

AFSTUDEERPROJECT  TOEGEPASTE INFORMATICA

Schooljaar 2019-2020

**Clustering zoekresultaten afkomstig van verschillende documentbeheerssystemen in een visuele omgeving**

***Auteurs:***

Arin Er

Stef Michiels

Stijn Saenen

Pieter-Jan Van Der Schueren

Inhoud

[Technische analyse 4](#_Toc34596542)

[Functionele analyse 8](#_Toc34596547)

[Plan van Aanpak 10](#_Toc34596548)

[Scope and Vision 11](#_Toc34596549)

[User Stories 14](#_Toc34596553)

[Story 01. contact lijst 14](#_Toc34596554)

[Story 02. options 14](#_Toc34596558)

[Story 03. netwerk 15](#_Toc34596561)

[Story 04. Node info 16](#_Toc34596564)

[Story 05. Legende 17](#_Toc34596567)

[Story 06. Relatie info 18](#_Toc34596570)

[Story 07. Tussenlaag 18](#_Toc34596571)

[Story mapping 19](#_Toc34596572)

[Domain Class Diagram 20](#_Toc34596573)

[Problemen en oplossingen 21](#_Toc34596574)

[Gebruikershandleiding 22](#_Toc34596575)

[Installatie handleiding 22](#_Toc34596576)

[De webpagina openen 22](#_Toc34596577)

[Zoekfunctie 23](#_Toc34596578)

[Visualisatie 24](#_Toc34596579)

[Legende 25](#_Toc34596580)

[Navigeren tussen nodes 25](#_Toc34596581)

[Installatiehandleiding 25](#_Toc34596582)

[Scripts 26](#_Toc34596583)

[Verslagen van meetings 30](#_Toc34596589)

[Donderdag 13/02: Meeting bij klant 30](#_Toc34596590)

[Donderdag 20/02: Meeting via Microsoft Teams 31](#_Toc34596591)

[Donderdag 27/02: Meeting bij klant 33](#_Toc34596592)

[Maandag 02/03: Meeting via Microsoft Teams 34](#_Toc34596593)

[Spike: Flask – Django 35](#_Toc34596594)

[Gannt-chart 38](#_Toc34596605)

[Bronnen 39](#_Toc34596606)

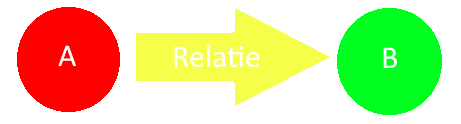
[Slot 39](#_Toc34596607)

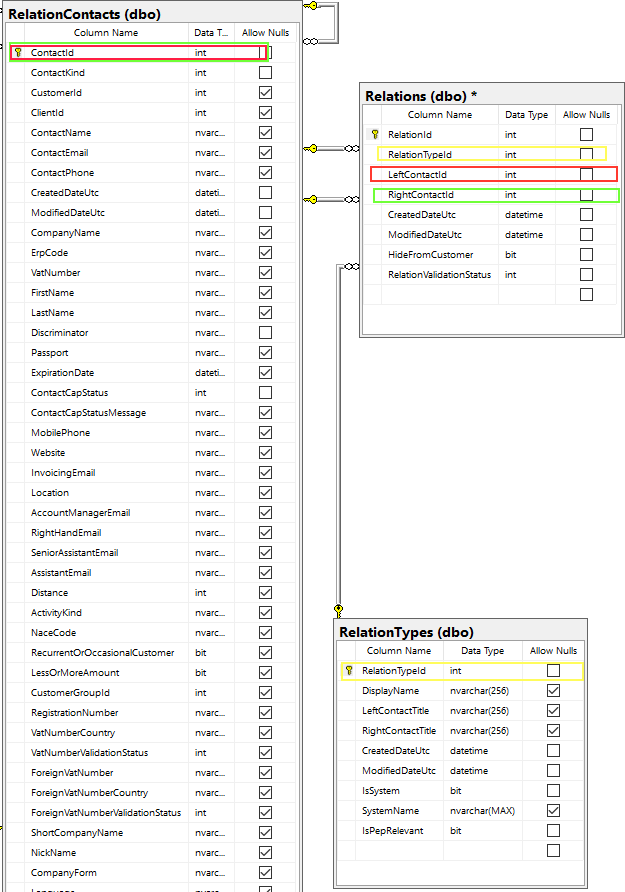
# Technische analyse

### In een SQL database (in de cloud) zijn al gedefinieerde relaties bekend. Reverse engineer deze SQL database om de relaties te begrijpen en visualiseer deze relaties.

select a.ContactName, t.LeftContactTitle, b.ContactName

from RelationContacts a inner join relations y ON (a.ContactId = y.LeftContactId) inner join RelationContacts b ON (b.ContactId = y.RightContactId) inner join RelationTypes t ON (t.RelationTypeId = y.RelationTypeId)





De daadwerkelijke visualisatie op het platform zou werken door relevante entiteiten van de DB af te halen en om te zetten in een JSON bestand, die gelezen kan worden door D3JS code die een visuele weergave zou genereren.

#### Minimum vereisten:

* Het visualiseren van de resultaten uit de relatie structuur

#### Overige vereisten:

* Het gebruiken van een key-woord dat niet in een tekst voorkomt als zoekterm.   
  (topicmodelling)

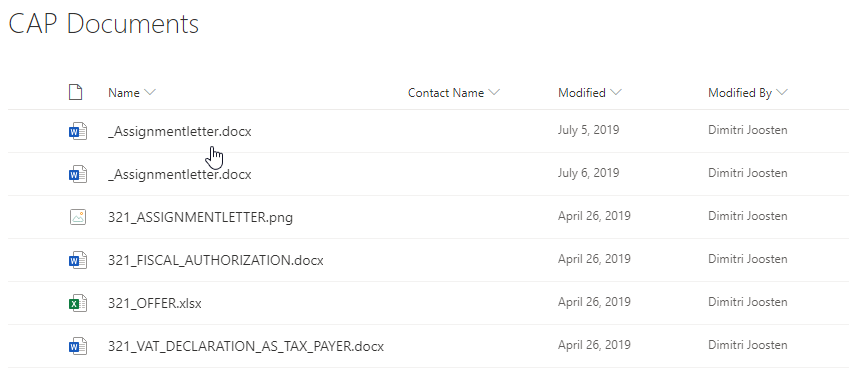
• Bij het zoeken rekening houden met de afgelegde weg en op basis hiervan documenten

voorstellen.

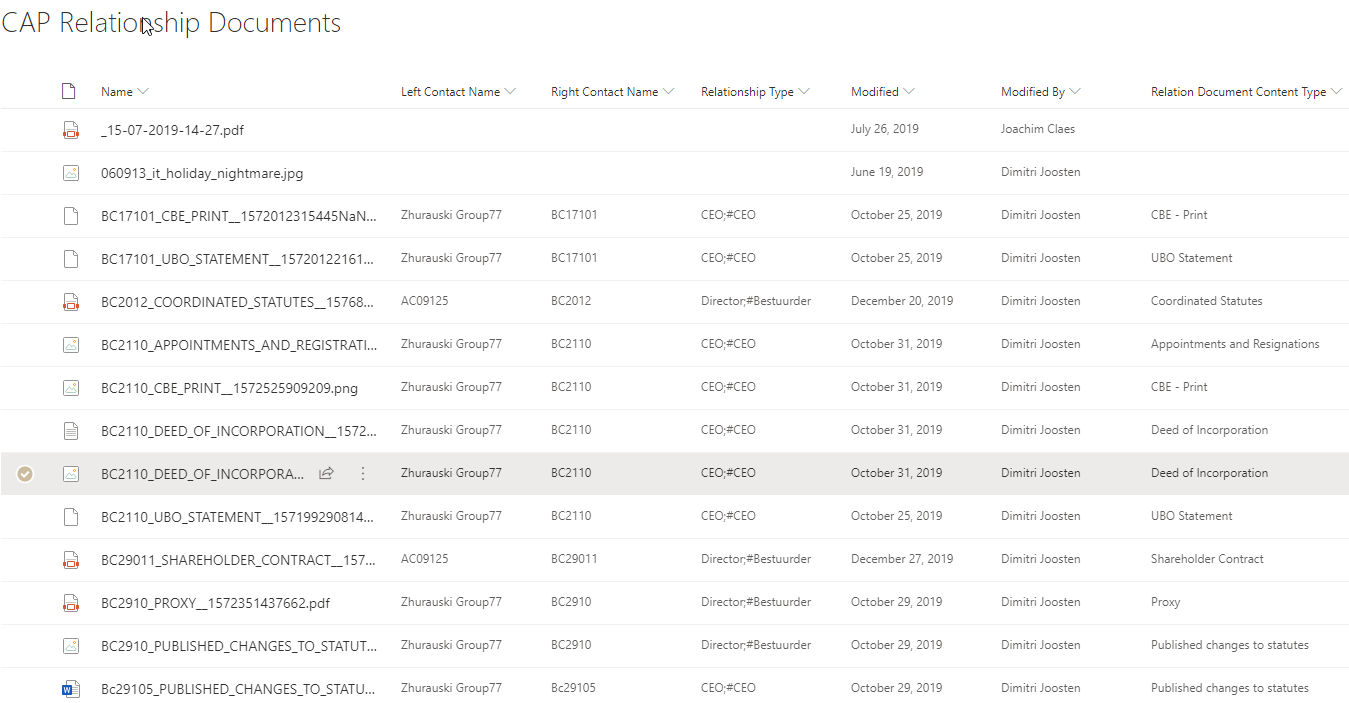
### 2. Door het combineren van een azure active directory & de SQL database is het mogelijk om te achterhalen wie welke toegangsrechten heeft voor bepaalde documenten. Achterhaal wie er toegang heeft tot welke documenten en visualiseer dit.

