

3 Uitwerking stageopdracht

3.1 Aanpak

Om de opdracht tot een goed einde te brengen moet er natuurlijk ook een goede aanpak zijn. Het eerste belangrijk aspect van een goede aanpak is het hebben van afspraken. Er worden afspraken gemaakt in verband met diverse codeerregels waar Argeüs zich aan houdt. Dit houdt dan dingen in zoals de naamgeving van variabelen, functies, enzovoort. Een voorbeeld van een codeerregel waar ik mij aan zal moeten houden is dat ik alle variabelen en functies in het Engels zal moeten zetten. Dit wordt toegepast omdat er vroeger ook soms Nederlandse termen gebruikt werden. Dit maakte het zoeken naar bepaalde klassen en functies moeilijker.

Ook worden er afspraken gemaakt wat betreft versiebeheer. De *commit messages* moeten bijvoorbeeld een vaste structuur hebben zodat Bitbucket overzichtelijk blijft. Binnen Argeüs is er bijvoorbeeld de afspraak dat *commit messages* moeten beginnen met het ticketnummer van een feature. Deze moet gevolgd worden door een korte beschrijving van wat er toegevoegd wordt aan de *branch*. Een voorbeeld van zo een *commit message* kan bijvoorbeeld dit zijn: 'MSP-2021 Add French Translations'.

Er worden ook algemene afspraken gemaakt in verband met hulp vragen aan andere collega's. Als er ergens een probleem is, mag er altijd hulp gevraagd worden aan andere collega's. Deze zullen hulp aanbieden zodra ze er de tijd voor hebben.

Voordat er aan het nieuwe systeem gewerkt kan worden, wordt er onderzoek gedaan naar het huidige systeem. Er wordt in kaart gebracht hoe het systeem werkt en wat hier nu de voor- en nadelen van zijn (zie 3.2.3).

Nadien wordt de nieuwe implementatie uitgewerkt. Tijdens de implementatie wordt er rekening gehouden met de aspecten waar Argeüs van af wilt. Ook wordt er tijdens de implementatie zoveel mogelijk gebruik gemaakt van *design patterns*. Dit om de code zo proper mogelijk te houden.

Ten slotte wordt de code om de twee weken gereviewd. Op deze manier wordt de code gedubbelcheckt. Als de code gedubbelcheckt wordt, zal de kwaliteit van de code ook hoger gaan liggen omdat fouten er sneller uitgehaald kunnen worden.

3.2 Het huidige systeem

3.2.1 Inleiding

Het huidige systeem werkt met een *templatingsysteem*. Het neemt een bestaand Word-document en vult dit in. Z'n document moet goed opgesteld zijn, dus alle koppen, secties, tabellen, enzovoort moeten geschikte namen hebben om een goed overzicht te houden. Verder moeten alle velden die ingevuld moeten worden, beschikken over een unieke parametervariabele.

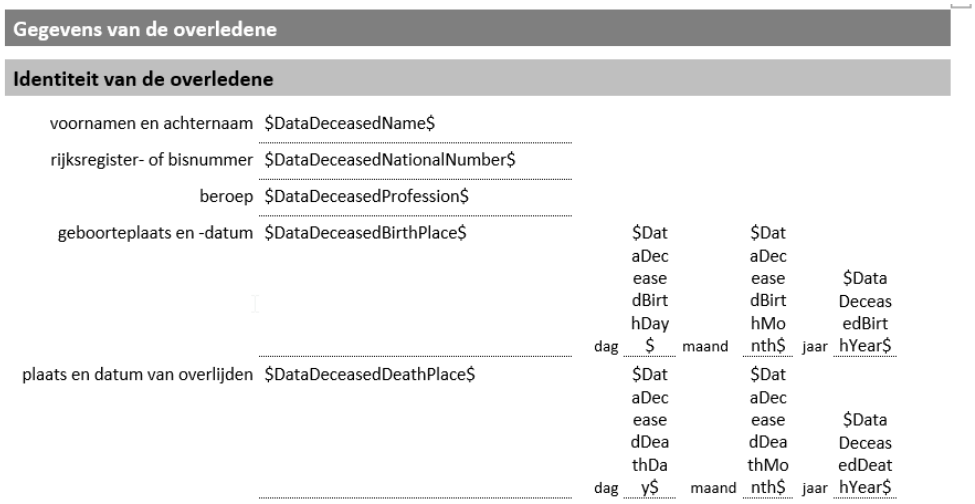
3.2.2 Werking

Om de werking van het huidige systeem zo goed mogelijk toe te lichten wordt elke stap die genomen moet worden om een bepaalde sectie in te vullen, uitgelegd.

3.2.2.1 Voorbereiding Word-document

De eerste stap is het voorbereiden van het Word-document dat nadien ingevuld zal worden door het systeem.

Stel dat de sectie “Gegevens van de overledene” geïmplementeerd moet worden. Hiervoor moet er een tabel aanmaakt worden. De lay-out en de positie van de tabel hangt af van het model dat geïmplementeerd wordt. Hier wordt de lay-out van het VLABEL-document toegepast (zie figuur 5).



Figuur 4: Template: Gegevens van de overledene

In bovenstaande afbeelding is te zien hoe de sectie eruit ziet. Er zijn twee hoofdingen toegevoegd met elk een eigen stijl. Verder is ook de tabel te zien. Deze is ook simpel toegevoegd met Word en beschikt over diverse invulvelden. Elk invulveld heeft zijn eigen parametervariabele. Aan de hand van deze parametervariabele wordt later het document ingevuld.

Om dit te realiseren wordt er gebruikgemaakt van de functionaliteiten binnen Word.

3.2.2.2 Verzamelen en invullen van data

Nu de template is aangemaakt, is het document klaar om ingevuld te worden. De volgende stap is dus het invullen en ophalen van de data. Hiervoor werd ooit de functie ‘createDeceasedData’ geschreven (zie figuur 6). Deze staat in voor het ophalen van de data en invullen van de sectie “Gegevens van de overledene”.

```
private void createDeceasedData() {
  List<Object> lst = new ArrayList<>();
  try {
    Tbl table = templateAdapter.createTemplate(DecTemplate.DATA_DECEASED);
    fillVariables(table.getContent(), factory.getDeceasedData(deceased));
    lst.add(table);

    addObjects(lst, true);
  }
  catch (ProcessingException e) {
    LOG.error("Error creating table for DeceasedData", e);
  }
}
```

Figuur 5: Functie voor het invullen van template "Gegevens van de overledene"

In bovenstaande afbeelding is de functie te zien die er voor zorgt dat de template ingevuld wordt. Eerst gaat deze de tabel ophalen van het Word-document. Deze wordt dan in een klasse 'Tbl' gestoken. In deze klasse zitten de verschillende parametervariabelen.

Nadat de tabel is opgevraagd, worden de gegevens opgehaald die de klant heeft ingevuld via de website. Dit gebeurt via een *factory* die op zijn beurt gebruik maakt van de diverse *services*. Via de *services* wordt de databank aangesproken waar alle data in zit. De *factory* geeft een tweedimensionale *array* terug met daarin alle data die nodig is om de tabel in te vullen en een bijpassende sleutel die verwijst naar een bepaalde parametervariabele.

Via de functie 'fillVariables', wordt de opgehaalde tabel ingevuld. Hiervoor gebruikt hij de data van de opgehaalde tabel en de opgehaalde data vanuit de *factory*. Deze functie gaat kijken naar de verschillende parametervariabelen van de tabel en gaat op zoek naar de data in de *array* aan de hand van de sleutel.

Nadien wordt de tabel in het effectieve Word-document ingevuld door de functie 'addObjects'.

3.2.3 Voor- en nadelen

Tabel 2: Voor- en nadelen van het huidig systeem

	Word-functionaliteiten	Analyseerbaar	Onderhoudbaar
Oud systeem	✚	✖	✖

Een voordeel van het *templatingsysteem* is dat alle Word-functionaliteiten tijdens het opstellen van het document makkelijk te gebruiken zijn. Het is makkelijker om posities van elementen te bepalen, groottes van tabellen in te geven, enzovoort.

Verder zijn er twee grote nadelen aan het werken met een *templatingsysteem*. Door hiermee te werken wordt het analyseren van de applicatie moeilijker. Stel dat er zich een fout voordoet bij het invullen van een tabel. Via code is het moeilijk te analyseren vanaf waar de data fout loopt. Er moet altijd gekeken worden op twee plaatsen, in de code en in de template.

Ook valt het systeem niet goed te onderhouden. Als er wijzigingen gemaakt moeten worden in de templates kan dit heel wat werk met zich meebrengen. Dit omdat er altijd gecheckt moet worden dat de parametervariabelen uniek moeten zijn, dat er niks gaat verspringen, enzovoort.

Verder moeten ook alle templates in andere talen aangepast worden. Argeüs heeft bijvoorbeeld de klanten de mogelijkheid gegeven om aangiftes ook in het Frans te genereren. Dus moeten ook alle Franse templates aangepast worden.

3.3 Het nieuwe systeem

3.3.1 Inleiding

Het nieuwe systeem zal werken met een generator. Deze documentgenerator gaat vanuit Java-code een Word-document opbouwen met behulp van een *library* die eerder al geschreven was binnen Argeüs.

Verder wilt Argeüs ook dat de nieuwe implementatie een duidelijke scheiding heeft tussen het opbouwen van het document en het opbouwen van de data.

Om de data goed gestructureerd op te bouwen wordt er gewerkt met *data transfer objects* (DTO's) en *factories*. Hieronder zal toegelicht worden hoe dit juist geïmplementeerd wordt bij het nieuwe systeem.

3.3.2 Werking

Om het systeem zo goed mogelijk toe te lichten zullen alle stappen die genomen moeten worden om een sectie te implementeren, uitgelegd worden. Ook in deze uitwerking wordt de sectie "Gegevens van de overledene" uitgewerkt.

3.3.2.1 Data Transfer Object

Argeüs wil graag dat de structuur van het nieuwe systeem duidelijk is en dat de data duidelijk opgebouwd wordt. Daarom wordt er gekozen om met DTO's te werken.

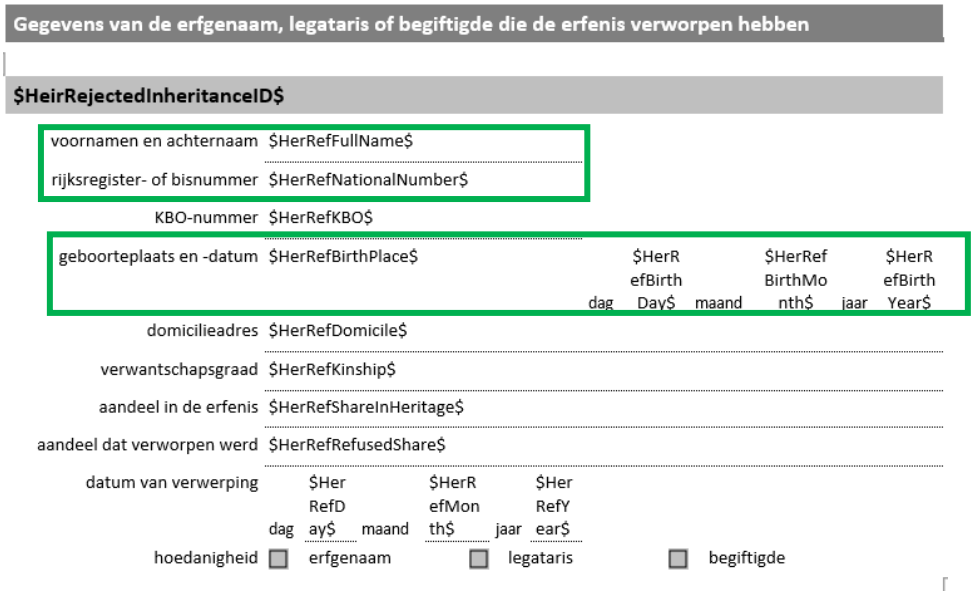
Deze DTO's zullen later gebruikt worden in de secties om de data op de juiste plek in te vullen. Dit brengt ook een groot voordeel met zich mee dat later besproken wordt.

In dit voorbeeld wordt de DTO uitgewerkt die nodig is voor de sectie "Gegevens van de overledene". Hiervoor moet er eerst gekeken worden naar welke data juist nodig is. Dit wordt gedaan door naar de oude template (zie figuur 4) te kijken. Eerst wordt de oude template bekeken omdat dit document helemaal moet nagebouwd worden in het nieuwe systeem.

Uit de tabel kunnen de volgende variabelen afgeleid worden:

- Voor- en achternaam (String)
- Rijksregister- of bisnummer (String)
- Beroep (String)
- Geboorteplaats (String)
- Geboortedatum (Date)
- Plaats van overlijden (String)
- Datum van overlijden (Date)

Wat er opvalt, is dat er een aantal variabelen terugkomen in andere secties van het aangiftedocument. De beste keuze voor een programmeur is dan om die variabelen in een klasse te zetten zodat hij er andere klassen van kan afleiden. Op deze manier moet hij de code geen twee keer schrijven.



Figuur 6: Template van sectie "Gegevens van de erfgenaam, legataris of begiftigde die de erfenis verworpen hebben"

Eerst wordt de DTO van de basisgegevens aangemaakt. Deze krijgt de naam 'BasicIdentityDataDto'. De klasse beschikt over verschillende variabelen. Deze variabelen hebben elk hun *getters* en *setters*. De *getters* dienen om data van een object op te vragen. De *setters* daarentegen worden gebruikt om waardes aan de variabelen te koppelen.

```java
package be.argeus.msp.server.report.dto;

import java.util.Date;

import org.apache.commons.lang3.StringUtils;

public abstract class BasicIdentityDataDto {

  private String name = "";
  private String nationalNumber = "";
  private String profession = "";
  private String placeOfBirth = "";
  private Date dateOfBirth;

  public final String getName() {
    return StringUtils.defaultString(name);
  }

  public final void setName(String name) {
    this.name = name;
  }

  public final String getNationalNumber() {
    return StringUtils.defaultString(nationalNumber);
  }

  public final void setNationalNumber(String nationalNumber) {
    this.nationalNumber = nationalNumber;
  }

  public final String getProfession() {
    return StringUtils.defaultString(profession);
  }

  public final void setProfession(String profession) {
    this.profession = profession;
  }

  public final String getPlaceOfBirth() {
    return StringUtils.defaultString(placeOfBirth);
  }

  public final void setPlaceOfBirth(String placeOfBirth) {
    this.placeOfBirth = placeOfBirth;
  }

  public final Date getDateOfBirth() {
    return dateOfBirth;
  }

  public final void setDateOfBirth(Date dateOfBirth) {
    this.dateOfBirth = dateOfBirth;
  }
}
```

Figuur 7: BasicIdentityDataDto-klasse

Wat er misschien opvalt in figuur 8 is dat er gebruik gemaakt wordt van 'StringUtils'. Deze *library* wordt gebruikt bij het *getten* van stringvariabelen.

De methode 'defaultString' zorgt ervoor dat er nooit een *null* teruggegeven kan worden. Het gaat kijken of de string die hij moet retourneren *null* is en als hij dat is dan zal de *library* deze omzetten naar een lege string. Dit wordt gedaan om fouten tegen te gaan in verband met *nullpointer exceptions*.

Nu de klasse waarvan er gaat afgeleid worden gereed is, kan de DTO van de sectie "Gegevens van de overledene" aangemaakt worden. Deze krijgt de naam 'DeceasedIdentityDto' en wordt afgeleid van de klasse 'BasicIdentityDataDto'. De klasse moet twee extra variabelen hebben, namelijk: de geboorteplaats en -datum. Deze variabelen hebben uiteraard ook weer de bijhorende *getters* en *setters*.

```
1  package be.argeus.msp.server.report.dto;
2
3  import java.util.Date;
4
5  import org.apache.commons.lang3.StringUtils;
6
7  public class DeceasedIdentityDto extends BasicIdentityDataDto {
8
9    private String placeOfDeath;
10   private Date dateOfDeath;
11
12   public final String getPlaceOfDeath() {
13     return StringUtils.defaultString(placeOfDeath);
14   }
15
16   public final void setPlaceOfDeath(String placeOfDeath) {
17     this.placeOfDeath = placeOfDeath;
18   }
19
20   public final Date getDateOfDeath() {
21     return dateOfDeath;
22   }
23
24   public final void setDateOfDeath(Date dateOfDeath) {
25     this.dateOfDeath = dateOfDeath;
26   }
27
28 }
29
```

Figuur 8: DeceasedIdentityDto-klasse

3.3.2.2 Factory

Voor elke DTO moet er een aparte *factory* gemaakt worden. Alle logica die nodig is om een bepaalde DTO aan te maken, staat in die *factoryklasse*. Zo blijft de code zo goed mogelijk gescheiden van elkaar en blijft het overzichtelijk.

Eerst dient er een *factory* aangemaakt te worden voor de klasse 'BasicIdentityDataDto'. Deze zal instaan voor het invullen van de variabelen van die klasse en krijgt de naam 'BasicIdentityDataDtoFactory'.

Met de functie 'fill' zal de *factory* een meegegeven DTO, van de klasse 'BasicIdentityDataDto', opvullen. Deze functie krijgt de variabelen 'declarationClient' en 'locale' mee. Met deze variabelen zal de *factory* de nodige data uit de *entity* halen en aan de variabelen van de meegegeven DTO koppelen.

```
1  package be.argeus.msp.server.factory;
2
3  import java.util.Locale;
7
8  public final class BasicIdentityDataDtoFactory {
9
10   public static void fill(final BasicIdentityDataDto dto, final DeclarationClient declarationClient, final Locale locale) {
11     dto.setNationalNumber(declarationClient.getNationalNumber());
12     dto.setProfession(declarationClient.getProfession());
13     dto.setPlaceOfBirth(declarationClient.getBirthPlace(locale.getLanguage()));
14   }
15
16   private BasicIdentityDataDtoFactory() {
17     throw new IllegalStateException("Method factory class");
18   }
19
20 }
21
```

Figuur 9: BasicIdentityDataDtoFactory-klasse

Nu kan de *factory* voor de klasse 'DeceasedIdentityDto' aangemaakt worden. Deze bevat de methode 'create'. De methode 'create' krijgt een context variabele mee. Wat de context concreet is wordt nog volledig uitgelegd in de volgende stap.

Het doel van deze functie is het creëren van een object van de klasse 'DeceasedIdentityDto'. Eerst worden er verschillende objecten opgehaald uit de context.

Daarna worden deze objecten gebruikt om de nodige data op te halen en aan de correcte variabele van de DTO te koppelen. Ook wordt er gebruik gemaakt van de 'BasicIdentityDataDtoFactory' om de basisgegevens in te vullen.

Eenmaal de DTO is opgevuld met alle data zal de methode de DTO retourneren.

Opmerkelijk is dat elke *factoryklasse* een *constructor* heeft. Deze zal een *exception* opgooien wanneer er een *factory* aangemaakt wordt. Dit omdat de *factories, method factories* zijn. Van deze soort *factory* mag er geen object aangemaakt worden.

```java
package be.argeus.msp.server.factory;

import java.util.Locale;

public final class DeceasedIdentityDtoFactory {

  public static DeceasedIdentityDto create(final DeclarationContext context) throws ProcessingException {
    final DeclarationClient declarationClient = context.getDeceasedDeclaration();
    final Client deceased = context.getClient();
    final Locale locale = context.getDocumentConfiguration().getLocale();

    final DeceasedIdentityDto dto = new DeceasedIdentityDto();

    dto.setName(StringUtils.defaultString(deceased.getFullName()));
    BasicIdentityDataDtoFactory.fill(dto, declarationClient, locale);
    dto.setDateOfBirth(deceased.getDateOfBirth());
    dto.setPlaceOfDeath(StringUtils.defaultString(declarationClient.getPlaceOfDeath(locale.getLanguage())));
    dto.setDateOfDeath(declarationClient.getDateOfDeath());

    return dto;
  }

  private DeceasedIdentityDtoFactory() {
    throw new IllegalStateException("Method factory class");
  }

}

```

Figuur 10: DeceasedIdentityDtoFactory-klasse

3.3.2.3 DeclarationContext

In het vorige gedeelte (2.6.2.2) werd de context aangehaald. Deze context is er om alle data te centraliseren. Deze context gaat alle data ophalen en opslaan voor dat het document gegenereerd wordt. Deze context gaat dus gebruik maken van de diverse *factories* die er aangemaakt zijn. De DTO's die hij ophaalt zal hij bijhouden zodat deze later gebuikt kunnen worden in de verschillende *documentparts*.

De 'DeclarationContext' zorgt ook voor het ophalen van data uit de database en frontend. Deze data wordt gebruik door de *factories* om de verschillende DTO's te creëren.

3.3.2.4 DeceasedIdentityDocumentPart

Elke sectie van het aangiftedocument heeft zijn eigen 'DocumentPart' gekregen. Een 'DocumentPart' is een interface waar van afgeleid kan worden. Voor de sectie die geïmplementeerd wordt tijdens deze tutorial wordt er de klasse 'DeceasedIdentityData' aangemaakt. Deze zal dus instaan voor het genereren van de sectie 'Gegevens van de overledene'.

```
13 public class DeceasedIdentityDocumentPart implements IDocumentPart {
14
15   private final DeceasedIdentityDto deceasedIdentityData;
16   private final boolean isExtraInfo;
17
18   public DeceasedIdentityDocumentPart(DeceasedIdentityDto deceasedIdentityData, boolean isExtraInfo) {
19     this.deceasedIdentityData = deceasedIdentityData;
20     this.isExtraInfo = isExtraInfo;
21   }
22
23   @Override
24   public void build(IDocumentBuilder b, Locale locale) throws ProcessingException {
25     b.addPart(new HeadingPart(2, TEXTS.get(locale, "Declaration-IdentityOfDeceased")));
26     if (isExtraInfo) {
27       b.addParagraphOfText(TEXTS.get(locale, "Declaration-IdentityOfDeceasedInfo")).setStyle(DeclarationStyles.EXTRA_INFO);
28     }
29
30     b.addTable(locale, tbd -> {
31       tbd.addRow(rb -> {
32
33         rb.addCell(TEXTS.get(locale, "Declaration-FirstNamesAndLastName")) //
34           .setWeight(3) //
35           .setMargins(3) //
36           .setStyle(DeclarationStyles.TABLE_DESCRIPTION);
37
38         rb.addCell(deceasedIdentityData.getName())//
39           .setStyle(DeclarationStyles.TABLE_INPUT_VALUE_RIGHT) //
40           .setWeight(8);
41       });
42
43       tbd.addRow(rb -> {
44
45         rb.addCell(TEXTS.get(locale, "NationalNumber")) //
46           .setWeight(3) //
47           .setMargins(3) //
48           .setStyle(DeclarationStyles.TABLE_DESCRIPTION);
49
50         rb.addCell(deceasedIdentityData.getNationalNumber())//
51           .setStyle(DeclarationStyles.TABLE_INPUT_VALUE_RIGHT) //
52           .setWeight(8);
53       });
54
55       tbd.addRow(rb -> {
56
57         rb.addCell(TEXTS.get(locale, "Profession")) //
58           .setWeight(3) //
59           .setMargins(3) //
60           .setStyle(DeclarationStyles.TABLE_DESCRIPTION);
61
62         rb.addCell(deceasedIdentityData.getProfession())//
63           .setStyle(DeclarationStyles.TABLE_INPUT_VALUE_RIGHT) //
64           .setWeight(8);
65       });
66
67       tbd.addRow(rb -> {
68
69         rb.addCell(TEXTS.get(locale, "Declaration-BirthplaceAndDate")) //
70           .setWeight(3) //
71           .setMargins(3) //
72           .setStyle(DeclarationStyles.TABLE_DESCRIPTION);
73
74         rb.addCell(deceasedIdentityData.getPlaceOfBirth())//
75           .setStyle(DeclarationStyles.TABLE_INPUT_VALUE_RIGHT) //
76           .setWeight(2);
77
78         DeclarationDocumentBuilderUtil.addDateToRow(locale, rb, deceasedIdentityData.getDateOfBirth());
79       });
80
```

```
81        tbd.addRow(rb -> {
82
83          rb.addCell(TEXTS.get(locale, "Declaration-PlaceAndDateOfDeath")) //
84            .setWeight(3) //
85            .setMargins(3) //
86            .setStyle(DeclarationStyles.TABLE_DESCRIPTION);
87
88          rb.addCell(deceasedIdentityData.getPlaceOfDeath())//
89            .setStyle(DeclarationStyles.TABLE_INPUT_VALUE_RIGHT) //
90            .setWeight(2);
91
92          DeclarationDocumentBuilderUtil.addDateToRow(locale, rb, deceasedIdentityData.getDateOfDeath());
93        });
94      });
95
96      b.addEmptyParagraph();
97      b.addSection(SectionEndType.CONTINUOUS);
98    }
99
100 }
101
```

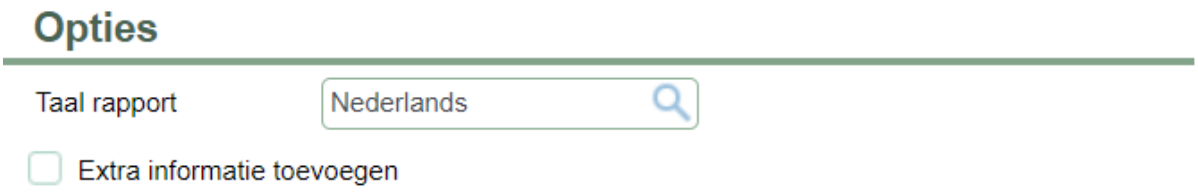
Figuur 11: DeceasedIdentityDocumentPart-klasse

Deze klasse heeft slechts één functie, genaamd 'build'. In deze functie wordt de sectie gecreëerd en wordt de data, vanuit de DTO, ingevuld. Het aanmaken van deze sectie wordt gedaan met behulp van de *library* 'IDocumentBuilder', die al eerder was geschreven binnen Argeüs.

Eerst wordt er een 'HeadingPart' toegevoegd aan de sectie. Deze zal een hoofding van niveau twee toevoegen. Er wordt ook een *string* meegegeven. Dit zal de tekst zijn van de hoofding.

Daarna wordt er een check gedaan of er extra info getoond moet worden bij de sectie. Dit is een optie die de gebruiker kan aanzetten in de frontend. Als hij deze heeft aangevinkt, zal er meer informatie getoond worden bij elke sectie.

Als de gebruiker de optie 'meer info tonen' heeft aangevinkt, zal er in het hele document extra info komen bij de verschillende secties. Deze informatie gaat over wat er precies ingevuld wordt in de sectie en wat dit betekent. Om deze info toe te voegen aan de sectie, wordt er gebruik gemaakt van de functie 'addParagraphOfText'. Deze zal een paragraaf toevoegen met als inhoud, de meegegeven tekst.



Figuur 12: Optie om extra info te tonen

Verder wordt er ook nog een tabel toegevoegd. In deze tabel komt de data terecht. Met behulp van lambda's wordt de tabel opgebouwd. Er worden rijen toegevoegd met de functie 'addRow'. Deze rijen kunnen op hun beurt weer cellen toevoegen met de functie 'addCell'.

De cellen worden opgevuld met vertaalde *strings* en data uit de DTO. Met behulp van de *getters* weten we precies welke data we ophalen en op welke plaats deze terecht komen in de sectie. De

cellen kunnen ook nog wat extra configuratie meekrijgen. Zo wordt er bijvoorbeeld bij bepaalde cellen een marge en een stijl toegepast. Ook kan ingesteld worden hoeveel kolommen zo een cel inneemt.

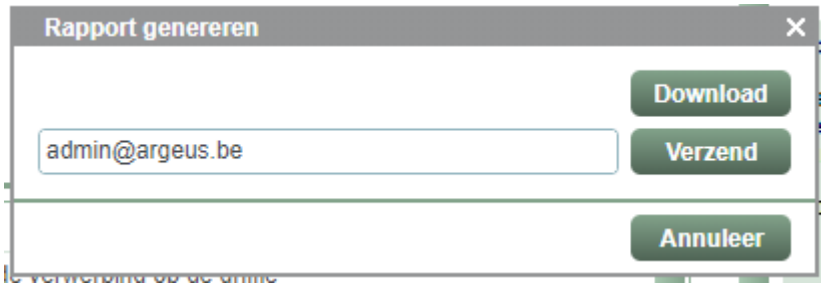
Er wordt ook gebruik gemaakt van een *utilityklasse* genaamd 'DeclarationDocumentBuilderUtil'. Met behulp van deze *utilityklasse* en de 'addDateToRow' functie wordt er een datumrij toegevoegd aan de tabel.

Na het opbouwen van de tabel wordt er nog een witregel toegevoegd. Dit is mogelijk door de functie 'addEmptyParagraph' aan te spreken. Deze zal een lege paragraaf genereren.

Tenslotte wordt er nog een sectie-einde toegevoegd om de secties af te ronden.

3.3.2.5 Gegenereerd resultaat

Als de gebruiker nu in de frontend op de downloadknop drukt zal de implementatie een aangifte genereren.



Figuur 13: Downloadformulier aangifte

Het eerste wat de applicatie zal gaan doen is het ophalen en *setten* van alle data in de context. Vervolgens zal de applicatie alle secties gaan genereren en zal er een Word-document gedownload worden.

Identiteit van de overledene

voornamen en achternaam							A. R.
Rijksregister- of bisnummer							58.10.24-186.91
Beroep							Schepen - Medewerkster Europees Parlement
geboorteplaats en -datum	Hasselt (3500)	dag	1	maand	10	Jaar	1958
plaats en datum van overlijden	Hasselt (3500)	dag	14	maand	11	Jaar	2018

Figuur 14: 'Identiteit van de overledene' gegenereerde sectie (Nederlands)

Identité du défunt

Si le lieu de naissance du défunt ou le lieu du décès est à l'étranger, entrez également le pays concerné.

prénom et nom							A. R.
Numéro de Registre national ou bis							58.10.24-186.91
Métier							Schepen - Medewerkster Europees Parlement
lieu et date de naissance	Hasselt (3500)	jour	1	mois	10	Année	1958
lieu et date du décès	Hasselt (3500)	jour	14	mois	11	Année	2018

Figuur 15: 'Identiteit van de overledene' gegenereerde sectie (Frans + extra info)

3.3.3 Voor- en nadelen

Het nieuwe systeem brengt heel wat voordelen met zich mee en lost heel wat problemen op.

- Het eerste voordeel van de nieuwe implementatie is dat de secties gemakkelijk te onderhouden zijn. Ontwikkelaars kunnen gemakkelijk tabellen toevoegen, paragrafen verwijderen, cellen aanpassen, … Ze bepalen de opbouw van het document volledig via code. Dit is handig voor moesten er andere modellen geïmplementeerd worden omdat de secties volledig herbruikbaar zijn.
- Het tweede voordeel is dat het nieuwe systeem perfect analyseerbaar is. Er is een duidelijke structuur opgebouwd. De ontwikkelaar kan perfect het programma gaan *debuggen* om te kijken waar het fout gaat omdat alles nu via code gaat.
- Het derde voordeel van het nieuwe systeem is dat de vertalingen centraal gebeuren. Er moeten niet twee verschillende secties gecreëerd worden (Nederlands en Frans). De vertalingen gebeuren in de sectie zelf. Zo wordt er zeker niks vergeten als er aanpassingen gedaan zouden moeten worden.
- Het laatste voordeel van het nieuwe systeem is dat het terug overzichtelijk is. In het oude systeem waren er veel klassen met een duizendtal lijnen code, alles stond in één klasse. Nu is er gebruik gemaakt van verschillende klassen waarover de code verdeeld is.

3.4 Resultaten

De Word-documentgenerator kan nu 42 van de 45 secties genereren. De generator kan ze in het Nederlands of Frans genereren afhankelijk van de keuze van de gebruiker. Ook kan de gebruiker er voor kiezen om meer informatie te tonen bij elke sectie.

Er zijn drie secties die niet volledig gegenereerd worden. Het systeem zal wel de tabellen, hoofdingen en paragrafen toevoegen aan het document maar zal geen data invullen. Dit omdat er nog een aantal nieuwe features gemaakt moeten worden in de applicatie zelf. Dit viel buiten de scope dus moest er enkel voor gezorgd worden dat de generator de *boilerplate* kon genereren.

De 42 geïmplementeerde secties zijn ook grondig getest geweest door een *Quality Assurance Engineer* binnen Argeüs. De fouten die uit de testen kwamen zijn volledig opgelost en zijn klaar om uitgerold te worden.

Argeüs heeft nu een systeem dat de problemen die ze hadden oplost. Het nieuwe systeem is goed onderhoudbaar en analyseerbaar. Dit is voor velen binnen Argeüs een positief iets. Ook de klanten zullen blij zijn. Het nieuwe systeem is namelijk sneller als het oude. Ze zullen minder lang moeten wachten vooraleer dat het document gegenereerd is.

In onderstaande tabel staan de grootten van de documenten en de snelheden van hoelang het duurt om een aangifte te genereren en downloaden. Voor deze resultaten werd hetzelfde dossier gebruikt.

Tabel 3: Grootte- en snelheidsresultaten

	Snelheid (seconden)	Grootte (kB)
Oude aangiftesysteem	5.24	76,0
Nieuwe aangiftesysteem	2.46	19,0

4 Besluit

De stage is na drie maanden ten einde gelopen. Het was een intensieve en leerrijke stage waar ik veel heb bijgeleerd en mezelf meer ontwikkeld heb. Daarom zal ik de verschillende vlakken waarin ik gegroeid ben toelichten.

4.1 Programmeervaardigheden

Tijdens de stage heb ik heel wat ervaring kunnen opdoen tijdens het ontwikkelen van het nieuwe systeem.

Door te werken met *pull requests* werd mijn code elke keer gereviewd door een andere ontwikkelaar binnen Argeüs en gaven ze me altijd correcte feedback. Door deze feedback te verwerken werd mijn code *cleaner* en *cleaner*. De feedback die ik gehad heb zal ik dus nog voor de rest van mijn carrière gebruiken.

Ook had ik zelf nog nooit gewerkt met een Eclipse Scout project. De opbouw van de frontend was iets dat ik nog nooit gezien had. Door een beetje te onderzoeken hoe het in elkaar zit, had ik veel meer inzicht op het project.