A) Documenten:

1. CAP Documents:



2. CAP Relationship Documents:



3. REST API

Door gebruik te maken van de sharepoint rest api zou het mogelijk moeten zijn om de CAP Documents en de CAP Relationship documenten in de database op te slaan met zijn gerelateerde informatie. <https://docs.microsoft.com/en-us/sharepoint/dev/sp-add-ins/complete-basic-operations-using-sharepoint-rest-endpoints>

#### Minimum vereisten:

* Het visualiseren van de toegangsrechten uit de azure active directory en SQL.
* Het ontwikkelen van een tussentaal die gebruikt kan worden voor de visualisatie en die op een efficiënte manier kan gaan zoeken tussen de documenten. Vergelijking tussen Neo4J en SQL

### 3. De SQL database is slecht opgebouwd en bevat 1 entry met een enorme Json string. Parse deze Json string zodat hier entiteiten uit gehaald kunnen worden. Sla deze entiteiten opnieuw op, zodat dit een volgend vertrekpunt kan zijn voor de visualisatie.

Soorten JSON-bestanden uit de tabel ContactCAPS:

1) Json met enkel het deel RISKValidation:  
<https://puu.sh/FayCa/7d3b0bff01.json>

2) Json met het deel Contact met constumer entiteit info

2a) JSON met Contact met als type family

<https://puu.sh/FayOd/4fa80b4c16.json>

2b) JSON met Contact met als type Business

<https://puu.sh/FayMA/3b3549e79b.json>

2c) JSON met Contact met als type Individual

<https://puu.sh/FayNo/9cb22bdd31.json>

3) Json met het deel contact, productinformatie, etc… zonder customer entiteit info

3a) JSON met Contact met als type family

<https://puu.sh/FayKm/51b96c0916.json>

3b) JSON met Contact met als type Business

<https://puu.sh/FayIo/0e3aa0846a.json>

3c) JSON met Contact met als type Individual

<https://puu.sh/FayJA/df7c4300b7.json>

Deze JSON strings moeten geparsed worden en omgezet worden naar database objecten, en op de overkoepelende database gezet worden.

#### Minimum vereisten:

* Een vergelijking tussen een SQL database en een Json data structuur.

#### Overige vereisten:

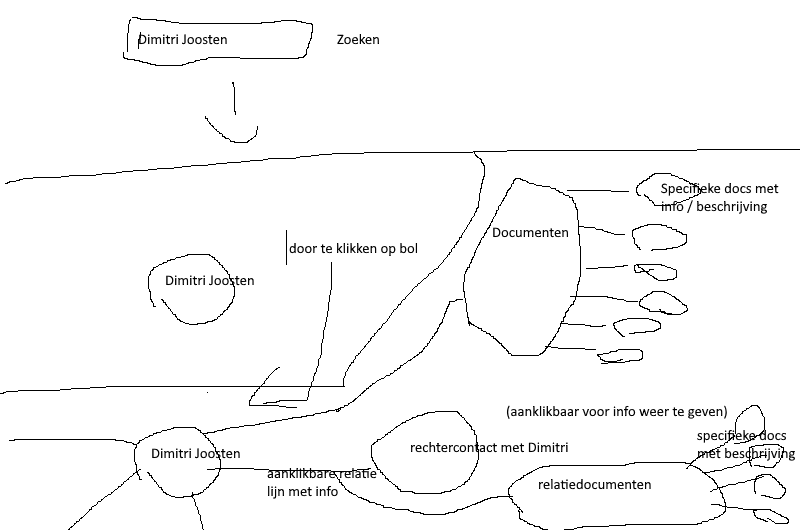
* Visueel antwoord kunnen geven op volgende vragen:

Welke klant gebruikt welke producten? (Informatie uit de Json string)

Welke klanten hebben een hoog risico? (Informatie uit de Json string)

Van welke klanten zijn de UBO/gecoördineerde statuten onvolledig?   
(Zeer gewaardeerd!) (meerdere DB)

### Vragen:

* Voor de visualisatie, van welke entiteiten vertrekken we?  
  Hoe moet het er uitzien?  
  
* \* In bovenstaande tekening kan je voor Relationship CAP documenten ook nog een tak met filtering voor documentstype omdat ze op de sharepoint al gefilterd zijn op documenttype. De documenten uit CAP documents hebbe deze typering echter niet, dus moeten we wachten op andere groepjes om deze date op type te ordenen en die daar op filteren?
* Is er een makkelijkere manier om de toeganginformatie op specifieke documenten te verkrijgen en weer te kunnen geven op ons portaal ipv ze met de rest api van sharepoint af te halen en ze in de databank stoppen?
* Welke informatie moet er weergegeven worden voor specifieke contactenteiten en documenten? Tot nu toe voor contacten ga ik uit van info zoals naam, email, bedrijf, contacttype, … Maar het zou erg helpen als dit gespecifieerd zou kunnen worden.

# Functionele analyse

“Van Havermaet heeft enorm veel (meerdere terabytes) ongestructureerde data. Een deel van deze data is manueel gestructureerd gemaakt, het grootste deel is dat echter nog niet. De opdracht van jullie bestaat er uit om een (Proof of Concept) PoC te ontwikkelen om op een visuele manier de juiste documenten terug te vinden. In een eerste fase is het de bedoeling om een tussentaal te definiëren zodat er efficiënt gezocht kan worden naar de juiste documenten en dat links tussen verschillende documenten gedefinieerd kunnen worden. In een tweede fase is het vervolgens de bedoeling om deze links te visualiseren en ervoor te zorgen dat er in deze visuele setting eenvoudig naar documenten gezocht kan worden. Bij onderstaand voorbeeld FOODPF.html is er een visualisatie, maar het zoeken naar de juiste informatie is niet gemakkelijk door o.a. leesbaarheid en hoeveelheid van links. Het is van belang om gedurende de gehele opdracht rekening te houden met de performantie aangezien het gaat om een enorme hoeveelheid aan documenten gaat.”

**Problemen:**

* Ongestructureerde data, waarvan een deel manueel ingevoerd
* Moeilijk om documenten terug te vinden
* Opvraging data
* JSON – String

**Oplossing:**

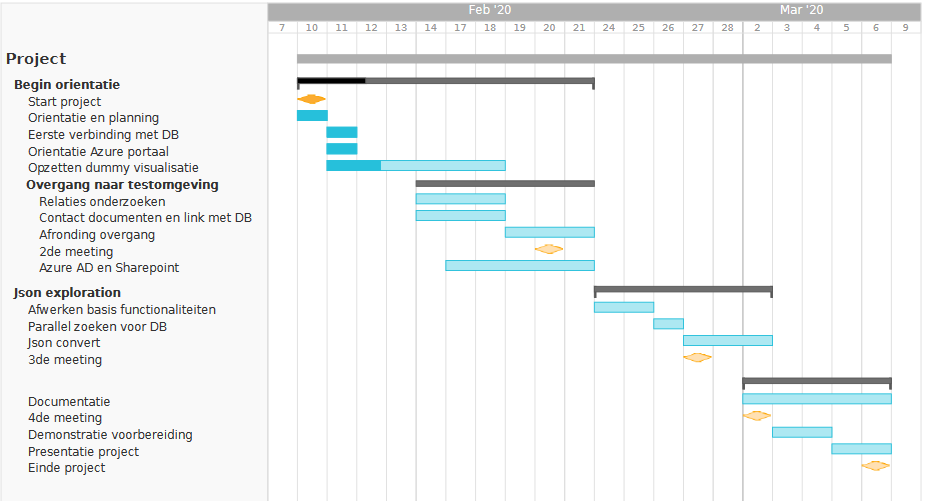
* Visualisatie op uniforme manier, met informatie over entiteiten en hun relaties
* Het visualiseren van de toegangsrechten uit de azure active directory en SQL.
* Inhoud van de JSON bestanden omzetten naar entiteiten en deze met samen met de originele sql databank objecten in een nieuwe databank plaatsen
* Informatie over bestanden ophalen via sharepoint API calls en in DB opslaan
* Het gebruiken van een key-woord dat niet in een tekst voorkomt als zoekterm. (topic modelling)
* Bij het zoeken rekening houden met de afgelegde weg en op basis hiervan documenten voorstellen.

**Technologieën:**

* Microsoft Azure: Verwerking van documenten
* Azure Active Directory
* Sharepoint: Opslag van de documenten
* Sharepoint API
* SQL & Json (en ETL)
* D3js voor visualisatie

**Groepsrollen:**

* Projectleider: Arin Er
* Contactpersoon: Arin Er
* Hoofdprogrammeur: Stef Michiels
* Database verantwoordelijke(n) Pieter-Jan Van Der Schueren, Stijn Saenen
* Sharepoint en AAD verantwoordelijke: Pieter-Jan Van Der Schueren
* Scrum-master: Stijn Saenen



# Plan van Aanpak

|  |  |
| --- | --- |
| **TODO** | **Uitgevoerd door** |
| Week1: |  |
| Info verkrijgen (Database credentials, ...) | Stijn en Arin |
| Platform voorbereiden | Pieter-Jan en Stef |
| Eerste call met klant | Groep |
| Kleine testprojectjes met dummy data | Groep |
|  |  |
| Week2: |  |
| Database analyse | Pieter-Jan en Stijn |
| JSON - interpreteren | Arin |
| Conceputalisatie van data clustering | Stef |
| Contact met klant (vragen en opmerkingen) | Groep |
|  |  |
| Week3: |  |
| Afwerken data analyse | Pieter-Jan en Stijn |
| Clustering | Stef en Arin |
| Visualisatie op platform | Pieter-Jan |
| D3JS learning curve / research | Arin |
| Kiezen visualisatie vorn | Stef |
| Contact met klant (vragen en opmerkingen) | Groep |
|  |  |
| Week4: |  |
| Afwerken rapporten | Stijn |
| Presentatie maken en voorbereiden | Arin en Stijn |
| Demo maken en voorbereiden | Pieter-Jan en Stef |
| Contact met klant (vragen en opmerkingen) | Groep |

# Scope and Vision

#### Background

Van Havermaet heeft enorm veel (meerdere terabytes) ongestructureerde data. Een deel van deze data is manueel gestructureerd gemaakt, het grootste deel is dat echter nog niet.   
De opdracht bestaat er uit om een (Proof of Concept) PoC te ontwikkelen om op een visuele manier de juiste documenten terug te vinden. In een eerste fase is het de bedoeling om een tussentaal te definiëren zodat er efficiënt gezocht kan worden naar de juiste documenten en dat links tussen verschillende documenten gedefinieerd kunnen worden. In een tweede fase is het vervolgens de bedoeling om deze links te visualiseren en ervoor te zorgen dat er in deze visuele setting eenvoudig naar documenten gezocht kan worden. Het is van belang om gedurende de gehele opdracht rekening te houden met de performantie aangezien het gaat om een enorme hoeveelheid aan documenten gaat..