Ik had ook nog nooit met een vertalingssysteem gewerkt. Zelf had ik totaal geen idee hoe dit eigenlijk in elkaar zat. Al begon ik te snappen hoe het precies werkt en is het eigenlijk totaal niet moeilijk. Toch is het handig om te weten voor moest ik ooit zelf een applicatie maken die in verschillende talen beschikbaar moet zijn.

4.2 Persoonlijk vlak

Doorheen de stageperiode heb ik veel vergaderd, gesproken voor een groep mensen en anderen moeten helpen als ze vastzaten op een stuk waar ik iets van kende. In het begin kon ik nooit echt goed mezelf verwoorden. Ik kon problemen of oplossingen slecht formuleren. Door zoveel samen te zitten is dat wel echt veranderd.

Ook stelde ik vroeger niet zo snel een vraag als ik even vast zat. Ik wou het altijd liever zelf proberen op te lossen en bleef daardoor soms te lang aan iets vastzitten. Hier is ook verandering ingekomen. Als ik vast zat en het duurde mezelf langer dan 20 minuten om op te lossen, ging ik hulp vragen aan een collega. Ik heb geleerd dat het niet erg is om iets te vragen en dat het soms de beste oplossing is.

4.3 Sociaal vlak

Op sociaal vlak ben ik ook wel wat veranderd. In het begin dacht ik namelijk dat er op een kantoor altijd een zakelijke sfeer moest zijn en gedroeg mij dan ook zo. Naarmate de stage voorbijging begon er toch wat sfeer in te komen en was iedereen wat losser en socialer. Mijn blik op de sfeer binnen een bedrijf is dus wel wat veranderd.

4.4 Andere

Tijdens het uitwerken van de aangiftegenerator heb ik natuurlijk ook heel wat informatie opgenomen in verband met aangiftes en successierechten. Dit is allemaal nieuwe informatie voor mij geweest die eigenlijk nog wel best handig is om te weten. Die kennis kan ik gebruiken voor moest ik ooit zelf een aangifte doen.

II. Onderzoek topic

1 Vraagstelling

Argeüs biedt verschillende softwarepakketten aan. De frontend van deze software wordt opgebouwd met behulp van het Eclipse Scout 5-framework. Deze versie van Eclipse Scout bereikt stillaan zijn einde qua ondersteuning en zijn technische opbouw sluit ook minder goed aan bij de structuur die wenselijk is in nieuwe projecten.

Op de markt is er een grote diversiteit aan frontend- en applicatieframeworks. Argeüs wil graag weten welk framework op dit moment de beste keuze is voor zijn toekomstige projecten. Argeüs wil een vergelijkende studie zien waaruit kan worden afgeleid wat de beste keuze is voor het bedrijf. Een goed framework kan immers heel wat voordelen met zich meebrengen, zowel technisch als economisch.

Het doel van het onderzoek is het framework te vinden dat het beste bij Argeüs past. Om dit te weten te komen wil Argeüs graag een vergelijking zien van diverse frameworks. De frameworks worden vergeleken aan de hand van verschillende aspecten. Testbaarheid, kostprijs, documentatie en ondersteuning vanuit de community zijn hier voorbeelden van.

2 Methode van onderzoek

Het onderzoek begint met het analyseren van de markt. Er wordt eerst gekeken naar welke frameworks er allemaal zijn op de markt.

Eenmaal er een tiental frameworks gevonden zijn, worden deze in een globale samenvatting verwerkt om een eerste overzicht te krijgen van de aspecten waarin ze verschillen. Zo kan iedereen binnen Argeüs direct een beeld krijgen van alle frontendframeworks en waar deze in uitblinken.

Nadat er een globaal overzicht gecreëerd is, wordt er samengezeten met alle collega's binnen Argeüs. Zo worden alle aspecten die het bedrijf belangrijk vindt duidelijk. Aan de hand van deze aspecten worden er een vijftal frameworks gekozen die het best aansluiten bij de keuzes van Argeüs.

Deze vijf frameworks worden verder uitgewerkt. De aspecten die Argeüs belangrijk vindt, worden bij elk van de vijf frameworks goed toegelicht en er wordt vermeld waarom dat aspect bij dat bepaalde framework past.

Eens deze vijf frameworks uitgewerkt zijn, worden er twee frameworks uitgekozen die het beste bij Argeüs passen aan de hand van de sterkte van de aspecten bij ieder framework. Van deze frameworks wordt er een prototype uitgewerkt. Door middel van dit prototype zullen de diverse aspecten naar boven komen. Zo kan Argeüs direct een beeld krijgen van de mogelijkheden binnen dat framework.

3 Resultaten

3.1 Literatuurstudie

3.1.1 Introductie

In deze literatuurstudie worden er drie bronnen besproken en vergeleken. Deze bronnen zijn artikels waarin drie frameworks besproken worden, namelijk: React, Vue en Angular.

De bronnen zullen eerst kort besproken worden. Er zal toegelicht worden wat er in dit artikel is uitgeschreven en onderzocht is. Daarna wordt de effectieve inhoud van deze bronnen met elkaar vergeleken. Spreken ze elkaar toe of tegen?

3.1.2 Bron 1: Angular vs React vs Vue: Which Framework to Choose in 2019 [2]

Het eerste wat er in deze bron besproken wordt is de geschiedenis van elk framework. Zo worden er bijvoorbeeld al drie verschillende onderwerpen besproken: hoe is het ontstaan, hoe is de groei van de populariteit, welke licentie hangt er aan vast en hoe vaak wordt het framework gevraagd op de markt?

In het tweede gedeelte van deze bron wordt de community en de hoeveelheid updates besproken. Dit wordt vergeleken aan de hand van de GitHub *repositories* van de frameworks. Er wordt gekeken naar de hoeveelheid *watchers, stars, forks, commits* en *contributors*.

	Angular	React	Vue
# Watchers	3.3k	3.7k	5.7k
# Stars	43k	71k	122k
# Forks	11k	16k	17k
# Commits in last month	446	339	81
# Contributors	798	1.8k	240

Figuur 16: Bron 1 - community vergelijking

Het derde gedeelte van dit artikel gaat over de migraties van de frameworks. Er wordt besproken hoe gemakkelijk of moeilijk het is om het framework up te daten in een project. Het is handig dat dit besproken wordt omdat bedrijven en ontwikkelaars niet graag hebben dat hun code opeens niet meer werkt na het framework te updaten.

In het vierde gedeelte van dit artikel worden er verschillende aspecten van de frameworks besproken. De aspecten die besproken worden zijn: de grootte van het framework, de componenten en de leercurve.

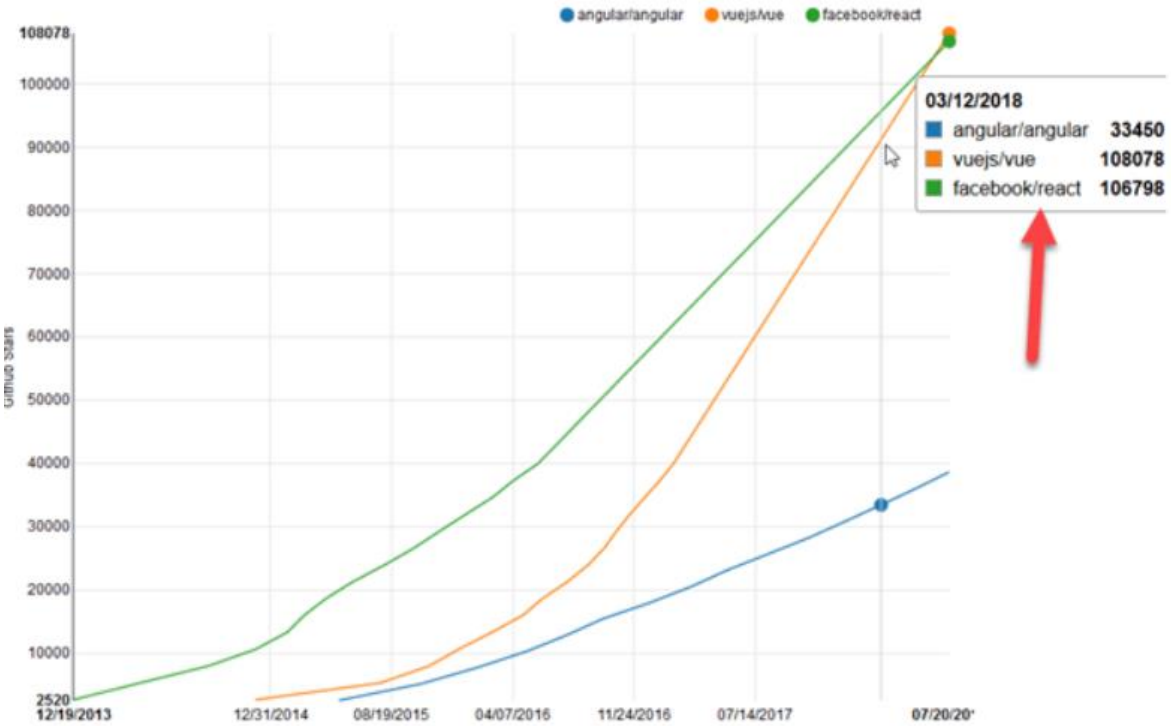
Tenslotte rond Shaumik Daityari zijn artikel af door te bespreken welk framework hij zou aanraden afhangend van de situatie.

3.1.3 Bron 2: React vs. Angular vs. Vue.js: A Complete Comparison Guide [3]

Het tweede artikel dat besproken wordt is anders opgebouwd dan het eerste. In dit artikel worden er eerst een aantal vragen opgesteld waar doorheen het artikel antwoorden op worden gegeven. Een vraag die bijvoorbeeld gesteld wordt is: 'Is het framework volwassen genoeg om grootschalige applicaties te bouwen ?'.

In de eerste sectie van het artikel wordt de geschiedenis van de frameworks besproken. Hier bespreken ze door wie het ontwikkeld is, door wie het onderhouden wordt en welke applicaties er al gebouwd zijn met behulp van dit framework.

Daarna wordt de populariteit en de markttrend van de frameworks besproken. Dit wordt gedaan met behulp van een grafie die het aantal GitHub-sterren toont per framework. Door naar deze cijfers te kijken wordt er bepaald hoe populair een framework is en hoe goed het in de markt ligt.



Figuur 17: Bron 2 - Populariteit grafiek

In de derde sectie wordt de *support* en groei van de frameworks besproken. Er wordt ook in vermeld waarom deze aspecten nu belangrijk zijn binnen een framework. Er wordt aangehaald dat frameworks die onderhouden worden door grote spelers zoals Google en Facebook, meer updates en *releases* uitrollen tegen over een framework zoals Vue dat onderhouden wordt door een community.

Vervolgens wordt er besproken of het gemakkelijk is om ontwikkelaars te vinden die bezig zijn met een bepaald framework. Er wordt aangehaald dat de syntax een grote rol speelt voor ontwikkelaars. De auteur haalt aan dat het gemakkelijker is om ontwikkelaars te vinden die Angular gebruiken. Dit omdat veel ontwikkelaars liever in Typescript programmeren dan pure Javascript zoals in React.

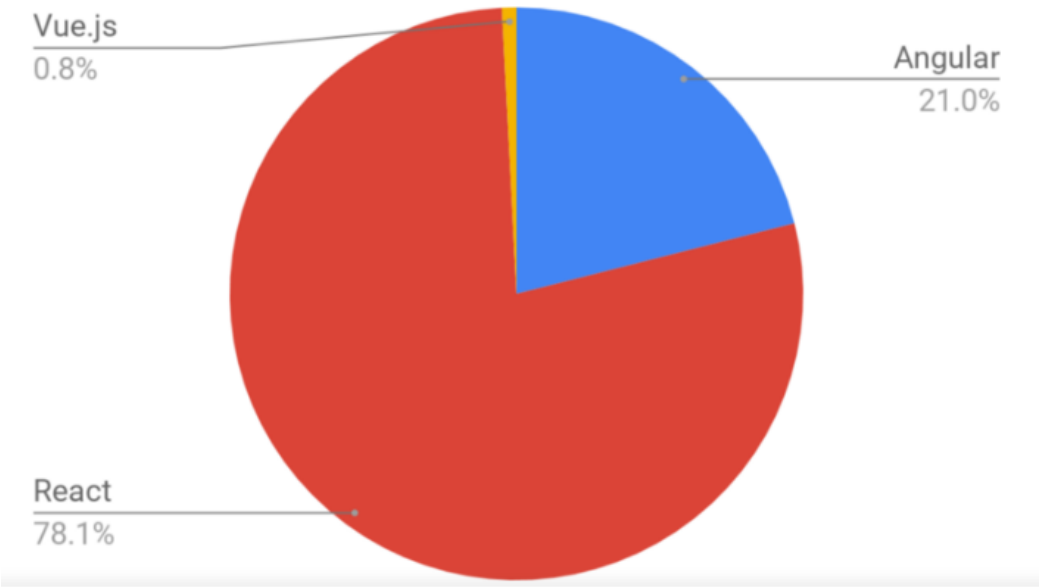
Na het bespreken van de programmeertalen wordt er gesproken over het ontwikkelen van *'native apps'*. Er wordt vermeld welke mogelijkheden er zijn om *native* applicaties te bouwen met behulp van het framework en wat de ontwikkelaar hiervoor kan gebruiken. Zo wordt er bijvoorbeeld gezegd dat React een *libary* 'React Native' heeft en dat ontwikkelaars deze kunnen gebruiken om applicaties te ontwikkelen.

Ook in dit artikel worden de leercurves van de frameworks vermeld. Er wordt besproken waarom een framework een moeilijke of een gemakkelijke leercurve heeft aan de hand van verschillende redenen.

Tenslotte wordt er vermeld welke frameworks de auteur aanraad in verschillende situaties. Als een ontwikkelaar bijvoorbeeld een framework wilt met veel flexibiliteit dan zou hij voor React moeten kiezen. Angular wordt aangeraden wanneer er geprogrammeerd wilt worden in Typescript.

3.1.4 Bron 3: React vs Angular vs Vue.js — What to choose in 2019? [4]

Het laatste artikel is eerder *straight to the point*. Het artikel begint met het bespreken van hoe snel de frameworks ontwikkelen en hoe dit invloed heeft op de concurrentie en ontwikkelaars. Daarna wordt er gekeken naar de cijfers qua job aanbiedingen. Wat wordt er gevraagd op de markt? Dit wordt geanalyseerd aan de hand van de website 'Indeed.com'.



Figuur 18: Bron 3 - jobaanbiedingen per framework

Na het bespreken van de jobaanbiedingen, wordt er van elk framework toegelicht wat de voor- en nadelen ervan zijn. Zo wordt er bijvoorbeeld aangehaald dat een voordeel van Angular het werken in Typescript is. Een nadeel dat ze van Angular aanhaalden is dat het relatief slechtere performantie levert.

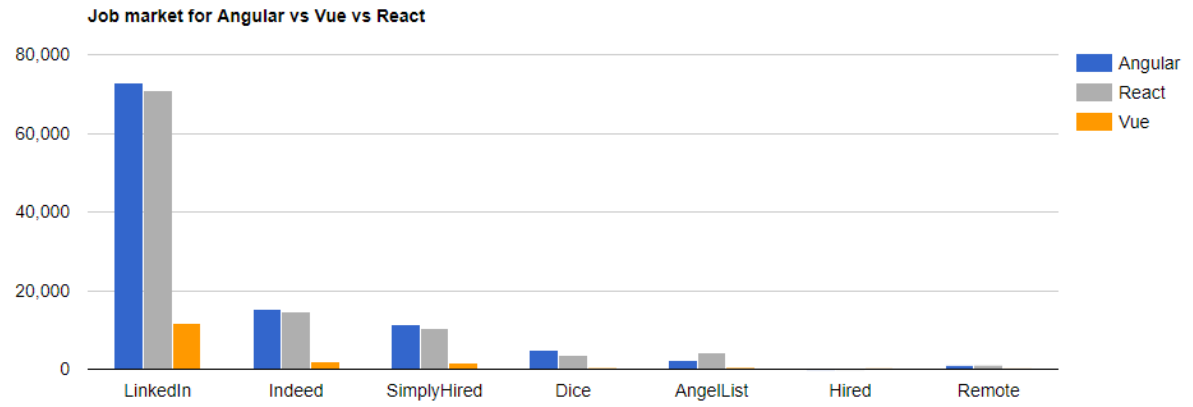
De auteur sluit zijn artikel af door mee te delen dat er geen substantieel verschil is tussen de frameworks. Hij zegt dat het gewoon wat tijd kost om gewoon te geraken aan een framework. Er wordt niet aangeraden welk framework in welke situatie past.

3.1.5 Inhoudelijke vergelijking van de bronnen

In de drie bronnen zijn er heel veel aspecten aan bod gekomen. Een aantal van die aspecten kwamen terug in de drie bronnen. Deze aspecten zullen met elkaar vergeleken worden.

Het eerste aspect dat twee van de drie bronnen besproken hebben is de populariteit van de frameworks. In de eerste en tweede bron werd de populariteit van de frameworks geanalyseerd aan de hand van de GitHub-sterren. Beide bronnen gaan akkoord dat React en Angular al een hele tijd de grote spelers zijn binnen de frontendmarkt en dat Vue bezig is met een grote opmars.

Ook werd er in twee bronnen onderzocht naar welk framework nu het meeste gevraagd werd op de markt. Bron 1 heeft hier een uitgebreidere analyse voor gedaan. Hier werd er geanalyseerd aan de hand van verschillende websites met jobaanbiedingen in plaats van naar één website zoals bij bron 3. Dit heeft zeker invloed gehad op de cijfers. Volgens bron 3 zou React het meest gevraagd zijn maar in bron 1 is er duidelijk te zien dat Angular toch de boven hand heeft. Hierin verschillen de bronnen van mening.



Figuur 19: Bron 1 - jobaanbieding analyse

In bron 1 en 2 werden ook de communities van de frameworks vergeleken. Beiden bronnen komen overeen dat React en Angular zeker een groot voordeel hebben omdat ze ondersteund worden door grote bedrijven zoals Facebook en Google. Ook zijn ze het eens over de groei van de Vue-community.

In alle bronnen wordt de grootte van Vue geanalyseerd en wat hier de voordelen van zijn. Ze zijn akkoord dat Vue een hele kleine grootte heeft en dat dit zeker een invloed kan hebben bij ontwikkelaars afhangend van wat voor soort applicatie ze gaan bouwen. Ze zijn akkoord dat Vue een goede keuze zou zijn als de ontwikkelaar kleine applicaties zouden bouwen, ook wel *microapps* genoemd.

Angular wordt door alle drie de bronnen aangeraden als de ontwikkelaars applicaties wilt programmeren in Typescript. Dit omdat Angular een exceptionele *support* levert voor het programmeren in Typescript.

Ook de leercurve van de frameworks wordt in alle drie de bronnen vermeld. Bron 1 en 2 zijn akkoord dat Angular een relatief moeilijke leercurve heeft. Als reden geven ze mee dat als een ontwikkelaar pas zou beginnen aan Angular hij nog wat lessen moet nemen in verband met Typescript. Ook omdat Angular een aantal dingen heeft die het op zijn eigen manier doet. In bron 3 zeggen ze dat Angular relatief makkelijk aan te leren is door de goede documentatie die geleverd wordt.

De leercurve van React vinden alle bronnen gemakkelijk. Als reden geven ze mee dat React een puur Javascript framework is. Als de ontwikkelaar kennis heeft van Javascript zal hij zich heel snel thuis voelen binnen het framework. In bron 2 wordt wel nog meegegeven dat React soms wel hekelpuntjes heeft omdat het net zoals Angular een aantal dingen op zijn eigen manier doet.

Alle drie de bronnen komen ook overeen dat Vue de gemakkelijkste leercurve heeft. Het framework is *straight forward* omdat het niet veel *overhead* heeft. Ontwikkelaars kunnen met Vue een applicatie bouwen op de *old-fashioned* Javascript manier waar HTML en Javascript zich in één file bevinden.

In bron 2 en 3 halen ze beide aan dat Angular het aangeraden framework is voor grootschalige applicaties. Dit vinden ze omdat de structuur en opbouw van Angular meer overzichtelijk blijft tegenover een framework zoals Vue.

De bronnen hebben allemaal een uniek slot. De derde bron heeft wel het kortste slot hierin sluit de auteur zijn artikel af door te zeggen dat elk framework evenwaardig is en dat het gewoon tijd kost om gewend te worden aan het framework. Er worden geen richtlijnen gegeven in welke situaties elk framework past.

Bron 1 en 2 sluiten wel af door te zeggen in welke situaties de frameworks passen. Dit biedt een beter slot van het artikel. Beide artikels vermelden dat Angular het meest volwassen framework is en dat het het beste framework is voor ontwikkelaars die graag in Typescript programmeren. Het object georiënteerd programmeren is in Angular ook het sterkst aanwezig volgens beide bronnen.

Volgens bron 1 is React net volwassen genoeg om volwaardige applicaties mee te creëren. Dit omdat er al heel veel functionaliteiten in React zitten door de vele contributies van de communities. Beide bronnen zeggen dat React een goed framework zou zijn voor beginners die van flexibiliteit en Javascript houden.

Bron 1 vermeld dat Vue nieuw is in de 'arena' van de frontendframeworks. Dit door de groeiende populariteit van het framework. Beide bronnen raden Vue aan als framework voor ontwikkelaars die houden van *clean* code en pure Javascript en HTML. Dit omdat Vue weinig *overhead* heeft en eerder *straight to the point* is.

3.1.6 Conclusie

Alle drie de frameworks hebben hun sterke en zwakke kanten. Zo is er voor iedere situatie wel een meest geschikt framework. De keuze van het framework hangt dus van project tot project af.

3.1.7 Reflectie

Door een literatuurstudie te doen naar de drie grootste frontendframeworks heb ik al een veel beter zicht op de frameworks. Zelf had ik al gewerkt met Angular maar React en Vue waren onbekend terrein voor mij.

Ik heb dan ook heel wat informatie kunnen opdoen over de frameworks. Zelf wist ik niet hoe React en Vue eigenlijk in elkaar zaten qua structuur. Ook ben ik de vele voor- en nadelen van deze frameworks te weten gekomen.

Het was ook interessant om te zien welke frameworks bij welke situaties passen volgens verschillende mensen. Deze info zal ik zeker meenemen naar mijn onderzoek toe om zo tot het beste resultaat te komen.

3.2 Fase 1 - De diverse frontendframeworks

3.2.1 Inleiding

Het onderzoek wordt gestart met het onderzoeken van welke frameworks er allemaal beschikbaar zijn op de markt. Hiervoor zijn er verschillende websites geraadpleegd. Uit deze bronnen zijn een tiental frameworks gekomen. [5] [6]

Deze frameworks zullen vergeleken worden aan de hand van verschillende aspecten zoals bijvoorbeeld: testbaarheid en communitygrootte. Er zal verder in dit eindwerk nog besproken worden waarom deze aspecten gekozen worden.

Tabel 4: Overzicht van de frameworks

				
Angular	React.js	Vue.js	Vaadin	Eclipse Scout 8
				
Ember.js	Aurelia	Mithril.js	Polymer	Riot

3.2.2 Vergelijking van de frameworks

Tabel 5: Vergelijkend overzicht van de frameworks

Naam	Onderhouden door	Laatste release	Licentie	Programmeertaal	Rendering type	Testbaarheid	GitHub stars	Stack Overflow questions
Angular [7]	Google	Maart, 2017	MIT-licentie	TypeScript	Client-side rendering		46.1 K	157.8 K
ReactJs [8]	Facebook	Februari, 2019	MIT-licentie	JavaScript	Client-side rendering		124 K	130.1 K
VueJs [9]	Community	Maart, 2019	MIT-licentie	JavaScript	Client-side rendering		131 K	31.5 K
Vaadin [10]	Vaadin Ltd.	Maart, 2019	Open sourceversie: 'liberal Apache 2.0 license' Pro versie: commerciële licentie	Java	Server-side rendering		1.5 K	4.5 K
EmberJs [11]	Community	Oktober, 2018	MIT-licentie	JavaScript	Client-side rendering		20.8 K	23 K
Eclipse Scout 8 [12]	Eclipse	December, 2018	'Eclipse Public License 1.0'	Java	Mixed rendering		0	122
Aurelia [13]	Community	Juni, 2018	MIT-licentie	JavaScript	Client-side rendering		10.9 K	3 K
MithrilJs [14]	Community	Februari, 2019	MIT-licentie	JavaScript	Client-side rendering		10.8 K	202
Riot [15]	Community	Februari, 2019	MIT-licentie	JavaScript	Client-side rendering		13.5 K	51
Polymer [16] [17]	Community	December, 2018	BSD 3-Clauselicentie	JavaScript	Client-side rendering		20.8 K	8 K

3.2.2.1 Bespreking van de aspecten:

Onderhouden door

Voor sommige *developers* is het belangrijk dat ze weten door wie het framework onderhouden wordt. Dit kan verschillende redenen hebben. Een van de redenen is bijvoorbeeld dat de frameworks die grote bedrijven zelf onderhouden hebben ook vaak gebruikt worden binnen datzelfde bedrijf. Het is dan ook een framework van hoge kwaliteit en zitten er features in die andere frameworks, gemaakt door een community van *developers*, niet hebben.

Verschillende developers hebben daarentegen liever een framework dat onderhouden wordt door een community. Dit omdat de community's achter deze frameworks nog vaak updates uitrollen. Verder worden featureaanvragen vaak sneller aangenomen en geïmplementeerd.

Om te weten te komen hoe andere ontwikkelaars hierover denken heb ik aan een tiental andere ontwikkelaars gevraagd hoe zij hierover denken (zie 3.3.1).

Laatste release

De laatste *release* is zeker iets om in het oog te houden. Sommige frameworks hebben al een hele tijd geen *release* meer gehad. Deze frameworks worden met de tijd steeds meer *outdated* en worden niet meer onderhouden.

Het is dus het beste om een framework te kiezen dat nog vaak *stable releases* uitrolt. Zo beschikken *developers* altijd over de meest recente features die oudere frameworks niet hebben. Verder betekent het dat het framework ook nog goed onderhouden wordt. Als een framework goed onderhouden wordt gaan de *bugs* die er in zitten ook snel opgelost worden.

Licentie

Alle frameworks hebben een licentie. Er zijn twee soorten licenties voor frameworks, namelijk de opensourcelicenties en de betalende licenties. Vaadin heeft bijvoorbeeld een gratis versie met een opensourcelicentie en een betalende versie met zijn eigen licentie.

Er zijn verschillende open sourcelicenties. Elke licentie heeft zijn eigen regels waaraan een ontwikkelaar zich moet houden als hij de code wil gebruiken. Hieronder worden de licenties, die terugkomen in bovenstaande tabel, kort beschreven.

- **MIT License**
  In de MIT-licentie staat dat de ontwikkelaar eigenlijk bijna alles mag doen met de software. Er zijn geen restricties. De ontwikkelaar mag de code zonder limieten gebruiken, kopiëren, wijzigen, *mergen*, publiceren, distribueren en sublicenties toevoegen. Ten slotte mag een ontwikkelaar frameworks, onder een MIT-licentie, verkopen. Dit mag enkel als hij zich aan de voorwaarden houdt. [18]

- **Apache License 2.0**
  De Apache komt vrij goed overeen met de MIT-licentie maar dan met restricties. Als een ontwikkelaar bijvoorbeeld iets verandert aan de Apache *licensed* code, moet hij dat vermelden. Verder moet hij, als hij een product verkoopt met software die Apache *licensed* code gebruikt, dit ook vermelden in de naam van zijn product. [19] [20]

- **BSD 3-Clause License**
  Ook de BSD3-Clauselicentie komt overeen met de MIT-licentie. Het enige verschil is dat een ontwikkelaar de namen van de *developers* die de originele versie geschreven hebben, niet mag gebruiken om een eigen versie te adverteren. [21]

Programmeertaal

De programmeertaal is ook een belangrijk aspect. De meeste *developers* hebben wel een favoriete programmeertaal (zie 3.3.1). Graag zouden ze deze dan ook zoveel mogelijk gebruiken. Binnen Argeüs werken er enkel Java-ontwikkelaars; de keuze was dan al snel gemaakt om voor Eclipse Scout 5 te kiezen. Hierdoor moesten ze geen Javascript of Typescript bestuderen en gebruiken.

Uit dit voorbeeld is dus af te leiden dat de programmeertaal een invloed heeft op de keuze van het framework.

Rendering-type

De manier van het *renderen* is zeker een aspect om niet te vergeten. Het gewenste type kan bij elke applicatie verschillen en is afhankelijk van verschillende aspecten binnen de applicatie. Als het bijvoorbeeld om een webapplicatie gaat, is het beter om voor *client side rendering* te opteren. Als er veel *requests* gedaan moeten worden naar een server, is de beste keuze *server-side rendering*.

De manier waarop een applicatie moet omgaan met het *renderen* ervan is dus zeker iets dat een ontwikkelaar moet weten vooraleer hij een framework kiest. [22] [23]

Testbaarheid

De mogelijkheid om te testen binnen een framework is tegenwoordig een zeer belangrijk aspect. Een eerste voordeel is dat een ontwikkelaar al zijn code kan testen om na te gaan of deze wel degelijk doet wat wordt gevraagd. Een pluspunt hiervan is dat, mocht hij ooit code wijzigen en wilen kijken of alles nog werkt, hij gewoon de testen opnieuw moet runnen.

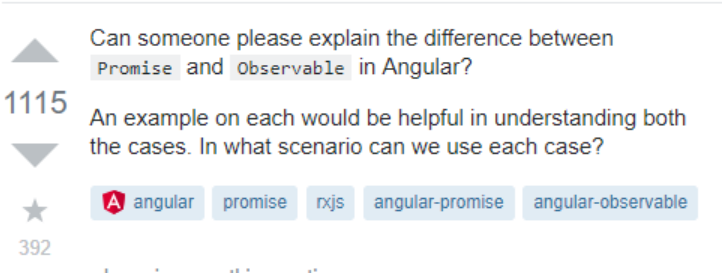
Verder biedt het gebruik van een *testlibrary* ook de mogelijkheid om die testen in een *pipeline* te steken. Als niet alle testen slagen, kan de code die ervoor zorgt dat de applicatie niet meer werkt, tegengehouden worden. De code moet dan herbekeken en aangepast worden voordat deze op de *repository* terecht komt.

GitHub-stars

GitHub-*stars* worden als een soort meetsysteem gebruikt om te weten te komen hoe populair een *repository* is. *Developers* kunnen binnen GitHub *repositories* "*starren*". Dit doen ze als ze een project fijn, interessant of goed uitgewerkt vinden. Aan de hand van GitHub-*stars* krijgt een ontwikkelaar dus snel een beeld van hoe populair het framework is. [24]

Stack Overflow questions

Stack Overflow kan ook als een meetsysteem gebruikt worden. Ontwikkelaars kunnen op dit platform vragen stellen als ze ergens mee vast zitten. Aan deze vragen hangen één of meerdere *tags*. Eén van die *tags* is vaak het framework. Dus hoe meer vragen en oplossingen er zijn hoe groter de community erachter is.

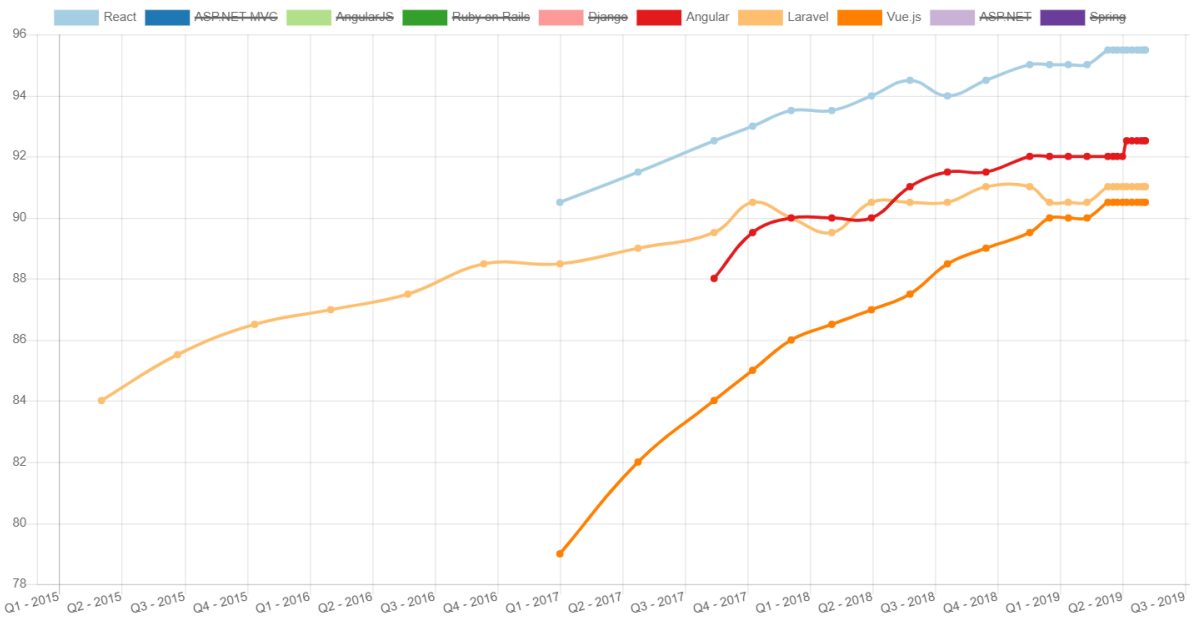


Figuur 20: Stack Overflow-vraag

Kleine toelichting:

De maatstaven, GitHub *stars* en Stack Overflow-vragen, (GitHub *stars* & Stack Overflow *questions*) die hierboven gebruikt zijn uiteraard niet wetenschappelijk onderbouwd. Deze cijfers bepalen niet of een framework goed of slecht is. Ze geven echter wel een blik op de populariteit. Dus hoe meer mensen er mee bezig zijn hoe meer GitHub *stars* en Stack Overflow *questions* er zijn.