#### Business Opportunity

Concreet wordt het volgende verwacht:

1. In een SQL database (in de cloud) zijn al gedefinieerde relaties bekend. Reverse engineer

deze SQL database om de relaties te begrijpen en visualiseer deze relaties.

2. Door het combineren van een azure active directory & de SQL database is het mogelijk om te achterhalen wie welke toegangsrechten heeft voor bepaalde documenten. Achterhaal wie er

toegang heeft tot welke documenten en visualiseer dit.

3. De SQL database is slecht opgebouwd en bevat 1 entry met een enorme Json string. Parse

deze Json string zodat hier entiteiten uit gehaald kunnen worden. Sla deze entiteiten

opnieuw op, zodat dit een volgend vertrekpunt kan zijn voor de visualisatie.

#### Business Improvement Objectives

BO-1: Visualiseer de verschillende contactpersonen en hun relaties.

BO-2: Combineer informatie vanuit verschillende bronnen zodat alle informatie over de contactpersonen en hun gerelateerde documenten en andere data overzichtelijk gegroepeerd zijn.

BO-3: Verbind contactpersonen met producten via de json string in de database tabel ContactCaps.

#### Success Metrics

SM-1: Maak een applicatie ter visualisatie van de gegevens op een overzichtelijke manier, volgens de wensen/verwachtingen van de klant.

SM-2: Maak de applicatie makkelijk uitbreidbaar en maak het makkelijk om de applicatie op te starten en te gebruiken.

#### Vision Statement

Om het mogelijk te maken dat we deze verschillende bronnen van data kunnen visualiseren met 1 tussenlaag, gaan we een kopie van de originele database maken en daar data van andere bronnen, bv de active directory, aan toevoegen, zodat we deze kunnen linken aan de overeenkomstige contactpersonen uit de database.

#### Business Risks

RI-1: Als ter te veel data in 1 keer wordt weergegeven of opgeslagen in het geheugen van de applicatie kan dit de performantie van de applicatie beschadigen.

RI-2: Als er te veel data is om te visualiseren wordt het zeer moeilijk om dit overzichtelijk te houden.

RI-3: Er moet op een correcte manier de data op de database en data van andere bronnen gelinkt worden zodat de data op een correcte manier gestructureerd is en niet onder een verkeerde persoon wordt weergegeven door bv. Een foute query.

### Scope and Limitations

#### Major Features

FE-1: Visualisatie relaties, personen, documenten, beveiligingsinformatie, …

#### Stakeholders profiles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stakeholder | Major Value | Attitudes | Major Interests | Constraints |
| Opdrachtgever Benny Westaedt | Groepering van informatie van verschillende bronnen en duidelijke visualisatie. | De applicatie moet duidelijk zijn in de bronnen van de informatie en de informatie moet duidelijk weergegeven worden. | De applicatie moet zorgen als goede basis voor het verzamelen en groeperen van de verschillende gegevens bij Van Havermaet. | De applicatie moet de data op een performante manier kunnen verwerken en visualiseren. |

.

### 3.Software requirements specifications

**3.1 Users and Characteristics**

**User**

Er zijn geen verschillende soorten gebruikers gedefineerd in onze opdracht.  
Iedere gebruiker is dus met andere woorden hetzelfde voor deze opdracht.  
De gebruiker moet in staat zijn om de applicatie meteen te kunnen begrijpen en gebruiken.

#### 3.2 Operating Environment Constraints

OE-1: De webapplicatie moet makkelijk integreerbaar zijn op een webserver of andere omgeving.

#### 3.3 Design and Implementation Constraints

CO-1: Er moet een legende zijn om de visualisatie uit te leggen.

CO-2: De visualisatie moet overzichtelijk zijn.

CO-3: Al de meest belangrijke informatie moet weergegeven worden.

### 4.External Interface Requirements

#### 4.1 User Interfaces

UI-1: De user interface voor de applicatie zal zijn in de vorm van een simpele web applicatie, met een zoekfunctie, visualisatie gedeelte en wat tekstvelden met informatie.

#### Software Interfaces

SI-1: Web applicatie

SI-1.1: Heeft een zoekfunctie

SI-1.2: Toont de relaties van een persoon en informatie over de contactpersonen

#### 4.3 Communications Interfaces

CI-1: Er is regelmatig email verkeer met contactpersonen van Van Havermaet en onze begeleiders van de school.

CI-2: Er is een wekelijkse teams meeting met onze contactpersonen bij Van Havermaet en onze schoolbegeleiders.

CI-3: Onze schoolbegeleiders komen elke dag een paar keer langs.

# User Stories

## Story 01. contact lijst

Als een user

Wil ik een contactpersoon kunnen opzoeken aan de hand van hun naam

Zodat ik een persoon kan selecteren waarvan ik zijn/haar relaties van wil zien

### Technical details:

* De lijst moet gebruik maken van autocompletion

### Wireframes (optional):



### Acceptance criteria:

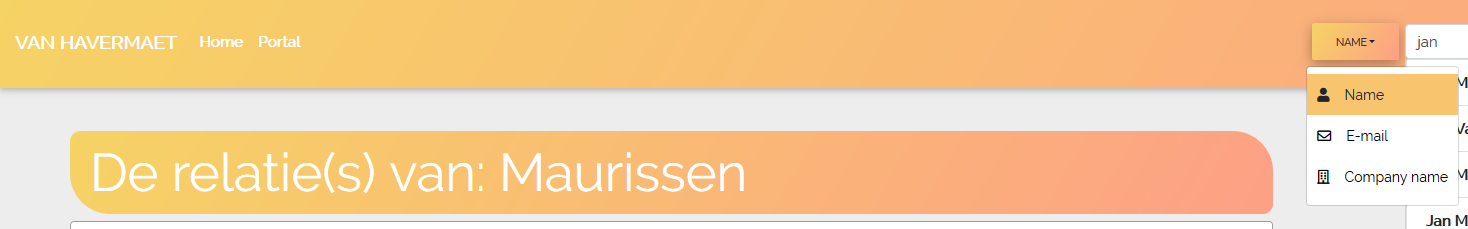
* Van de home pagina wil ik kunnen navigeren naar de ‘Van Havermaet’ website
* Van de home pagina wil ik kunnen navigeren naar het Van Havermaet’ portaal

## Story 02. options

Al seen user

Wil ik een contactpersoon willen kunnen opzoeken aan de hand van zijn naam, email en Bedrijfsnaam

### Wireframes (optional):



### Acceptance criteria:

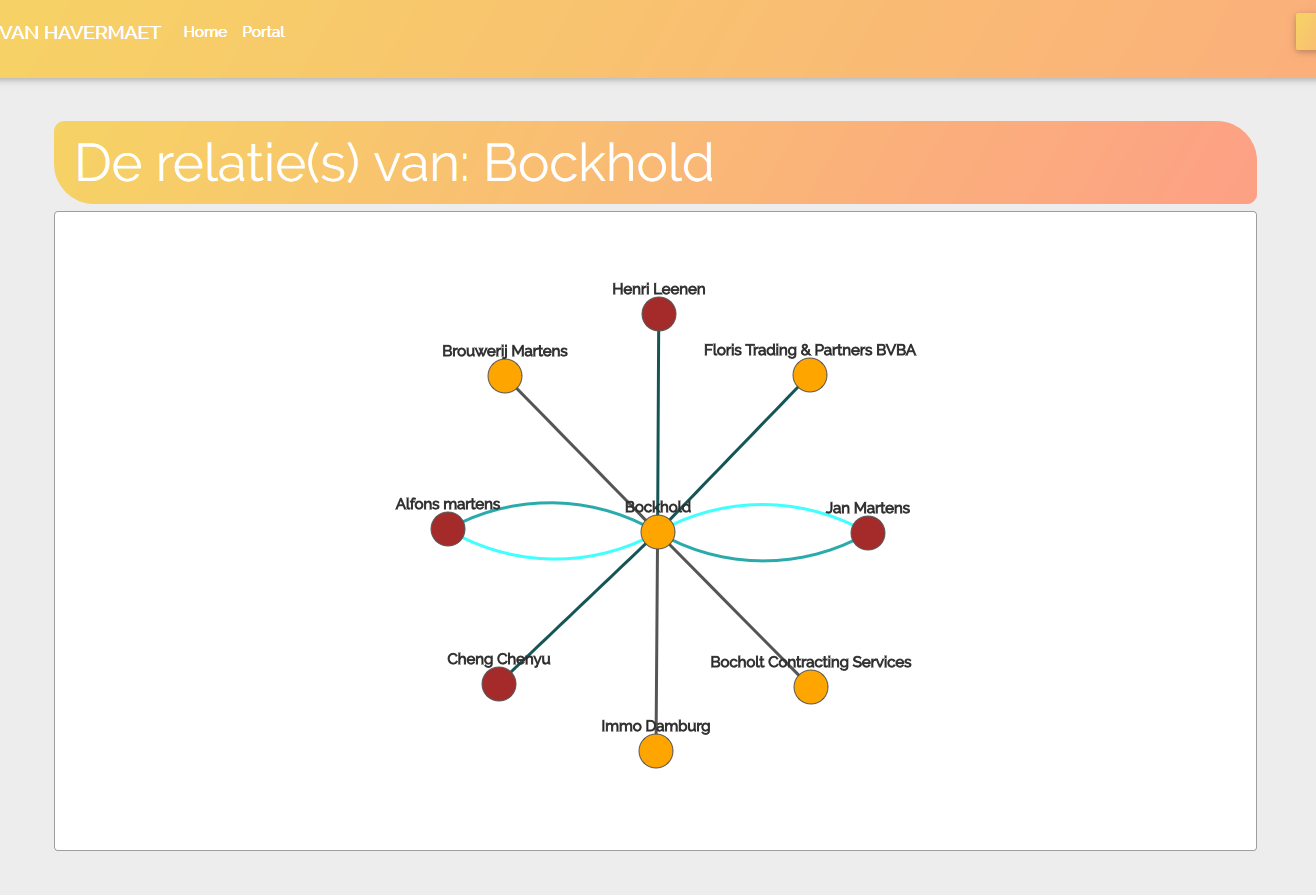
* Eens de selectie is gemaakt moet deze geactiveerd blijven tot de gebruiker deze zelf wisselt (toggle)