Er zijn echter wel verschillende websites die frameworks gaan vergelijken aan de hand van dergelijke maatstaven zoals bijvoorbeeld GitHub *stars*. Met behulp van deze maatstaven gaan ze dan een score geven aan het framework. [25]



Figuur 21: Grafiek frameworkscores

3.3 Fase 2 – De filtratie

3.3.1 Wat vindt Argeüs belangrijk?

Om te weten te komen wat iedereen binnen Argeüs belangrijk vindt binnen een frontendframework, is er samengezeten met iedereen binnen het bedrijf. Op deze manier is iedereen betrokken bij het beslissingsproces.

Nadat afloop is er een samenvatting gemaakt van ieders mening. Hieronder is de samenvatting terug te vinden. Onder de titel 'Algemeen' is terug te vinden wat iedereen belangrijk vindt binnen Argeüs. Daarna wordt er per departement een overzicht gegeven van wat de werknemers specifiek belangrijk vinden.

3.3.1.1 Algemeen

Uiteraard kwamen sommige meningen overeen en werden er verschillende aspecten meerdere keren opgenoemd tijdens de interviews. De meningen die overeenkwamen worden hieronder toegelicht.

Een eerste aspect dat vaak terug naar boven kwam, is **design**. Iedereen is akkoord dat de frontend er modern uit moet zien. Hij moet mensen uitnodigen om het product te gebruiken. Ook wordt verwacht dat de frontend *responsive* is zodat de producten ook op smartphones, tablets, enzovoort bruikbaar zijn. Het toekomstige framework moet dus zeker een goede *UI-library* hebben.

Een aspect dat hierbij aansluit is de **gebruiksvriendelijkheid**. Iedereen binnen Argeüs gaat akkoord dat de gebruiksvriendelijkheid van de huidige producten veel beter kan. Er is nood aan een framework waarmee een duidelijke *flow* kan worden gemaakt zodat de klanten alles beter kunnen vinden. Zo zal Argeüs ook minder vragen binnen krijgen van de klanten.

Iedereen is het er ook over eens dat er veel **documentatie** moet zijn van het framework. Zo kunnen de ontwikkelaars sneller ontwikkelen omdat ze functies binnen het framework kunnen opzoeken om ze daarna misschien te gebruiken in plaats van ze zelf te schrijven.

Ook moet de frontend goed **onderhoudbaar** zijn. Kleine aanpassingen mogen bijvoorbeeld niet veel tijd kosten. Het moet makkelijk zijn om pagina's aan te passen, toe te voegen, verwijderen, enz. Met het huidige frontendframework is dit niet altijd zo simpel, omdat Eclipse Scout 5 op een vreemde manier websites opbouwt.

De grootte van de **community** is voor velen ook een zeer belangrijk aspect. Als er een grote community achter een framework zit, is het te verwachten dat het framework ook goed werkt. Verder is een grote community ook handig voor ontwikkelaars. Als een ontwikkelaar bijvoorbeeld vast zit met een probleem met betrekking tot het framework, wordt hij al snel verder geholpen.

3.3.1.2 Developmentteam

Binnen Argeüs zijn er vier ontwikkelaars. Zij hebben heel wat aspecten opgenoemd.

Alle ontwikkelaars vermelden dat de **leercurve** van het framework zo goed mogelijk moet zijn. Dit hangt natuurlijk af van verschillende aspecten die eerder al vermeld zijn. Om een goede leercurve te hebben moet een framework voldoende documentatie voorzien en een grote community hebben die ontwikkelaars snel terug op het juiste pad helpt. Ook mag de groei van het framework niet te snel zijn noch te traag.

Wat de ontwikkelaars nog aanhaalden is dat het framework een bepaalde **structuur** moet volgen. Graag zouden ze een structuur zoals MVC of MVVM zien. Zo blijft alles goed overzichtelijk en is de code ook veel gemakkelijker om te onderhouden.

Omdat Argeüs in een DevOps-omgeving werkt, zouden de ontwikkelaars graag de mogelijkheid hebben om hun frontendcode te **testen**. De testen die ze dan schrijven kunnen dan bij in de *pipeline* gestoken worden. De *developers* weten dan direct of er iets kapot is gegaan in de omgeving waar ze naar willen pushen.

Continuïteit is ook een vernoemd aspect. Hiermee bedoelen ze dat ze een framework willen dat nog lang gebruikt en geüpdatet zal worden.

De ontwikkelaars gaven aan dat het **aantal *components*** die al in het framework zitten ook belangrijk zijn. Als er al veel *components* inzitten, moeten ze er zelf minder creëren.

Uiteraard is er ook gevraagd aan de ontwikkelaars in welke **taal** ze het liefst zouden willen programmeren binnen een frontendframework. Hieruit is gebleken dat TypeScript de favoriet is, gevolgd door JavaScript.

De meeste ontwikkelaars hadden ook al wat **ervaring** met bepaalde frameworks. De meeste waren al in contact gekomen met Angular. Sommigen zijn ook al wat bezig geweest met Vue.

Tijdens de bevraging gaven de ontwikkelaars ook dingen mee die ze *nice to have* vonden.

- Een van die dingen die ze graag zouden hebben is dat het framework '*Hot reloading*' heeft. Dit houdt in dat als er aanpassingen gedaan worden in de code, en de applicatie is aan het runnen, de webpagina automatisch refresht. Dus ze hoeven niet telkens de applicatie helemaal te herstarten om hun aanpassingen te zien.

- Wat ze ook graag willen is dat het framework diverse IDE's ondersteunt. Als een framework diverse IDE's ondersteunt, kan de ontwikkelaar zelf kiezen welk IDE hij wilt gebruiken om te ontwikkelen.

- De mogelijkheid van *in browser databases* zouden ze ook fijn vinden. Dit biedt veel verschillende mogelijkheden voor de rekenmotor die in de huidige applicaties gebruikt wordt.

- Een ingebouwde designtool zou ook een pluspuntje zijn. Zo zouden de ontwikkelaars minder werk moeten steken in de HTML en CSS van de frontend.

3.3.1.3 Testteam

Tijdens het onderzoeken van wat de testers belangrijk vinden is ondervonden dat ze niet zo veel kennen van frontendframeworks. Dit is uiteraard logisch aangezien ze niet bezig zijn met ontwikkelen. Wel hebben ze een belangrijk aspect aangehaald.

Het aspect wat de testers specifiek aanhaalden is de testbaarheid van het framework. Qua testbaarheid bedoelen ze dat er de mogelijkheid moet zijn om de UI degelijk te testen. Hiervoor kunnen ze dan een testtool gebruiken zoals Selenium.

3.3.1.4 Managementteam

Het managementteam haalde diverse aspecten aan die bij de andere teams niet aanbod kwamen. Dit omdat ze op een heel andere manier denken dan IT'ers.

Een eerste aspect is bijvoorbeeld dat het framework de mogelijkheid moet bieden voor **internationalisatie**. Er moet dus een systeem zijn waarmee de applicatie in verschillende talen bruikbaar is afhankelijk van wat de gebruiker instelt. Dit is zeker een vereiste binnen het framework omdat de applicaties ook gebruikt worden in Wallonië en Brussel.

De **kost** van het framework mag ook niet te hoog liggen op expliciete vraag van de *sales director*. Hij heeft liever een gratis of goedkoop framework dat even goed is als de andere die wel betalend zijn.

Verder werd er ook gesproken over het werken met **thema's**. Argeüs gaat in de toekomst een product ontwikkelen dat ze dan aan verschillende klanten gaan verkopen. Argeüs wil de klanten graag laten voelen dat het product van de klant wel degelijk van de klant is. Het moet dus mogelijk zijn dat iedere klant zijn eigen thema heeft en dus zijn hun eigen huisstijl kan inbrengen bij de applicatie.

Ook is het belangrijk dat er een goede communicatie is tussen Argeüs en haar klanten. Op dit moment bestaat er een invulformulier dat klanten kunnen invullen als ze vragen hebben en waarop niet alles ingevuld hoeft te worden. Het zou interessant zijn een **validatie** te hebben op alle velden binnen zo'n formulier.

Verder werd er gedacht aan de mogelijkheid om misschien een **chatbot** te introduceren. Een goede chatbot, kan immers heel wat mensen opvangen die met basisvragen komen. Dat bespaart veel tijd bij de servicedesk.

Ten slotte geeft de *business director* van Argeüs al mee dat hij een voorkeur heeft voor het frontendframework genaamd Angular, omdat hij hier al wat *research* naar gedaan heeft en hier al goede ervaringen mee heeft.

3.3.1.5 Infrastructuurteam

Het infrastructuurteam vindt verschillende aspecten belangrijk. Zij dachten vooral ook aan de **security**kant van het framework. Het framework moet in staat zijn om data van de frontend naar de *back end* geëncrypteerd door te sturen. Dit is een belangrijk punt aangezien alle applicaties van Argeüs werken met gevoelige data.

Uiteraard moet het framework ook goed bestand zijn tegen aanvallen zoals bijvoorbeeld Cross-site scripting. Er moet dus een goede **validatie** zijn op de invulvelden.

Verder had het infrastructuurteam ook laten weten dat ze graag een frontend hebben die **makkelijk te *deployen*** valt. Hiermee bedoelen ze dat ze niet te veel aanpassingen willen doen aan instellingen en *properties* van de frontendapplicatie om deze te *deployen*.

3.3.2 Welke frameworks zijn het meest geschikt?

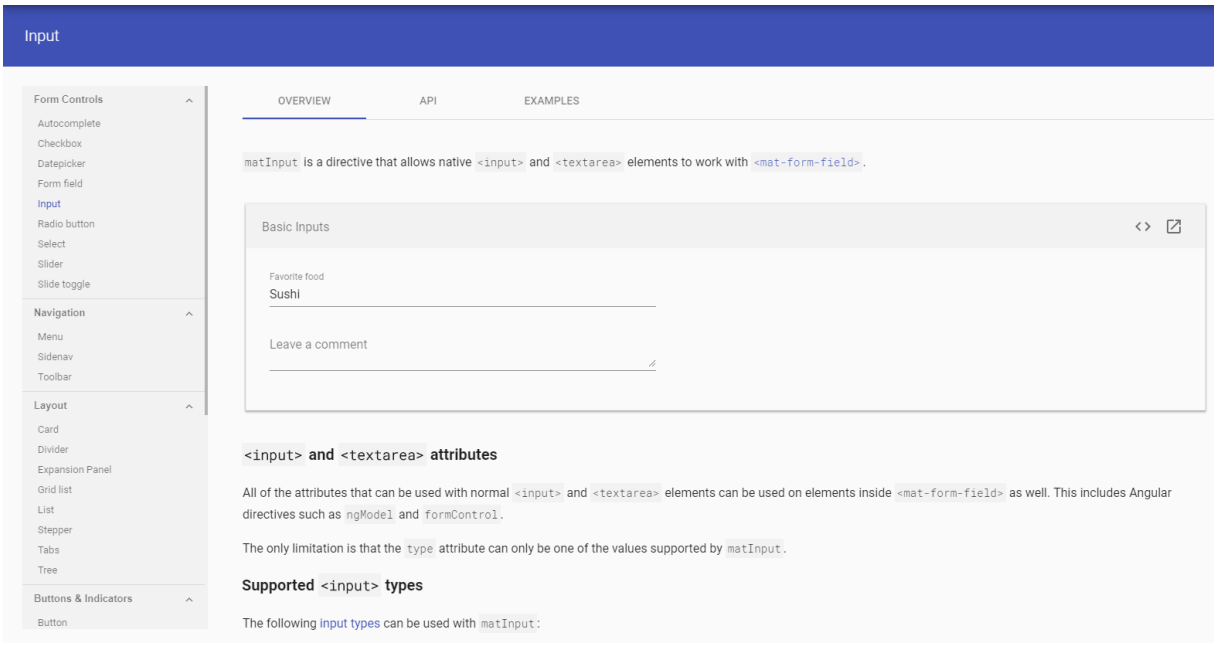
In dit gedeelte worden er vijf frameworks geselecteerd aan de hand van de aspecten die Argeüs belangrijk vindt. Van deze vijf frameworks worden er een aantal sterke en zwakke punten toegelicht.

3.3.2.1 Angular

Angular is één van de vijf gekozen frameworks. Angular komt goed overeen met de aspecten die Argeüs belangrijk vindt. Deze aspecten zullen naar boven komen tijdens het bespreken van de sterke punten van Angular. [26] [27]

Sterke punten

Het eerste sterke punt van Angular zijn de componenten. Angular bevat een heel galerij aan componenten, deze galerij wordt Angular Material genoemd. Angular biedt een super gebruiksvriendelijke website aan waarop ontwikkelaars opzoek kunnen gaan naar het geschikte component. Over dit component krijgen ze heel veel informatie. Zo wordt er bijvoorbeeld een overzicht gegeven waarin beschreven wordt waarvoor het component dient en hoe het gebruikt moet worden. Daarnaast kunnen ze ook kijken naar API-referenties. Ook kan de ontwikkelaar naar een aantal voorbeelden gaan kijken. In deze voorbeelden krijgt de ontwikkelaar te zien hoe de frontend en de code eruitzien. Er worden verschillende voorbeelden gegeven. Elk voorbeeld heeft iets speciaals. [28]



Figuur 22: Angular Material Input voorbeeld

Angular heeft ook een *component-based* architectuur. Deze structuur resulteert in een hogere kwaliteit van de code. Componenten kunnen in elkaar gebruikt worden en toch onafhankelijk zijn van elkaar.

Werken met een *component-based* architectuur brengt ook heel wat voordelen met zich mee die overeenkomen met de aspecten van Argeüs.

De ontwikkelaars haalden aan dat ze graag zoveel mogelijk code willen hergebruiken. Door te werken met de *component-based* architectuur zal dit zeker gaan. De gemaakte componenten kunnen zoveel als de ontwikkelaars willen hergebruikt worden. Ze moeten het component één keer aanmaken om ze vervolgens door heel de applicatie te gebruiken.

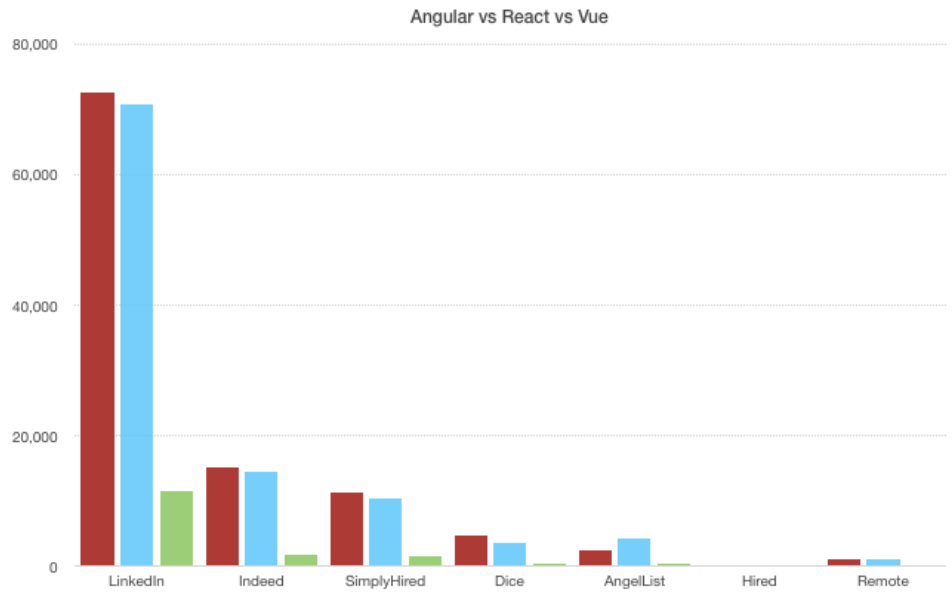
Ook zijn alle componenten te testen. De code van de componenten valt heel goed te testen. Hiervoor kunnen verschillende *libraries* gebruikt worden. Eén voorbeeld van zo een *library* is Karma. Met behulp van deze *library* kan een ontwikkelaar testen gaan schrijven om bijvoorbeeld na te gaan of de juiste data wordt doorgegeven. Ook kan hij het gedrag van de componenten testen. [29]

Nog een reden waarom Angular gekozen was is omdat er geprogrammeerd word met Typescript. Dit omdat een groot deel van de ontwikkelaars een voorkeur hebben voor Typescript.

Argeüs wou ook graag een framework hebben dat nog lang ondersteund zal worden. Ook hier heeft Angular weer een pluspuntje mee behaald. Google heeft namelijk laten weten dat ze *Long-Term Support* zullen leveren aan de technologie. Ze gaan Angular dus blijven onderhouden en updates uitrollen.

Uit verschillende cijfers kan afgeleid worden dat Angular een grote community heeft. Angular heeft bijvoorbeeld het tweede hoogste aantal *npm-package* downloads. Een *npm-package* van Angular dient gedownload te worden als iemand een Angular-applicatie wilt bouwen. Ook zijn er 156000 vragen op Stack Overflow te vinden met de tag Angular. Dit toont aan dat er veel ontwikkelaars bezig zijn met Angular.

Er is ook een onderzoek geweest naar welke frameworks het meest gevraagd werden in onlinevacatures. Het onderzoek ging wel enkel over de frameworks: Angular (rood), React (blauw) en Vue (groen). Uit de cijfers is gebleken dat Angular het meest gevraagd werd. Dit geeft ook een beeld op wat bedrijven willen hebben. [30]



Figuur 23: Grafiek van gevraagde frameworks in onlinevacatures

Zwakke punten

Hoewel het werken met een *component-based* architectuur veel voordelen met zich meebrengt, hangt er ook een nadeel aan. Het project kan ingewikkeld worden naar mate het aantal componenten. Een component heeft heel wat instellingen nodig. Deze worden in aparte files opgeslagen. Een voorbeeld hiervan is bijvoorbeeld het verhaal van de *dependency injection*. Als een component bepaalde *dependencies* nodig heeft, dienen deze in een aparte file geïnjecteerd te worden.

De documentatie van de *Command Line Interface* (CLI) binnen Angular valt ook niet goed mee. Hiervan is niet veel te vinden maar als een ontwikkelaar ergens mee vast zit kan hij wel snel verder geholpen worden door de community.

3.3.2.2 EmberJs

Ook EmberJs had een aantal aspecten die overeenkwamen met die van Argeüs. Daarom is het ook één van de gekozen frameworks. [31] [32] [33] [34]

Sterke punten

Uit de verschillende bronnen is af te leiden dat de documentatie van Ember van goede kwaliteit is. De *application programming interface* (API) van Ember is goed en duidelijk beschreven zodat ontwikkelaars er zo snel mogelijk mee aan de slag kunnen gaan. Ook de CLI van Ember is er goed omschreven. Op de website: https://ember-cli.com/ (zie bron 35) staan er heel wat *guidelines*, voorbeelden van commando's, enzovoort. De verschillende mogelijkheden met de API en CLI staan dus allemaal op die website beschreven. Dit maakt het ook wat gemakkelijker voor de ontwikkelaars omdat alles bij elkaar staat. [35]

Bij Ember staan ze ook voor '*Convention over configuration*'. Dit betekent dat ontwikkelaars veel minder moeten gaan configureren voordat ze aan een project beginnen. Ember is zelf zo slim om te detecteren wat het doel van de file is en stuurt bij waar het nodig is. Het uiteindelijke voordeel van dit is dat er veel minder code moet geschreven worden.

Ember heeft ook een groot aantal voorgeschreven componenten. Deze kan de ontwikkelaar goed gebruiken indien hij deze nodig heeft en bespaard er dus ook veel tijd mee. Zelf kan de ontwikkelaar natuurlijk ook zelf componenten ontwikkelen en toevoegen aan zijn project. Hoe hij deze juist moet ontwikkelen staat ook perfect toegelicht in de documentatie van Ember. [36]

Het framework is een MVC framework. Dit soort architectuur had Argeüs ook graag gewild omdat ze al vaker met deze architectuur gewerkt hebben.

Ook Ember heeft testmogelijkheden. Ember heeft een *addon*, genaamd 'ember-qunit'. Met behulp van deze *addon* kan de ontwikkelaar verschillende soorten testen gaan schrijven. Hij kan *unit tests*, *container tests, rendering tests* en *application tests* gaan schrijven. De ontwikkelaar kan zijn code goed kwalitatief houden omdat hij zo goed als alles kan gaan testen. [37]

Figuur 24: Voorbeeld Ember Unit Test

Zwakke punten

Wat er af te leiden valt uit de bronnen is dat Ember een moeilijke leercurve heeft. De reden waarom dit zo een moeilijke leercurve heeft, is uit bronnen niet af te leiden. Er zijn geen echte redenen te vinden waarom het zo moeilijk is. De moeilijkheid wordt waarschijnlijk pas echt duidelijk wanneer er zelf voor de eerste keer mee ontwikkeld wordt.

Nog een nadeel van Ember is dat het een *heavy weight* framework is. Het is dus niet aangeraden om kleine projecten te doen in Ember.

Ten slotte heeft Ember nog een nadeel. De community is namelijk kleiner en kleiner aan het worden omwille van de opmars van andere frameworks zoals: React, Vue en Angular. Dit zorgt er voor dat er minder mensen met Ember gaan ontwikkelen en dat het framework uiteindelijk zal 'uitsterven'.

3.3.2.3 VueJs

Vue behoort tot de top drie van de webframeworkmarkt. Vue heeft dan ook veel gemeenschappelijke aspecten met Argeüs. Een aantal van deze aspecten zullen naar boven komen tijdens het bespreken van de voor en nadelen. [38] [39] [40] [41]

Sterke punten

Vue heeft een zeer goede documentatie. Net als Ember en Angular heeft het een eigen website waar alles wordt toegelicht. Er is documentatie te vinden dat gaat van "*What is Vue?*" tot het migreren van een oude vue versie naar een nieuwe. De documentatie is dus zeker een pluspunt. [42]

Ook kan er met VueJs in Javascript of Typescript geprogrammeerd worden. VueJs levert namelijk Typescript *support* en voegt bij elke update nog meer Typescript features toe aan het framework. Het is handig voor Argeüs dat ze zelf nog een keuze kunnen maken als ze Vue zouden kiezen. Ze kunnen het ook door elkaar gaan gebruiken maar dat is niet altijd aangeraden.

Verder kan bij het gebruik van het VueJs-framework zelf gekozen worden welk architectuur ze gaan aanbrengen in de applicatie. Vue is hier heel flexibel in en dat is handig omdat elk project verschillende belangheden hebben.

Testen schrijven voor een Vue-applicatie is ook een mogelijkheid. De CLI van Vue heeft namelijk built-in opties om *unit tests* te gaan schrijven met behulp van de *libraries,* 'Jest' of 'Mocha'. De ontwikkelaar heeft dus weer zelf de keuze in handen om zelf een keuze te maken welke *library* hij wilt gebruiken. [43]

```JS
import Vue from 'vue'
import MyComponent from './MyComponent.vue'

// helper function that mounts and returns the rendered text
function getRenderedText (Component, propsData) {
  const Constructor = Vue.extend(Component)
  const vm = new Constructor({ propsData: propsData }).$mount()
  return vm.$el.textContent
}

describe('MyComponent', () => {
  it('renders correctly with different props', () => {
    expect(getRenderedText(MyComponent, {
      msg: 'Hello'
    })).toBe('Hello')

    expect(getRenderedText(MyComponent, {
      msg: 'Bye'
    })).toBe('Bye')
  })
})
```

Figuur 25: Unit test in VueJs

Vue beschikt ook over wat voorbereide componenten. Deze kan de ontwikkelaar gaan gebruiken in zijn eigen applicaties zonder dat hij er zelf werk in moet steken. Dit bespaart weer veel tijd en kan de ontwikkelaar sneller werken. Uiteraard is het ook mogelijk om zelf componenten te gaan schrijven. Hiervoor is er een tutorial te vinden in de documentatie van VueJs.

Nog een belangrijk punt is dat Vue een gemakkelijke leercurve heeft. Dit is afgeleid uit de verschillende bronnen. Vue heeft namelijk veel mooie, groot geschaalde templates. Deze templates kunnen de beginners gebruiken om zichzelf wat wegwijs te maken binnen een Vue-applicatie.

De community van Vue is niet zo groot als die van Angular of React. Toch is het de derde grootste community van de frameworks. Dit is te af te leiden van de tabel die al eerder in dit eindwerk werd getoond (Tabel 4: Overzicht van de frameworks).

Een van de grootste voordelen van VueJs is de grootte ervan. Vue heeft een hele kleine *size* vergeleken met bijvoorbeeld Angular of Ember.

Name	Size
Ember 2.2.0	435K
Ember 1.13.8	486K
Angular 2	566K
Angular 2 + Rx	**766K**
Angular 1.4.5	143K
Vue 2.4.2	58.8K
Inferno 1.2.2	48K
Preact 7.2.0	**16K**
React 0.14.5 + React DOM	**133K**
React 0.14.5 + React DOM + Redux	139K
React 16.2.0 + React DOM	97.5K

Figuur 26: Groottes van webframeworks

Zwakke punten

Een van de nadelen van Vue is dat het misschien te flexibel kan zijn op vlak van architectuur. Dit omdat de ontwikkelaars zelf de vrijheid hebben om te kiezen welke structuur ze gaan implementeren. Als er veel ontwikkelaars features gaan toevoegen en per ongeluk van de architectuur afwijken, kan het zijn dat ze op die afwijking blijven verder gaan. Als dit zich zou voor doen zou het project een wirwar worden.

Omdat de community wat kleiner is dan die van Angular of React, zullen problemen minder snel opgevangen worden door de community. Ook zullen er minder aanvragen van de community komen dan bij Angular of React. Een aanvraag kan bijvoorbeeld een featureaanvraag zijn waar de ontwikkelaar zelf niet aan gedacht heeft maar toch heel handig zou kunnen zijn.

3.3.2.4 React

React is op dit moment het populairste framework. De sterke punten van dit framework komen goed overeen met de belangrijke aspecten van Argeüs. Dit is dan ook de reden waarom React gekozen is om verder uit te werken.

Sterke punten

React heeft de grootste community van alle webframeworks. Dit is af te leiden uit verschillende bronnen en onderzoeken. In onderstaande figuren staan een aantal grafieken die meer info geven op de grootte van de community. [44] [45] [46] [47] [48] [49]

Figuur 27: npm-package downloads React vs. Angular

Zoals in bovenstaande grafiek te zien is worden de *npm-packages* van React veel meer gedownload. Er worden dus meer React-projecten aangemaakt dan Angular-projecten. Het framework is dus zeker bij veel ontwikkelaars geliefd. Ook wordt de community ondersteund door het grote bedrijf Facebook.

Ook bij React is er de mogelijkheid om componenten te gaan hergebruiken. De ontwikkelaar kan een component gebruiken van React zelf of een eigen geschreven component, zoveel als hij wilt hergebruiken. Dit bespaart de ontwikkelaar veel code en tijd.

React blijft constant verder evolueren. Ze blijven nieuwe features toevoegen om het framework zo volledig mogelijk te laten voelen. Andere nieuwigheden die andere frameworks hebben, worden dus snel bijgehaald en geïmplementeerd.

Nog één van de grote pluspunten van React is de *Virtual DOM*. De DOM bevat de descriptie van de hiërarchische structuur van de website. De *Virtual DOM* van React werkt veel sneller dan de traditionele DOM van Angular bijvoorbeeld. De *Virtual DOM* is eigenlijk een kopie van de echte DOM en kan gebruikt worden om kleine gedeeltes van de website te updaten. Vermits de *Virtual DOM* veel kleiner is en onnodige informatie niet bijhoudt zal de *performance* van de website stijgen. Dit omdat websites tegenwoordig redelijk complex kunnen zijn.

Testen schrijven is binnen React ook mogelijk. Er zijn twee grote *libraries* die aangeraden zijn om te gebruiken, namelijk: Jest en Enzyme. Er kunnen hier ook weer diverse soorten testen geschreven worden. UI testen, javascript testen, … zijn allemaal mogelijke soorten. [50]

```
// Link.react.test.js
import React from 'react';
import Link from '../Link.react';
import renderer from 'react-test-renderer';

test('Link changes the class when hovered', () => {
  const component = renderer.create(
    <Link page="http://www.facebook.com">Facebook</Link>,
  );
  let tree = component.toJSON();
  expect(tree).toMatchSnapshot();

  // manually trigger the callback
  tree.props.onMouseEnter();
  // re-rendering
  tree = component.toJSON();
  expect(tree).toMatchSnapshot();

  // manually trigger the callback
  tree.props.onMouseLeave();
  // re-rendering
  tree = component.toJSON();
  expect(tree).toMatchSnapshot();
});
```

Figuur 28: Voorbeeld test in Jest

Ook bestaat er binnen ReactJS, React Native. Met React Native kunnen ontwikkelaars mobiele applicaties gaan creëren. Ze kunnen ze gaan ontwikkelen met dezelfde middelen die ze zouden hebben bij het ontwikkelen van een website. Ook worden de mobiele applicaties dan geprogrammeerd in Javascript.

Het migreren van React-applicaties is ook gemakkelijk. De ontwikkelaar moet enkele CLI-commando's invoeren die de dependencies in de package.json veranderen naar de gewenste versie. Daarna moet hij de applicatie herstarten en zal de migratie succesvol gelukt zijn.

Zwakke punten

De hoeveelheid van updates die React uitrolt kan ook een negatief effect hebben. Hoe vaker er nieuwigheden, veranderingen gebeuren aan het framework moet er natuurlijk ook documentatie voor geschreven worden. Dit is echter niet zo gemakkelijk als het tempo van de updates zo hoog ligt. Er is dus veel documentatie dat nog ontbreekt wat er ook voor zorgt dat het de leercurve gaat beïnvloeden.

React werkt met een 'View' structuur en niet met een MVC structuur. De ontwikkelaar moet zich slechts aan de 'V' houden van de MVC structuur. Dit kan nogal verwarrend worden omdat de ontwikkelaar veel vrijheid heeft op de andere lagen van de applicatie. Ook wordt hier niet echt op gecheckt of het overzichtelijk blijft door het framework, Vue heeft dit bijvoorbeeld wel.

3.3.2.5 Aurelia

Ook Aurelia heeft heel wat goede punten die overeenkomen met de belangrijke aspecten van Argeüs. Daarom is deze geselecteerd om verder uit te werken. [51] [52]

Sterke punten

Aurelia biedt ook *support* aan het programmeren in Typescript. Elke *library* binnen Aurelia heeft zijn eigen Typescript-*files*. Ook zijn er enkele *kits* die ontwikkelaars kunnen gebruiken. Zo zijn er bijvoorbeeld *beginnerkits* en meer geavanceerde *kits*.

Het managementteam had graag ook kunnen rekenen op *support*. Aurelia biedt *support* aan *enterprises* die hun framework gebruiken. Zo worden de bedrijven die het gebruiken dezelfde dag nog geholpen zodat ze verder kunnen. Moest er een kritische fout zijn, biedt Aurelia ook *live support*. Bedrijven kunnen dus een *live call* beginnen met een van de helpdeskmedewerkers. [53]

Componenten zijn ook goed te testen met de *testlibrary* van Aurelia zelf. Aurelia heeft een mooie tutorial die gevolgd kan worden om mensen te introduceren met de *library*. Ook kunnen er '*end to end*' testen geschreven worden met deze *library*. [54]

```
import {StageComponent} from 'aurelia-testing';
import {bootstrap} from 'aurelia-bootstrapper';

describe('MyAttribute', () => {
  let component;

  beforeEach(() => {
    component = StageComponent
        .withResources('src/my-attribute')
        .inView('<div my-attribute.bind="color">Bob</div>')
        .boundTo({ color: 'blue' });
  });

  it('should set the background color to provided color', done => {
      component.create(bootstrap).then(() => {
        expect(component.element.style.backgroundColor).toBe('blue');
        done();
      }).catch(e => console.log(e.toString()));
  });

  afterEach(() => {
    component.dispose();
  });
});
```

Figuur 29: Eigen geschreven componenttest in Aurelia

In bovenstaande figuur is een test te zien die een eigen geschreven component test. Uit de figuur is af te leiden dat de syntax redelijk simplistisch is.

Bij Aurelia is het integreren van andere frameworks/*libraries* ook goed ondersteund. Zo kunnen bijvoorbeeld frameworks zoals: React, Polymer, jQuery en Bootstrap gemakkelijk geïntegreerd worden.

Ook bij Aurelia geld de regel: '*Convention over Configuration*'. Er dient gehouden te worden aan de SOLID-principes zodat de code die geschreven moet worden zo klein mogelijk blijft. Zo blijft het makkelijker te onderhouden. Ook betekent dit dat er minder geprutst moet worden aan het doen werken van de API van het framework. Er kan meer gefocust worden op het ontwikkelen van de applicatie zelf.

Zwakke punten

Aurelia heeft een kleine community en brengt verschillende nadelen met zich mee. Een eerste nadeel is dat er niet veel oplossingen op internet staan. Dit omdat er niet zo veel mensen mee bezig zijn zoals bijvoorbeeld bij Angular of React. Er komen minder problemen/moeilijkheden online te staan die opgelost kunnen worden door anderen.

Ook de documentatie heeft zo zijn minpunten. Bij veel componenten en features ontbreekt de documentatie. Ontwikkelaars zullen dus zelf moeten onderzoeken hoe het werkt en dat vergt natuurlijk ook tijd.

Het aantal componenten binnen Aurelia zelf is ook een tegenvaller. Er zijn veel componenten die andere frameworks wel hebben. Dit probleem kan wel verholpen worden door de goede integratiemogelijkheden met andere frameworks.

3.4 Fase 3 – De potentiële frameworks voor Argeüs

3.4.1 Inleiding

In deze fase worden er twee prototypes uitgewerkt. Eén van Angular en één van VueJs. Beide prototypes zullen hetzelfde gaan doen. Er wordt een prototype van een kleine managementtool gemaakt. Deze managementtool zal instaan voor het aanmaken van gebruikers, het koppelen van producten aan gebruikers en het zien van de verschillende *subscriptions* (prijs die de klant betaalt voor de applicatie(s) per maand.

Het doel van deze prototypes is om de verschillende aspecten van het framework toe te lichten. Zo krijgt Argeüs direct een beeld op de mogelijkheden.