## Story 03. netwerk

Als een gebruiker

Wil ik het netwerk van relaties zien van een persoon wanneer ik op een persoon zijn naam druk in de lijst

### Wireframes (optional):



### Acceptance criteria:

* Elke verschillende soort relatie moet een andere kleur krijgen
* Elke verschillende soort node moet een andere kleur krijgen
* De naam van de contactpersoon node moet boven de node verschijnen

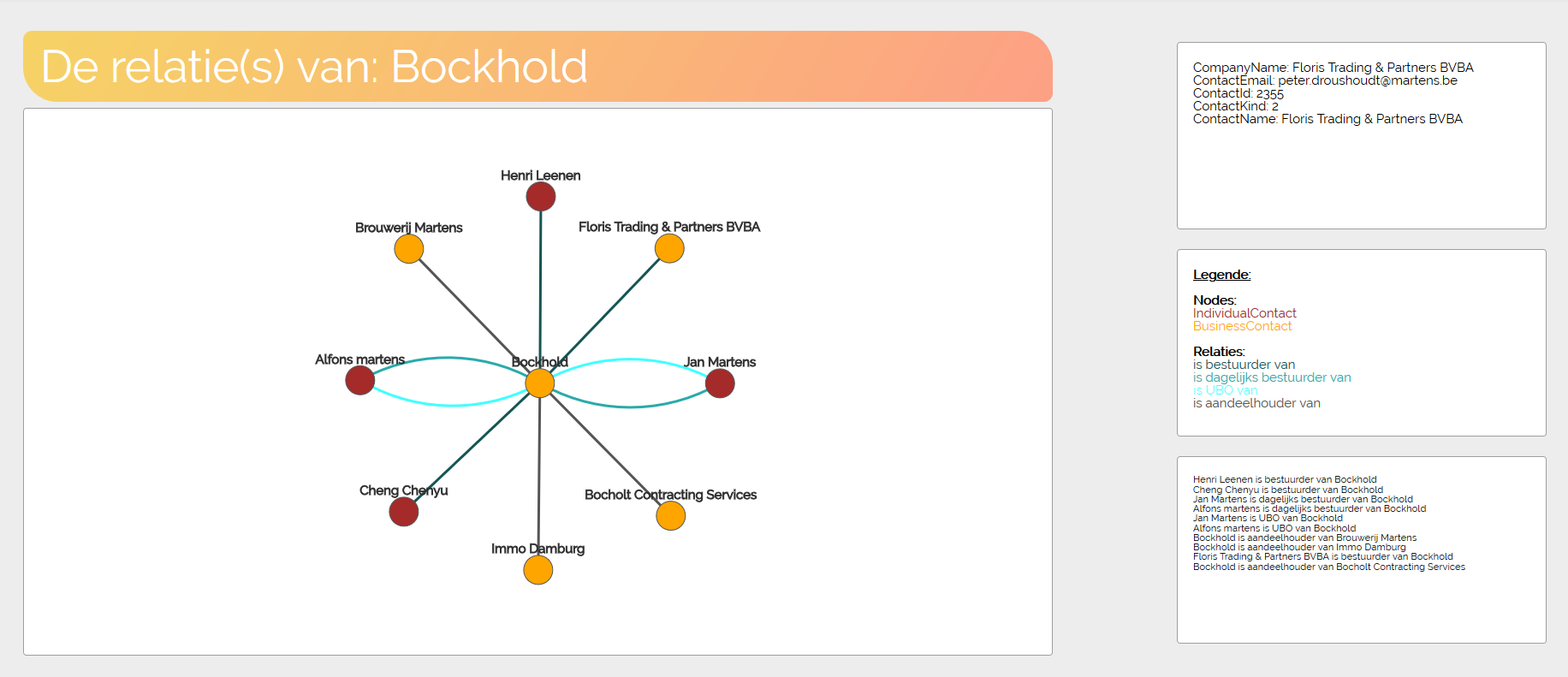
## Story 04. Node info

Als een gebruiker

Wil ik over een node kunnen hoveren

Zodat ik alle overige info van deze node te zien krijg

### Wireframes (optional):



### Acceptance criteria:

* Deze info moet in een box naast het netwerk verschijnen

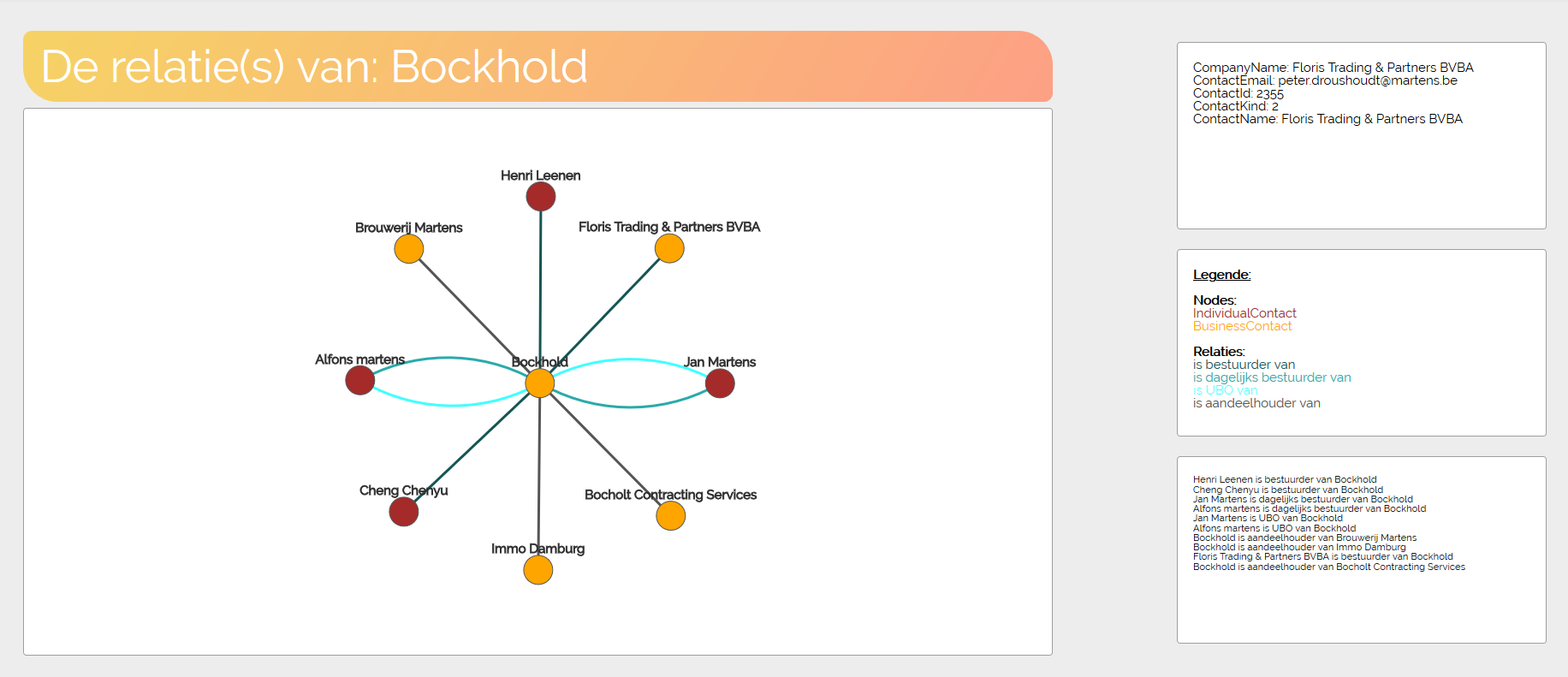
## Story 05. Legende

Als een gebruiker

Wil ik een legende zien van alle kleuren in het netwerk

Zodat ik weet wat elke kleur betekend in het netwerk

### Wireframes (optional):



### Acceptance criteria:

* De kleuren van de nodes en de relaties moeten gesplitst zijn

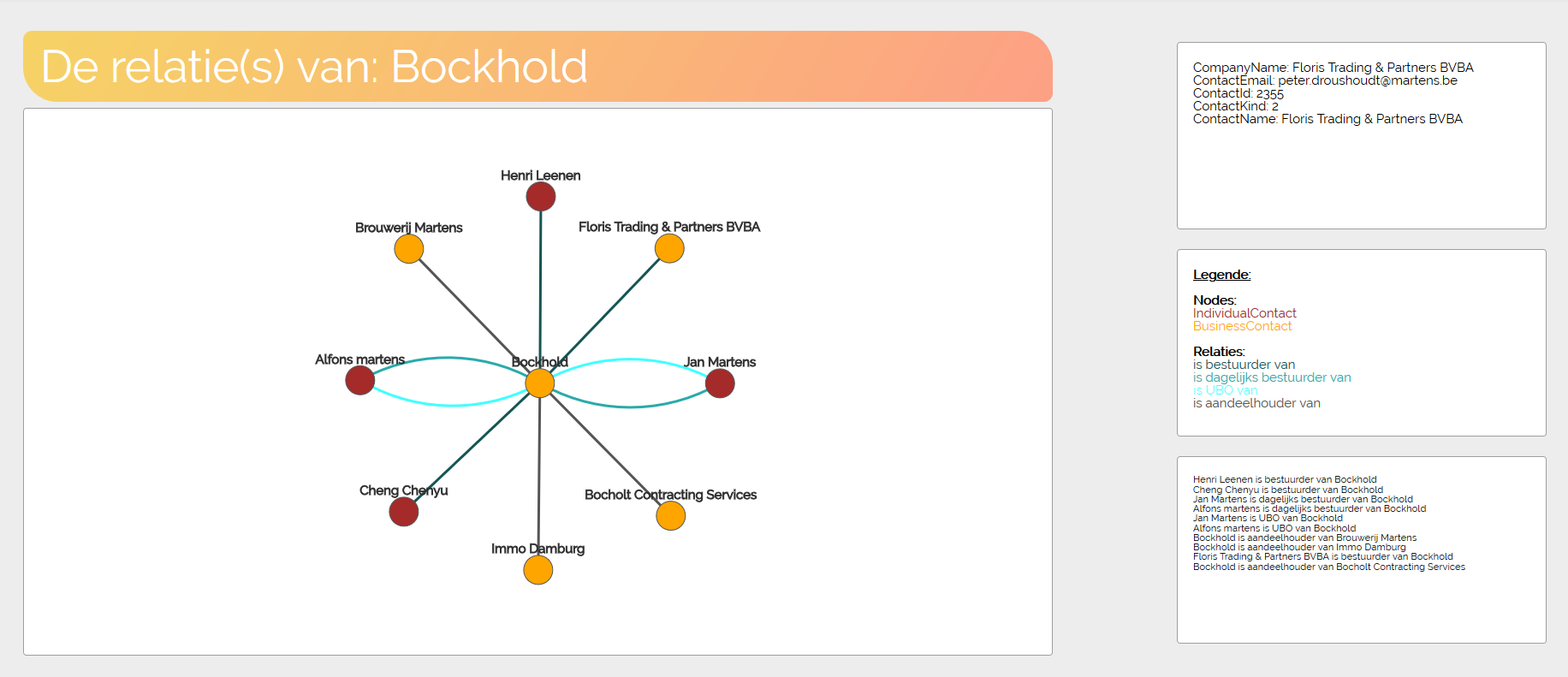
## Story 06. Relatie info

Als een gebruiker

Wil ik meer info over alle relaties zien

Zodat ik de richting van de relaties kan afleiden

#### Wireframes (optional):



#### Acceptance criteria:

* Deze info moet getoond worden in een box naast het netwerk

## Story 07. Tussenlaag

Als de back-end

Wil ik alle data die beschikbaar is kunnen raadplegen aan de hand van een tussenlaag met alle data er in

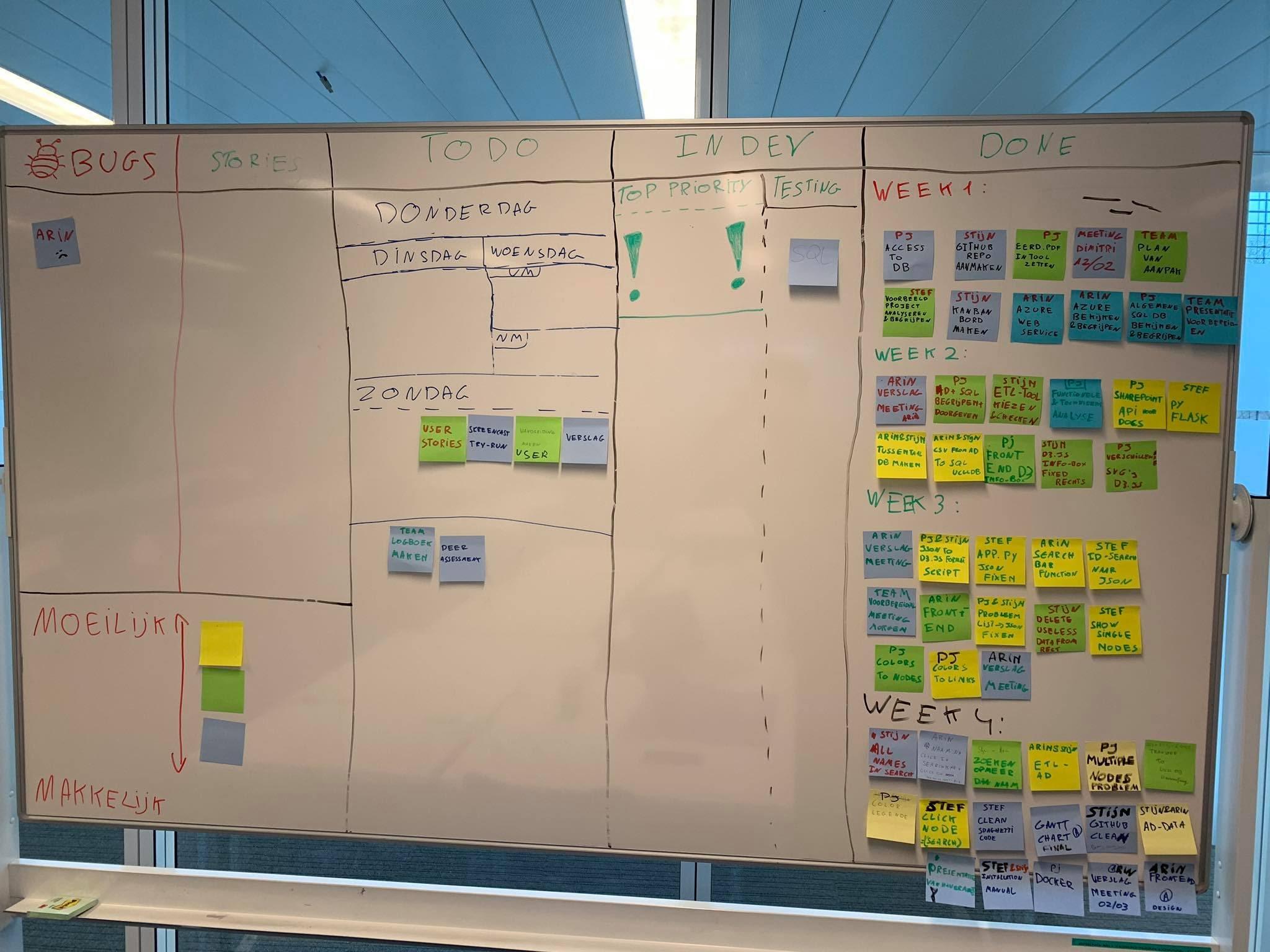
Zodat ik relaties kan creëren die de bronnen overschrijden