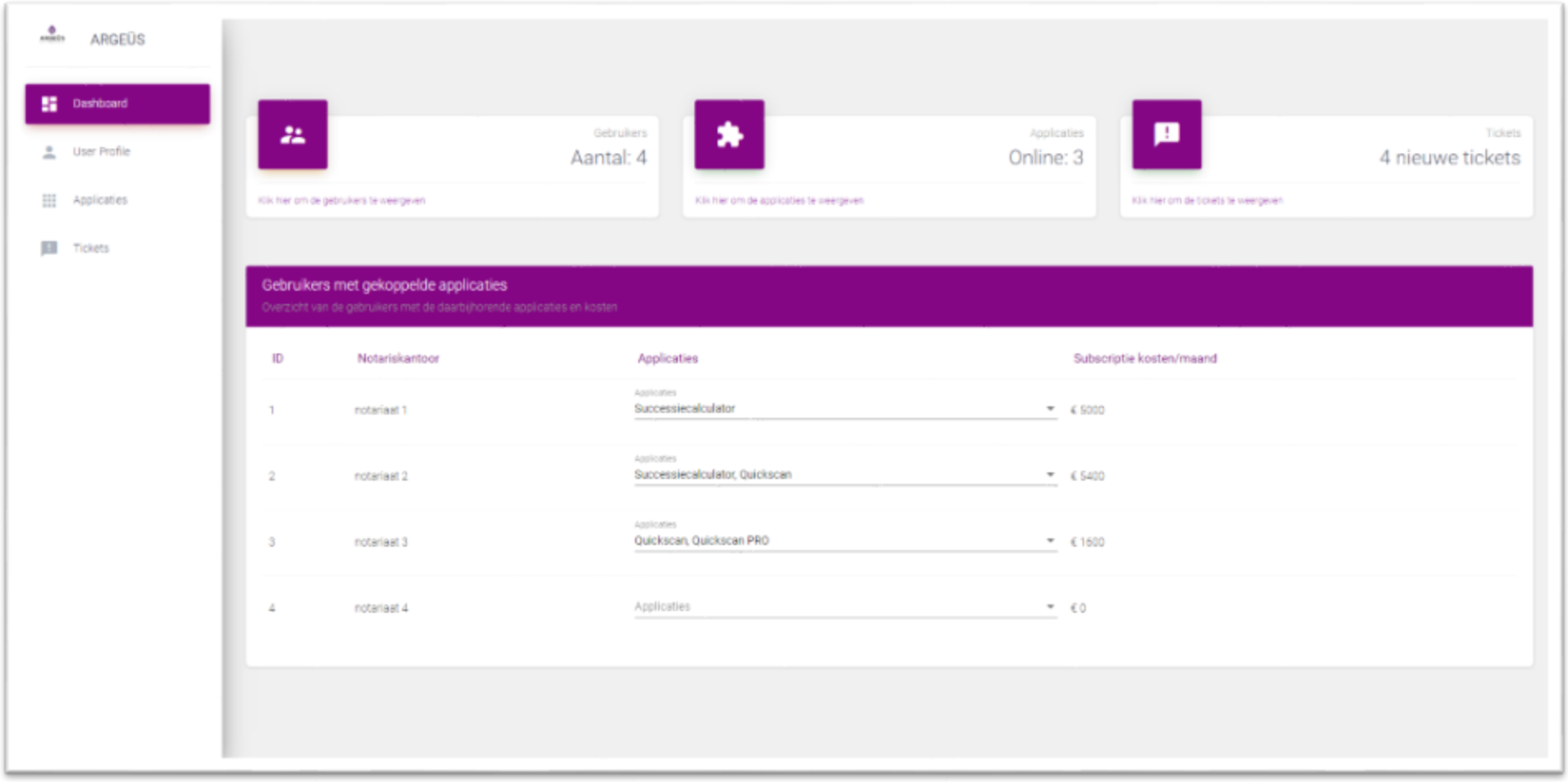
Angular en VueJs zijn niet willekeurig gekozen. De vijf frameworks van fase twee zijn kort besproken met verschillende ontwikkelaars binnen Argeüs. Hier is uitgekomen dat Angular en VueJs de beste keuzes zijn voor Argeüs.

Binnen Argeüs waren de meeste mensen al bekend met Angular. Ook waren er al een aantal die ervaring hadden met VueJs. Dit heeft natuurlijk een grote impact op de keuze omdat ze dan minder lang tijd moeten spenderen aan het aanleren van het framework.

De keuze tussen React en VueJs was echter niet zo gemakkelijk. Toch is er samen besloten om voor Vue te kiezen omdat er een aantal aspecten bij React minder goed overeenkwamen met wat Argeüs wou. Eén van die voorbeelden was bijvoorbeeld de architectuur van React.

3.4.2 Angular prototypes

In onderstaande afbeelding is het dashboard te zien. Hieronder zullen verschillende aspecten die terugkomen in het prototype besproken worden.



Figuur 30: Angular-prototype: Dashboard

3.4.2.1 Opbouw van het prototype

Er is gebruik gemaakt van de *component-based* architectuur. Er zijn verschillende componenten aangemaakt om de diverse schermen op te bouwen. Deze componenten kunnen doorheen de applicatie door gebruikt worden.

In onderstaande afbeelding is een klasse genaamd admin-layout.component.html te zien. In deze klasse worden er verschillende componenten gebruikt, namelijk de AppSideBar-component en de RouterOutlet-component.

```
<div class="wrapper">
    <div class="sidebar" data-color="danger" data-background-color="white" data-image="./assets/img/sidebar-1.jpg">
        <app-sidebar></app-sidebar>
    </div>
    <div class="main-panel">

        <router-outlet></router-outlet>

    </div>
</div>
```

Figuur 31: Angular-prototype: Admin Layout Component

Elk component heeft zijn eigen html en Typescript-file. Het is dus mogelijk om verschillende componenten samen te zetten om zo tot een volledige webpagina te komen.

3.4.2.2 Design

Om tot dit design te komen zijn er componenten van Angular en Angular Material gebruikt. Sommigen hiervan zijn dan nog eens extra gestyled met behulp van Bootstrap.

Hieronder is een voorbeeld te zien van hoe de *dropdownlist* is opgebouwd met Angular Material. Deze hebben allemaal al ingebouwde animaties.

```
<mat-form-field>
    <mat-label>Applicaties</mat-label>
    <mat-select (selectionChange)="onItemSelect(applicationDdl, $event, user)" [formControl]="applicationDdl" multiple>
        <ng-container *ngFor="let application of applications">
            <mat-option [value]="application">{{application.applicationName}}</mat-option>
        </ng-container>
    </mat-select>
</mat-form-field>
```

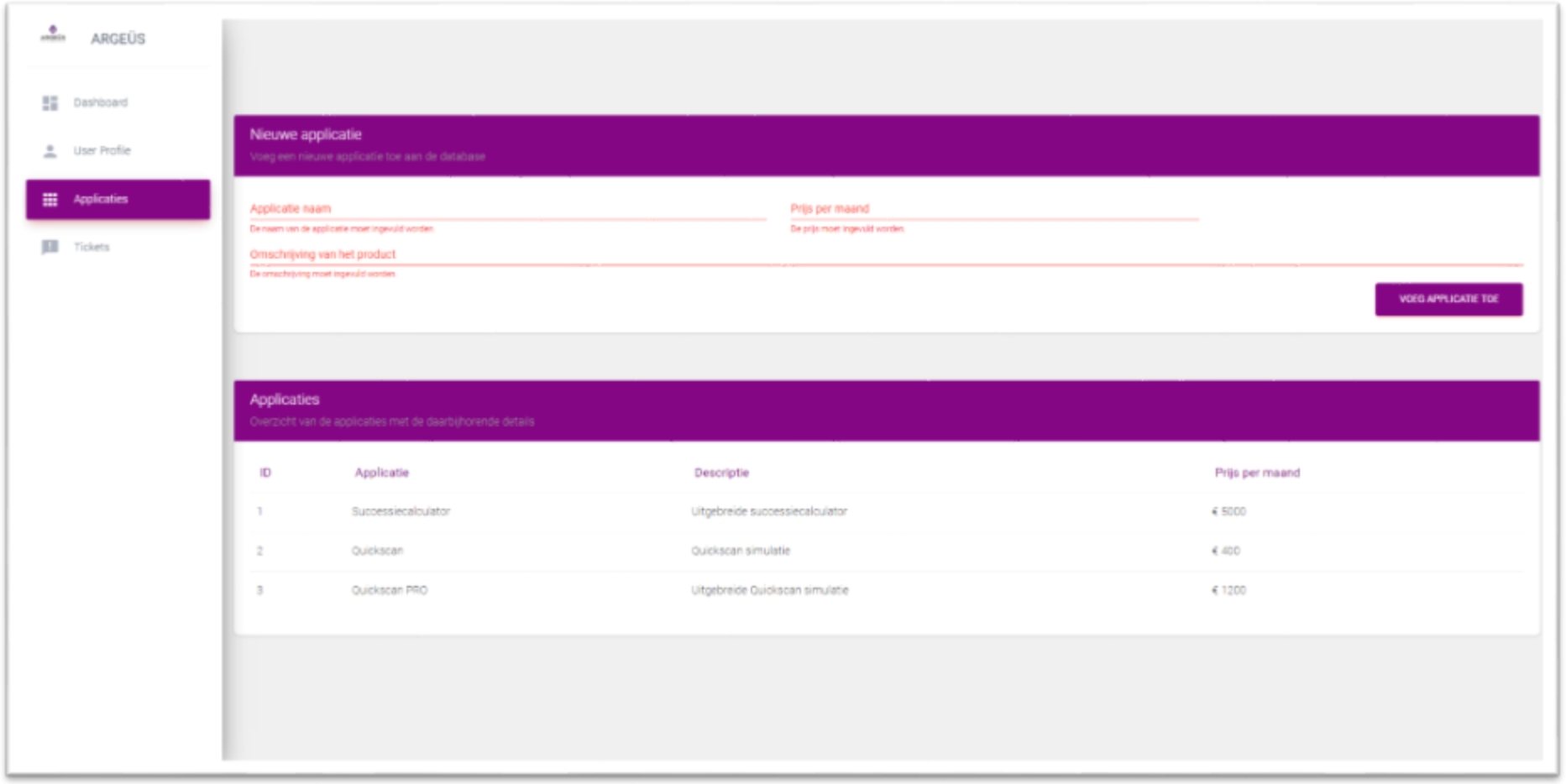
Figuur 32: Angular-prototype: Angular Material

In onderstaande afbeelding is te zien hoe er gebruik gemaakt wordt van Bootstrap om bepaalde html-elementen te stylen. Om met Bootstrap te stylen moet er aan een HTML-element een klasse gehangen worden. Deze klassen zorgen voor de opmaak van het element.

```
<div class="col-lg-4 col-md-6 col-sm-6">
    <div class="card card-stats">
        <div class="card-header card-header-warning card-header-icon" >
            <div class="card-icon" >
                <i class="material-icons" >
                    supervisor_account
                </i>
            </div>
            <p class="card-category">Gebruikers</p>
            <h3 class="card-title">Aantal: {{users.length}}</h3>
        </div>
        <div class="card-footer">
            <div class="stats">
                <a routerLink="/user-profile">Klik hier om de gebruikers te weergeven</a>
            </div>
        </div>
    </div>
</div>
```

Figuur 33: Angular-prototype: Bootstrap voorbeeld

3.4.2.3 Validatie



Figuur 34: Angular-prototype: applicatieoverzichtscherm

Validatie was ook één van de aspecten die Argeüs aanhaalde. Met Angular zijn er veel mogelijkheden met betrekking tot validatie. Er kunnen ongelimiteerd validatieregels toegevoegd worden aan een inputveld.

```
this.applicationForm = new FormGroup( controls: {
  applicationName: new FormControl( formState: '',  validatorOrOpts: [Validators.required, Validators.maxLength( maxLength: 24)]),
  description: new FormControl( formState: '',  validatorOrOpts: [Validators.required, Validators.maxLength( maxLength: 60)]),
  price: new FormControl( formState: '',  validatorOrOpts: [Validators.required, Validators.maxLength( maxLength: 60)]),
})
```

Figuur 35: Angular-prototype: Typescript validatiemogelijkheden

In bovenstaande figuur is de Typescript-code te zien die er voor zorgt dat er validatie is op de inputvelden.

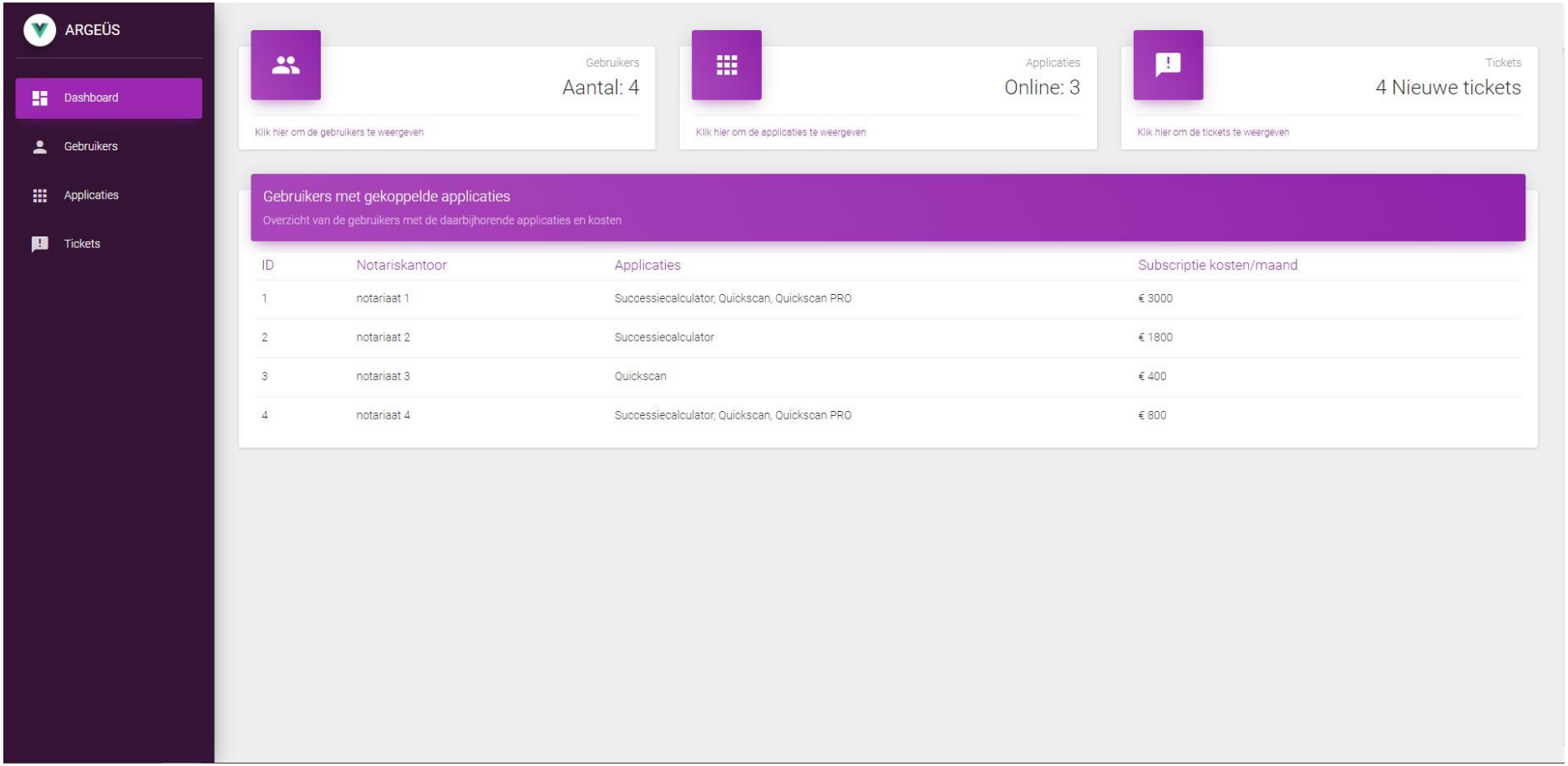
In de HTML van de component moeten er ook enkele regels toegevoegd worden. Door deze regels toe te voegen zal de gebruiker een gepaste foutmelding krijgen, zodat hij weet wat er mis is.

```
<mat-form-field class="example-full-width">
  <input id="applicationName" formControlName="applicationName" matInput placeholder="Applicatie naam" >
  <mat-error *ngIf="hasError('applicationName', 'required')">De naam van de applicatie moet ingevuld worden.</mat-error>
  <mat-error *ngIf="hasError('applicationName', 'maxlength')">De lengte van de naam mag max 60 characters bevatten.</mat-error>
</mat-form-field>
```

Figuur 36: Angular-prototype: HTML-validatieregels

3.4.3 VueJs prototype

Ook voor het prototype van VueJs is een dashboard uitgewerkt. Deze is bijna identiek aan de versie van Angular.



3.4.3.1 Opbouw van het prototype

Met Vue heeft de ontwikkelaar veel richtingen waar mee hij kan uitgaan wat betreft architectuur. In deze uitwerking is er gekozen om met de *component-based* architectuur te gaan werken. Deze is gelijk aan de architectuur van Angular. Er zijn *parent components* en *child components* gemaakt om zo een volwaardige pagina te genereren.