#### Technical detail

* Gebruik ETL voor de conversion

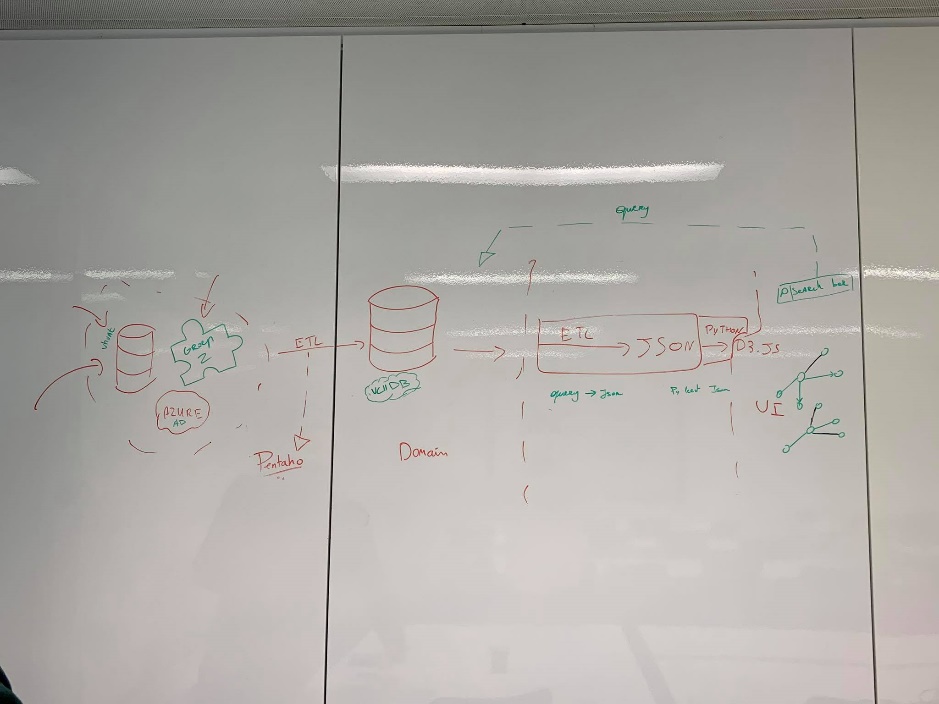
# Story mapping

Op de onderstaande foto is ons kanbanbord te zien met elke taak die we hadden gecreëerd. De taken waren met een kleurcode gerangschikt met een moeilijkheidsgraad. Je kan op de foto zien in welke week deze vielen. als je deze combineert met de Gantt chart zie je perfect welke acties er wanneer vielen en wie dat deze uit ging voeren. De onderverdeling van dinsdag, woensdag en donderdag ontstond er in de laatste week toen we specifiek moesten gaan plannen voor de presentatie.



# Domain Class Diagram

We hebben als groep geen gebruik gemaakt van een domain met classes erin. We hadden echter wel een schema voor hoe ons project conceptueel in elkaar zat.

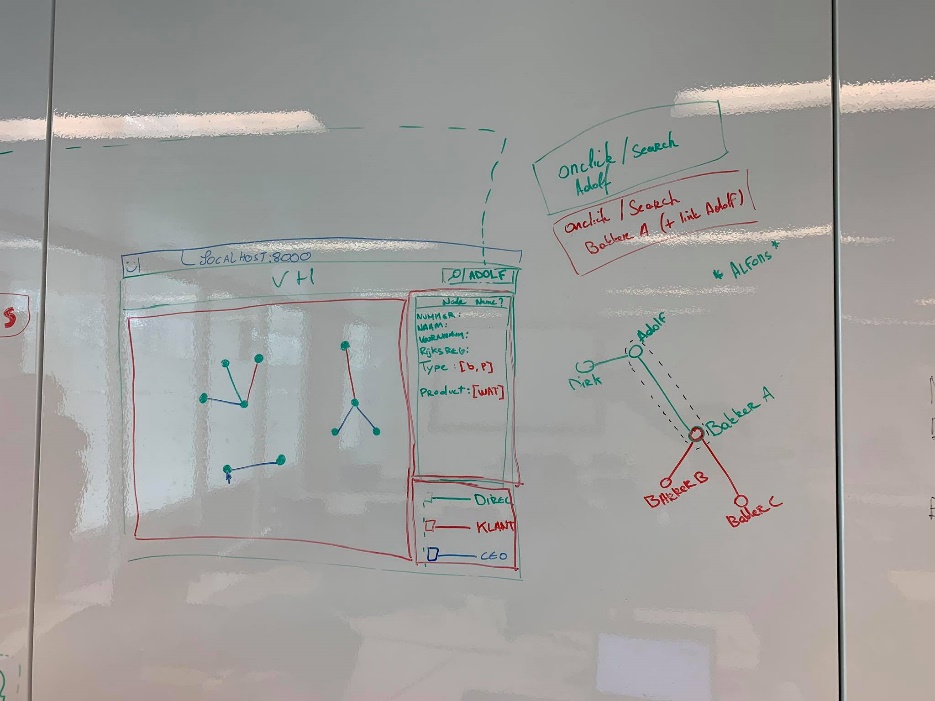


4

3

2

1



4

1. Was het startpunt van ons project. Alle verschillende bronnen van informatie die onderling relaties hadden. Hier konden nog bronnen aan toegevoegd worden of wegvallen.
2. Was de universele tussenlaag dat er voor ging zorgen dat alles op een uniforme manier wordt opgeslagen en er op dezelfde manier gezocht kan worden naar al de info. Zo kunnen relaties over verschillende bronnen ook ontstaan.
3. Was de call van de backend naar de tussenlaag die dan de uitkomst van de call ging omzetten naar een formaat dat de visualisatiesoftware kan begrijpen.
4. Was het eigenlijke visualiseren van de relaties aan de hand van D3.js en het netwerk dat we tekenen op die webpagina.