In onderstaande afbeeldingen is te zien hoe componenten in elkaar gebruikt kunnen worden.

```
<template>
    <div class="content">
        <div class="md-layout">
            <div class="md-layout-item md-medium-size-100 md-size-100">
                <edit-profile-form data-background-color="purple"></edit-profile-form>
            </div>
            <div class="md-layout-item md-medium-size-100 md-xsmall-size-100 md-size-100">
                <md-card>
                    <md-card-header data-background-color="purple">
                        <h4 class="title">Gebruikers toevoegen</h4>
                        <p class="category">Overzicht van de gebruikers</p>
                    </md-card-header>
                    <md-card-content>
                        <users-table table-header-color="purple"></users-table>
                    </md-card-content>
                </md-card>
            </div>
        </div>
    </div>
</template>

<script>
    import {EditProfileForm, UserCard} from "@/pages";
    import {UsersTable} from "@/components";

    export default {
        components: {
            EditProfileForm,
            UserCard,
            UsersTable
        },
    };
</script>
```

Figuur 37: Vue parent component voorbeeld

In afbeelding 34 is te zien hoe er gebruik gemaakt wordt van een *child component*. Hiervoor moet het *child component* enkel geïmporteerd worden. Daarna kan het component gebruik worden doorheen de *parent component*.

De *child component* heeft haar eigen HTML, CSS en Javascript. Door deze in een ander component te gebruiken zal de *child component* zijn gedeelte genereren. Op deze manier kan code goed gescheiden blijven en blijft de code goed overzichtelijk.

3.4.3.2 Design

Om tot dit design te komen is er gebruik gemaakt van Vue Material en Bootstrap. Er zijn een heel tal componenten gebruikt uit de Vue Material *library*. Een voorbeeld hiervan is bijvoorbeeld de knop die gebruikt wordt om een gebruiker toe te voegen. Verder zijn de elementen nog gestyled met behulp van Bootstrap.

```
<div class="md-layout-item md-size-100 text-right">
    <md-button class="md-raised md-success " data-background-color="purple"
               v-on:click="addItem(companyName, email, address, city, country, postalCode)">
      Voeg gebruiker toe
    </md-button>
</div>
```

Figuur 38: Vue Material button component

Hier is hoe het gegenereerde resultaat er zal uitzien in de frontend.

Figuur 39: Vue Material button

Als er bepaalde componenten niet aanwezig zouden zijn binnen Vue Material, kan er ook geïntegreerd worden met een ander framework zoals bijvoorbeeld React. Binnen het framework zijn er heel veel componenten aanwezig. Deze kunnen perfect gebruikt worden binnen het Vue framework.

3.4.3.3 Testmogelijkheden

Binnen Vue zijn er ook *libraries* om te gaan *unit testen*. In dit voorbeeld wordt er gebruik gemaakt van de test *library* 'Jest'.

Binnen deze test wordt er getest of een functie van een component wel degelijk een functie is. Het component dat getest moet worden dient geïmporteerd te worden. Daarna kan deze gebruikt worden om na te gaan wat het precies doet.

Het kan bijvoorbeeld zijn dat in een groot project iemand de functie per ongeluk verwijderd of aanpast. Als hij dan de testen zou runnen, zal hij te weten komen dat deze niet meer slaagt en weet hij wat hem te doen staat. De test moet terug slagen voor aleer dat de code *gepusht* mag worden naar de *repository*.

```
import {UsersTable} from "@/components";
import Vue from 'vue';

describe('UsersTable', () => {
    // Inspect the raw component options
    it('has a created hook', () => {
        expect(typeof UsersTable.get()).toBe('function');
    });
}
```

Figuur 40: Unit test in Jest

4 Discussie, aanbevelingen en conclusie

4.1 Welk framework wordt Argeüs aangeraden?

Er zijn verschillende frontendframeworks onderzocht. Er waren een aantal frontendframeworks die al goed overeenkwamen met de aspecten die Argeüs belangrijk vindt. Toch is er maar één framework dat er aangeraden wordt om te gebruiken in de toekomstige projecten. Het aangeraden frontendframework is Angular.

Angular is verkozen tot het beste framework voor Argeüs aan de hand van verschillende aspecten die hieronder beschreven worden.

Tijdens het onderzoek is er gevraagd welke aspecten de werknemers van Argeüs belangrijk vinden binnen een frontendframework. Angular kwam dan ook het beste overeen met deze aspecten. Zo vond bijvoorbeeld Argeüs de community en documentatie een grote rol spelen binnen een framework en dat zijn ook twee sterke punten van Angular.

Ook hadden alle ontwikkelaars binnen Argeüs al ervaring met Angular. Hier werd geen rekening mee gehouden tijdens het onderzoek maar het is altijd wel een voordeel als de ontwikkelaars al ervaring hebben met het framework. Hierdoor zullen de ontwikkelaars nog sneller van start kunnen gaan bij het ontwikkelen van een nieuw project. Ze zullen dus geen cursus gaan moeten volgen. Ze hebben al een basis en kunnen daar nog verder op bouwen.

Verder is Angular ook het framework dat het meest geschikt is voor grote projecten. Daarom is Vue bijvoorbeeld niet verkozen tot het ideale framework voor Argeüs. Tijdens het ontwikkelen van de prototypes werd het al snel duidelijk dat Vue snel onoverzichtelijk kan worden. Dit omdat er veel code op één plek kan zitten. Bij Angular is dit niet van toepassing, hier blijft alles goed gescheiden van elkaar en blijft het overzichtelijk. Er is hier rekeningen mee gehouden omdat Argeüs geen kleine projecten op de markt brengt. Het zijn altijd eerder grote projecten, waar gebruikers veel mee kunnen doen.

4.2 Aanbevelingen

Een logische volgende stap voor Argeüs zou het onderzoeken van *back end* frameworks zijn. Dit omdat Eclipse Scout 5 voor de *back end* en de frontend zorgt. Als ze dus van plan zijn om van frontendframework te veranderen zullen ze ook een nieuw *back end* framework moeten gaan zoeken.

4.3 Reflectie

Argeüs zat met de vraag welk frontendframework nu het meest geschikt is voor hun toekomstige projecten. Zelf was ik al heel geïnteresseerd in frontendframeworks omdat ik er nog niet veel mee bezig was geweest. Tijdens mijn stage periode ben ik zelfstandig en gemotiveerd begonnen met het onderzoeken van welk framework nu het beste bij Argeüs past.

Mijn taak was het opstellen van een onderzoek en het opzoek gaan naar het beste framework. Uiteraard heb ik ook af en toe wat hulp gekregen van mijn bedrijfspromotor tijdens deze uitwerking. Hij stuurde mij af en toe bij zodat ik enkel de nodige aspecten onderzocht.

Ik ben het onderzoek begonnen door allereerst te noteren wat de vraagstelling van het onderzoek nu precies was. Zo heb ik de scope van het onderzoek kunnen vastleggen. Daarna ben ik de methode van het onderzoek gaan vastleggen. Een goed onderzoek moet goed gestructureerd zijn. Op die manier wist ik precies wat ik ging doen en wanneer ik het ging doen. Ik had besloten het onderzoek in drie fases op te delen. Ik had me voorgenomen om in de eerste fase te gaan onderzoeken welke frameworks zich nu op de markt bevinden. Van deze frameworks ben ik dan een globale vergelijking gaan opstellen, waar later in het eindwerk nog naar gerefereerd kon worden. In fase twee ben ik de frameworks gaan filteren zodat er nog vijf frameworks overbleven. Dit had ik gedaan door na te gaan welke aspecten Argeüs nu echt belangrijk vind binnen een frontendframework. Van deze vijf frameworks ben ik dan de sterke en zwakke punten gaan onderzoeken. Door de sterke en zwakke punten te gaan onderzoeken ben ik tot twee frameworks gekomen. Ik ben tot deze twee frameworks gekomen omdat zij de sterkste aspecten hadden die overeenkwamen met die van Argeüs. Van deze twee frameworks ben ik dan een prototype gaan uitwerken waarmee ik een aantal aspecten praktisch kon toelichten aan Argeüs.

Zelf ben ik blij van het resultaat dat ik heb kunnen waarmaken. Ik kende nog niet zoveel frontendframeworks. Van veel had ik wel wat gehoord maar wist er niet echt iets van. Dat is nu veranderd. Door dit onderzoek te voeren is mijn zicht op frontendframeworks bijzonder hard vergroot. Ook heb ik wat van mijn programmeervaardigheden kunnen uitbreiden door de twee prototypes te maken.

Als ik dit onderzoek op nieuw zou moeten doen dan zou ik nog harder aan mijn prototypes gewerkt hebben. Ik weet van mezelf dat ik deze nog veel uitgebreider kon maken. Wel vond mijn bedrijfspromotor de prototypes die ik gemaakt had al voldoende.

Tijdens dit onderzoek heb ik altijd mijn best gedaan om zo een goed mogelijk resultaat te kunnen opleveren. Dit werd zeker geapprecieerd door mijn medewerkers binnen Argeüs.

Bibliografie

[1] Klu, „Nieuwsblad," [Online]. Available: https://www.nieuwsblad.be/cnt/dmf20160614_02339734. [Geopend 2019 03 14].

[2] S. Daityari, „Angular vs React vs Vue: Which Framework to Choose in 2019," [Online]. Available: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/. [Geopend 6 03 2019].

[3] A. Kumar, „React vs. Angular vs. Vue.js: A Complete Comparison Guide - DZone Web Dev," [Online]. Available: https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu. [Geopend 06 03 2019].

[4] TechMagic, „React vs Angular vs Vue.js - What to choose in 2019? (updated)," [Online]. Available: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d. [Geopend 04 03 2019].

[5] „frontendmasters," [Online]. Available: https://frontendmasters.com/books/front-end-handbook/2018/tools/apps.html. [Geopend 15 03 2019].

[6] A. Arkhypova, „yalantis," [Online]. Available: https://yalantis.com/blog/javascript-frameworks-how-to-make-your-choice/. [Geopend 15 03 2019].

[7] „Angular," [Online]. Available: https://angular.io/. [Geopend 15 03 2019].

[8] „reactJs," [Online]. Available: https://reactjs.org/. [Geopend 15 03 2019].

[9] „VueJs," [Online]. Available: https://vuejs.org/. [Geopend 15 03 2019].

[10] „Vaadin," [Online]. Available: https://vaadin.com/. [Geopend 15 03 2019].

[11] „EmberJs," [Online]. Available: https://emberjs.com/. [Geopend 15 03 2019].

[12] „Eclipse Scout," [Online]. Available: https://eclipsescout.github.io/8.0/technical-guide-js.html#overview. [Geopend 15 03 2019].

[13] „Aurelia," [Online]. Available: https://aurelia.io/. [Geopend 15 03 2019].

[14] „MithrilJs," [Online]. Available: https://mithril.js.org/. [Geopend 15 03 2019].

[15] „Riot," [Online]. Available: https://riot.js.org/. [Geopend 15 03 2019].

[16] „Polymer," [Online]. Available: https://www.polymer-project.org/. [Geopend 15 03 2019].

[17] „stackshare," [Online]. Available: https://stackshare.io/tools/top. [Geopend 15 03 2019].

[18] „opensource," [Online]. Available: https://opensource.org/licenses/MIT. [Geopend 29 03 2019].

[19] [Online]. Available: https://resources.whitesourcesoftware.com/blog-whitesource/top-10-apache-license-questions-answered. [Geopend 29 03 2019].

[20] „Apache," [Online]. Available: http://www.apache.org/licenses/LICENSE-1.0.txt. [Geopend 29 03 2019].

[21] „Opensource BSD-3-Clause," [Online]. Available: https://opensource.org/licenses/BSD-3-Clause. [Geopend 29 03 2019].

[22] Toptal, [Online]. Available: https://www.toptal.com/front-end/client-side-vs-server-side-pre-rendering. [Geopend 05 04 2019].

[23] storylens, [Online]. Available: https://storylens.com/@manjunath/server-side-vs-client-side-rendering---an-in-depth-overview. [Geopend 05 04 2019].

[24] „GitHub," [Online]. Available: https://help.github.com/en/articles/about-stars. [Geopend 29 03 2019].

[25] „Hotframeworks," [Online]. Available: https://hotframeworks.com/. [Geopend 3 05 2019].

[26] „Altexsoft," [Online]. Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/. [Geopend 10 05 2019].

[27] [Online]. Available: https://blog.techmagic.co/angular-2-vs-react-what-to-chose-in-2017/. [Geopend 10 05 2019].

[28] „Material Angular," Google, [Online]. Available: https://material.angular.io/components/input/overview. [Geopend 10 05 2019].

[29] „Karma," [Online]. Available: https://karma-runner.github.io/3.0/index.html. [Geopend 10 05 2019].

[30] „Tech Trends Showdown," [Online]. Available: https://medium.com/zerotomastery/tech-trends-showdown-react-vs-angular-vs-vue-61ffaf1d8706. [Geopend 10 05 2019].

[31] „Medium," [Online]. Available: https://medium.com/@vlascik/honest-look-at-ember-in-the-middle-of-2018-a0dc2787e506. [Geopend 15 05 2019].

[32] „Slant," [Online]. Available: https://www.slant.co/options/34/~ember-js-review. [Geopend 15 05 2019].

[33] „Stackshare," [Online]. Available: https://stackshare.io/emberjs#more. [Geopend 15 05 2019].

[34] „Voidcanvas," [Online]. Available: http://voidcanvas.com/prefer-ember-js-angular-react-js/. [Geopend 15 05 2019].

[35] „Ember-CLI," [Online]. Available: https://ember-cli.com/. [Geopend 15 05 2019].

[36] „Emberobserver," [Online]. Available: https://www.emberobserver.com/categories/ui-elements. [Geopend 15 05 2019].

[37] „EmberJs Guides," [Online]. Available: https://guides.emberjs.com/release/testing/. [Geopend 15 05 2019].

[38] „Pros and Cons of the Vue.js Framework," [Online]. Available: https://naturaily.com/blog/pros-cons-vue-js. [Geopend 15 05 2019].

[39] „Vue.js - Pros and cons," [Online]. Available: https://medium.com/@michealjroberts/vue-js-pros-cons-and-things-i-learnt-along-the-way-eb724a412113. [Geopend 15 05 2019].

[40] „What are the pros and cons of using Vue.Js," [Online]. Available: https://towardsdatascience.com/what-are-the-pros-and-cons-of-using-vue-js-3689d00d87b0. [Geopend 15 05 2019].

[41] „Pros and Cons of Vue framework," [Online]. Available: https://www.agriya.com/blog/pros-and-cons-of-vue-js-framework/. [Geopend 15 05 2019].

[42] „VueJs," [Online]. Available: https://vuejs.org/v2/guide/. [Geopend 15 05 2015].

[43] „VueJs," [Online]. Available: https://vuejs.org/v2/guide/unit-testing.html. [Geopend 15 05 2019].

[44] „Pros and Cons of ReactJS and React Native," [Online]. Available: https://xbsoftware.com/blog/price-of-popularity-pros-and-cons-of-reactjs-and-react-native/. [Geopend 16 05 2019].

[45] „The Pros and Cons of ReactJS for your Online business," [Online]. Available: https://curatti.com/pros-cons-reactjs/. [Geopend 16 05 2019].

[46] „Developing apps with React.js Pros and Cons," [Online]. Available: https://kinetiqsolutions.com/developing-apps-with-react-js-pros-and-cons/. [Geopend 16 05 2019].

[47] „React vs. Angular: The complete Comparison," [Online]. Available: https://programmingwithmosh.com/react/react-vs-angular/. [Geopend 16 05 2019].

[48] T. Froeyen, „Eindwerk_Froeyen_Toon," [Online]. Available: http://doks.pxl.be/doks/do/files/FiSe8ab2a8216440ce94016440fdb62b076e/Eindwerk_Froeyen_Toon.pdf?recordId=SEtd8ab2a8216440ce94016440fdb62a076d. [Geopend 16 05 2019].

[49] „Angular/React/Vue pros and cons," [Online]. Available: https://medium.com/@afonsobarros/angular-react-vue-pros-and-cons-75e161311e86.

[50] „Testing React Apps - Jest," [Online]. Available: https://jestjs.io/docs/en/tutorial-react. [Geopend 16 05 2019].

[51] „Aurelia as a next generation UI framework.," [Online]. Available: https://altabel.wordpress.com/2016/03/15/aurelia-as-a-next-generation-ui-framework-comparison-of-aurelia-angular-and-react-js/. [Geopend 17 05 2019].

[52] „Comparing Angular, Aurelia and React," [Online]. Available: https://www.ae.be/blog-en/comparing-angular-aurelia-react-js-framework/. [Geopend 17 05 2019].

[53] „Support | Aurelia," [Online]. Available: https://aurelia.io/support#live-support. [Geopend 17 05 2019].

[54] „Testing Components - Aurelia," [Online]. Available: https://aurelia.io/docs/testing/components#testing-a-custom-attribute. [Geopend 17 05 2019].

[55] „Ember-cli," [Online]. Available: https://ember-cli.com/ . [Geopend 28 05 2019].