# Problemen en oplossingen

**Probleem**: Active Directory en contacten uit DB met elkaar linken  
**Oplossing**: Met een powershell script de informatie uit de AD ophalen in CSV en dan met ETL tool op de database zetten en dan de informatie via queries met elkaar verbinden.  
  
**Probleem**: documenten van contactpersonen kunnen lezen & groeperen onder de jusite contactpersonen, connectie met sharepoint & documentinformatie afhalen?  
**Oplossing**: authenticatie met postman op de sharepoint omgeving configureren voor post en get requests te kunnen maken op de sharepoint omgeving

**Probleem**: de visualisatie code van de network.js wordt in voorbeelden altijd getoond door de links te maken op basis van de positie van de entiteiten.  
**Oplossing**: herschrijf de code om te werken op basis van de ID van de source en target van de links.  
  
**Probleem**: er waren verscheiden problemen met het parsen van de JSON, onder andere:  
De d3js code erkent onze json in memory niet als valide json, toont altijd undefined of null waarden. Sommige velden uit de database werden niet correct omgezet in json.

**Oplossing**: weghalen van een stuk code dat overbodig was voor onze implementatie dat voor porblemen zorgde. & Filtering van json velden.

**Probleem:**

Met de ETL waren er heel veel problemen die we moeilijk konden voorspellen. Er waren veel specifieke problemen met de ETL tool zelf. De rest van de problemen lagen aan de databank zelf die we hadden opgesteld in de tussenlaag. Als eerste probleem kwamen we veel problemen tegen met de tool omdat we deze nog moesten ontdekken. Het probleem met de ETL tool was dat we batch inserts niet mochten aanvinken tijdens het inserten van de data van Van Havermaet in de tussenlaag. Dit hebben we opgezocht door de specifieke error te googelen. Stackoverflow hielp veel.

**Probleem:** Het volgende probleem was het inserten van de id’s in de tables van de tussenlaag afkomstig van de databank van Van Havermaet. De tables die oorspronkelijk werden gemaakt hadden de optie uitgeschakeld om manueel id’s te inserten. Deze optie aanzetten lukte niet na lang proberen dus het aanzetten van de optie moest door de tabel volledig opnieuw aan te maken met de juiste constraints.

**Probleem:** Het derde probleem was het dynamisch maken van de visualisatie. Dit kwam doordat de eerste versie van de backend niet aangepast was om te werken met requests en responses. We werkten eerst met een python script die de data uit de tussenlaag haalde, transformeerde naar geldige JSON en daarna wegschreef naar een bestand die dan werd ingelezen door de visualisatiecode. Een aanzienlijk nadeel is dat je in dit geval altijd alle data naar de front end moet sturen, en dat je geen asynchrone web applicatie hebt.

**Oplossing:** Bij het zoeken naar een oplossing hebben we ontdekt dat er voor python 2 gevestigde web framworks zijn, Flask en Django. Na een afweging tussen de twee (zie Spike: Flask - Django) hebben we besloten om voort te gaan met het Flask framwork. Nu waren we verplicht de hele backend te herschrijven, dit loonde gelukkig want het Flask framwork bleek ideaal onze noden te voldoen en hielp ons de concepten die we in gedachte hadden te realiseren.

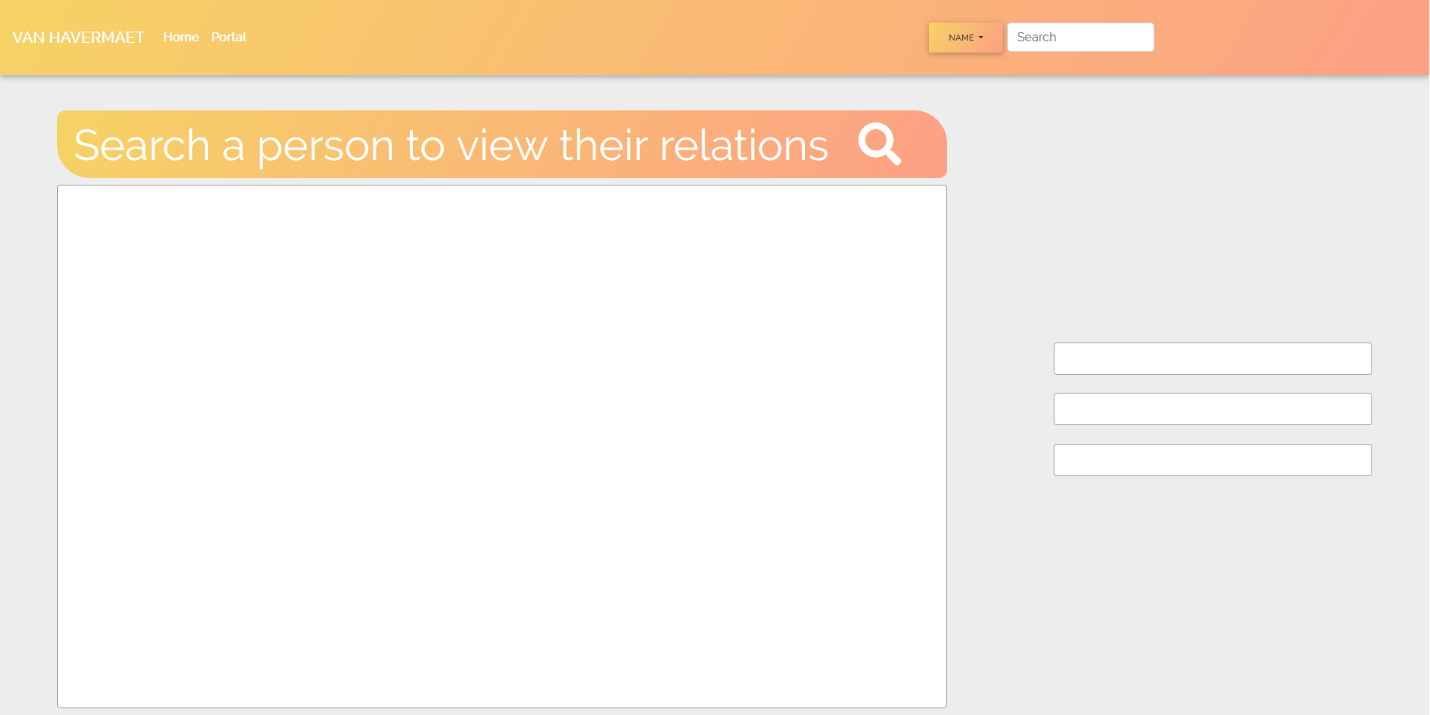
# Gebruikershandleiding

## Installatie handleiding

In deze handleiding gaan we niet bespreken hoe we onze applicatie en tussenlaag installeren en ervoor zorger dat men deze kan gebruiken/uitbreiden. Als u hierin geïnteresseerd bent, verwijs is graag naar onze repository waar dit mooi is uitgeschreven.

<https://github.com/stijnsaenen/Afstudeerproject>

## De webpagina openen

Wanneer u via u browser naar de webpagina surft, zal u de landingpage zien. Dit is de pagina waarop u terecht komt als u surft naar de webpagina. Vanaf hier heb je de keuze om via de navigatiebalk naar een andere webpagina te gaan, of via de zoekbalk een persoon op te zoeken om hiervan zijn relaties te bekijken.

3

2

1

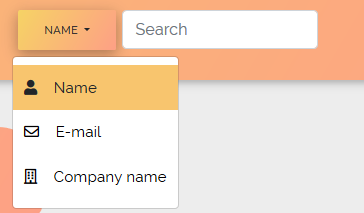
1. Als men klikt op de eerste link met tekst ‘Van Havermaet’, wordt de webpagina gewoon herladen en kom je terug terecht op de landingpage (= de pagina die hierboven staat)

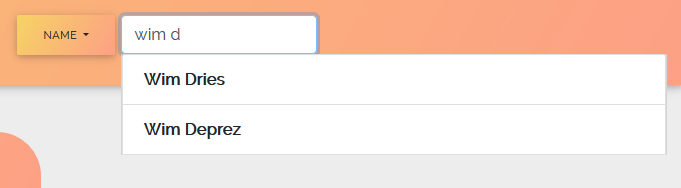
2 Als je klikt op de ‘Home’ link in de navigatiebalk, kom je terecht op de website van van ‘Van Havermaet’.

3 Als je klikt op de link ‘Portal’, word je gestuurd naar de webpagina van het portal van ‘Van Havermaet’. Hierop kan je al de documenten vinden die men de klanten van ‘Van Havermaet’ te maken hebben.

## Zoekfunctie

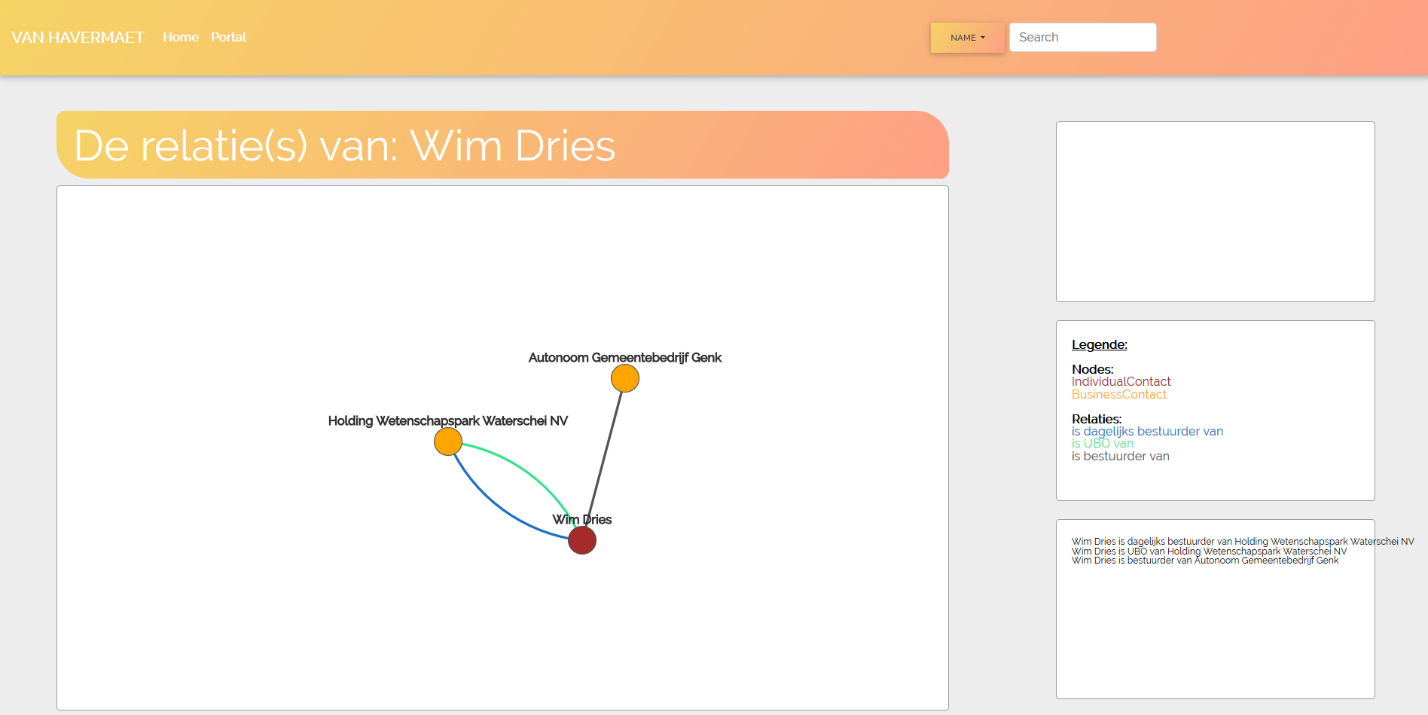
Het volgende wat je kan doen op de webpagina is het zoeken van personen om zijn/haar relaties weer te geven. Dit kan gedaan worden door gebruik te maken van de zoekfunctie. Je kan zoeken op 3 verschillende dingen: naam, e-mail en bedrijfsnaam. Je kan van verschillende opties veranderen door op onderstaande knop te klikken en in de lijst te selecteren op wat je wilt gaan zoeken.

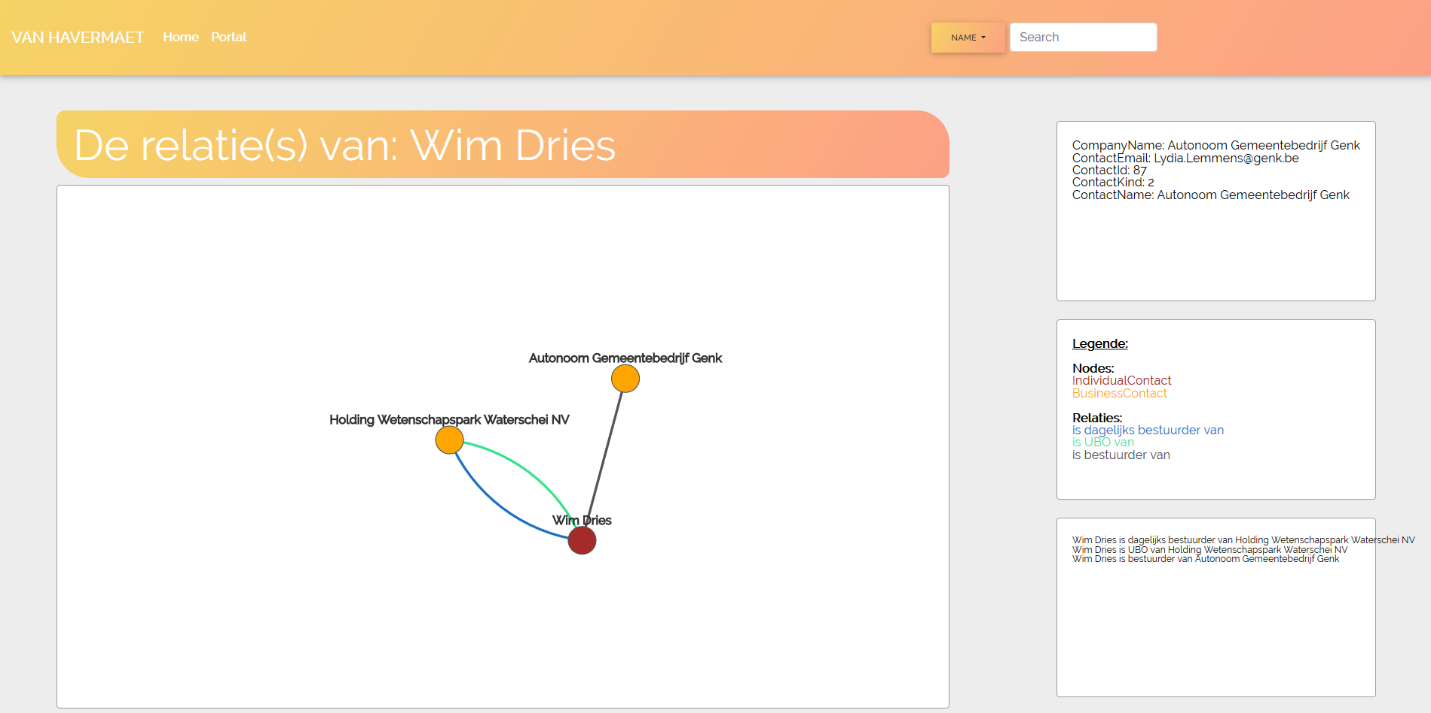


Als wij bijvoorbeeld op naam willen gaan zoeken, klik je in bovenstaande lijst op de optie ‘Name’. Vanaf dan is het mogelijk om in de zoekbalk via de naam van een persoon, naar deze persoon te gaan zoeken om zijn relaties weer te geven. Bij e-mail en bedrijfsnaam werkt dit op volledige zelfde manier. In dit voorbeeld gaan we de relaties opzoeken van persoon: “Wim Dries”. Eens je 3 letters, van de persoon die je wilt zoeken, hebt ingegeven in het invoerveld, komen er allerlei opties opgelijst onder de zoekbalk. Dit zijn de opties die overeenkomen met wat jij hebt ingeven in de zoekbalk, en helpen u om de juiste persoon aan te klikken.

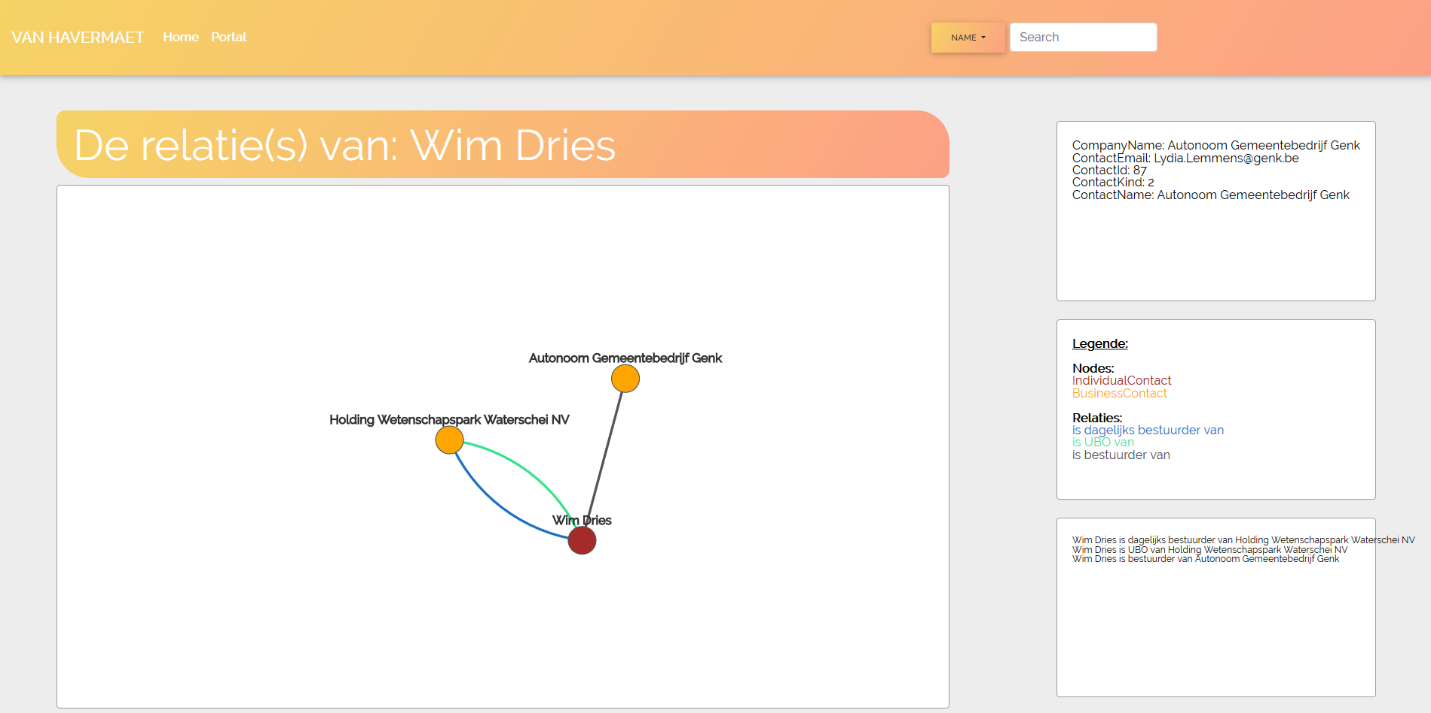
Het kan zijn dat je veel meer opties in de lijst te zien krijgt dan het aantal die wij op bovenstaande afbeelding te zien krijgen. Wat je dan kan doen is oftewel verder de naam invoeren in de zoekbalk (dit gaat ervoor zorgen dat de zoekopdracht specifieker word en daardoor krijg je minder resultaten te zien) of je kan door te scrollen in de lijst naar de juiste persoon gaan.

## Visualisatie

Eens je dan geklikt hebt op de persoon waarvan je zijn relaties wilt weergeven, krijg je een soortgelijke pagina te zien zoals op onderstaande afbeelding.

In het grote linkse vierkant zie je een schematische voorstelling van de relaties van de persoon die je net hebt opgezocht. Het is mogelijk om van elke node (= bolletje op schema) meer informatie te krijgen. Dit doe je door met je muis/cursor op een van de nodes te gaan staan/te hoveren. In de bovenste rechtse vierkant ga je dan meer informatie over deze node te zien krijgen (zie rood vierkant op onderstaande afbeelding).

## Legende

In het middelste vierkant (bruin) in de rechter kolom vind je een kleur legende waar in je het type van elke node en het type van elke relatie kan zien.

In het nodes deel (blauw) vind je al de kleuren en types van contacten die de nodes bevatten. In het relaties deel (groen) vind je al de kleuren en relatietypes die de relaties bevatten.

In het onderste vierkant (geel) in de rechtse kolom vind je de richting van de relaties. Hier zie je wie van wat precies is. VB: “Wim Dries” heeft 1 relatie met “Autonoom Gemeentebedrijf Genk”. De title van deze grijze relatie is: “is bestuurder van”. In het laatste vierkant kan je dan lezen dat deze relatie in volgende richting is opgebouwd: “Wim Dries is bestuurder van Autonoom Gemeentebedrijf Genk”.

## Navigeren tussen nodes

Het is mogelijk om te wisselen van visualisatie naar visualisatie op 2 manieren: zoeken via de zoekbalk (. zie hierboven in deel ‘Zoekfunctie’) of klikken op nodes. Deze tweede manier wijst zijn eigen uit. Je kan klikken op een node wat er dan voor zorgt dat je van die ene persoon al zijn relaties te zien zal krijgen. Als je door te klikken op een node terecht komt die geen relaties heeft, krijg je als resultaat gewoon de vorige visualisatie te zien.

# Installatiehandleiding

# Getting started

Hieronder vind je het volledige stappenplan om onze tool werkend en functioneel te krijgen. We hebben ook beschreven hoe je dit in de toekomst zou kunnen uitbreiden.

## Installatie

### Repository

Eerst en vooral beginnen we met het clonen van deze repo. Dit kan je doen via programma's zoals 'Github desktop' of 'GitKraken'.

Een andere manier om dit te doen is via command line (Windows PowerShell, Git Bash, CMD, ...). Je navigeert naar de locatie waar je de repo zou willen clonen adhv het commando 'cd'. Eens je in de juiste directory zit dan je volgend commando uitvoeren om je repo effectief te clonen.

git clone <https://github.com/stijnsaenen/Afstudeerproject.git>

### Toegang tot databank

Voor dit deel van het stappenplan hoef je direct niet iets specifieks te doen. In onze tool maken we gebruik van toegang tot de databank server van 'Van Havermaet'. Dit betekend dat je zonder problemen toegang tot deze databank als je onze tool gebruikt binnen in het netwerk van 'Van Havermaet'. Als je dus onze tool wilt gebruiken en je bent in het netwerk van 'Van Havermaet', mag je deze stap overslaan.

Als je van buiten het netwerk van 'Van Havermaet' toegang wilt, zal je een firewall exception moeten aanmaken om toegang te verkrijgen vanuit het netwerk waar je je dan in bevindt. In deze tutorial gebruiken we hiervoor [Microsoft SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15).

Eens je dit programma gedownload en geïnstalleerd heb, moet je connectie maken met de databank adhv onderstaande credentials:

- Server type: Database Engine

- Server name: vhuat.database.windows.net

- Authentication: SQL Server Authentication

- Login: azureDB\_admin

- Password: P@55w0rd

Hierna verschijnt er een pop-up venster waarin je de bepaalde firewall exception kan aanmaken adhv je IP-adres of een subnet. Na deze stap moet je je aanmelden op de azure omgeving. Dit doe je met deze credentials:

- E-mail: [admin@vanhavermaetuat.onmicrosoft.com](mailto:admin@vanhavermaetuat.onmicrosoft.com)

- Password: re@dyf0r@cti0n!

Als dit gelukt is, heb je toegang verkregen tot de databank en kan je gebruik maken van onze tool.

### ETL tool (Extraction - Transformation - Load)

### Dit deel is alleen toepasselijk als je de applicatie wilt uitbreiden! NIET nodig voor te applicatie te runnen

We hebben een ETL tool gebruikt om verschillende data sources samen te brengen en te combineren in 1 tussenlaag. Dit is in ons geval de ucllDB databank op de databank server van 'Van Havermaet'. Als er in de toekomst nog data sources zouden bijkomen die moeten toegevoegd worden aan de tussenlaag, kan dit gedaan worden via een ETL tool. Wij hebben hiervoor Pentaho Data Integration Spoon gebruikt.

In deze tutorial zullen we ook opnieuw Pentaho gebruiken. Voordat je dit programma opstart moet je ervoor zorgen dat je de juiste database driver hebt geinstalleerd. Deze vind je hier (<https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15>).

Eens je Pentaho hebt opgestart en de files open die in de repo op locatie 'Database\ETL - Pentaho' staan, zie je de transformations die wij gebruikt hebben om alles toe te voegen in de tussenlaag. In file 'vhuat db to tussenlaag(uclldb)' vind je de transformatie waar wij alle gegevens van uit de bestaande databank van 'Van Havermaet' transfereren naar onze tussenlaag. Dit werkt heel simpel: elke input node (rood) dat je ziet, staat voor een tabel en is op zijn beurt gelinkt met een output node (groen) die staat voor een tabel op de tussenlaag.

Als er nieuwe data in de tussenlaag moet komen, die komt van de bestaande databank van 'Van Havermaet' kan je gewoon een configuratie kopieren en de gegevens aanpassen (kiezen van welke, naar welke tabel er moet worden getransfereerd).

De tweede file 'csv-to-db' is een transformatie die we gebruikt hebben om csv data, die we via een script van de active directory omgeving hebben gehaald, om te zetten daar een tabel en deze ook weer in te voegen in de tussenlaag. Net zoals de vorige, kan je deze gewoon kopiëren en al de info van de nodes aanpassen om het om te zetten naar de csv/tabel die je nodig hebt.

### Installatie Package

Het is belangrijk dat je de correcte python versie gebruikt. Versie 3.7.x is een vereiste, bij voorkeur gebruik je versie 3.7.6.

Deze tutorial (https://www.youtube.com/watch?v=bXWlyOMYpRE) kan je helpen om python correct te installeren.

Bij versie selecteer je 3.7.6, volg de tutorial tot 7:45.

Je kan de installatie controleren met:

python --version

### Installatie packages

Vervolgens installeren we de benodigde packages. Om dit makkelijker te maken is er een requirements.txt file in je repo, waarin alle packages opleglijst staan.

Navigeer in cmd naar de /Flask map in je repo.

Je kan de packages installeren met het commando:

pip install -r requirements.txt

Het is belangrijk dat requirements.txt in dezlefde map staan als de map waarin je het commando uit voert.

### Serve de applicatie lokaal

Flask, het package die we gebruiken voor de backend, heeft een ingebouwde lokale webserver, je kan deze runnen in de /Flask map met het commando:

flask run

Indien dit niet werkt kan je simpelweg de app.py uitvoeren met het commando:

py app.py

of

python app.py

Tip: als je aanpassingen wil maken in de backend is het handig om Flask debug mode te activeren, dan herstart de server automatisch bij elke aanpassing:

SET FLASK\_DEBUG = 1

### Overzicht file structure

Naast app.py, die de backend bevat, toont de volgende boom de frontend files. De overige files, die hier niet getoond worden, bevatten voornamelijk libraries die in deze files worden gebruikt.

Flask

│

│

├───static

│ │

│ │

│ ├───js

│ ├───network.js

│ └───searchbar.js

│

│

└───templates

|───index.html

network.js is de kern van de frontend en bevat de code de het netwerdiagram construeert en alles er rond.

searchbar.js is de code met betrekking tot de zoekalk en de autocomplete.

Deze bestanden moeten verplicht in de /static map.

index.html is het hoofd html bestand, dit is het bestand dat allereerst wordt ingeladen als je de pagina bezoekt.

De bovenstaande js files zorgen voor de invulling ervan.

html files moeten verplicht in de templates map.

### Docker

In dit deel beschrijven we hoe we via Docker ervoor zorgen dat we de applicatie kunnen installeren en runnen via localhost:5000. Voor Docker heb je een linux machine nodig, je kan dit ook doen via een virtuele linux machine op je Windows computer.

1. Navigeer in de repo naar de Flask folder

2. Volg [deze tutorial](https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-using-the-repository) om docker te installeren in Linux. Je moet enkel het deel "Install using the repository" volgen en uitvoeren.

3. Voer de volgende commando's uit

sudo usermod -aG docker $UsernameVanRootGebruiker

docker build -t flask:latest .

docker run -d -p 5000:5000 flask:latest

docker ps

4. Dan zou je iets zien zoals dit:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS

9701 flask python app.py 3 min ago Up 4 min 0.0.0.0:5000

Dit betekend dat de applicatie succesvol draait op de localhost:5000 (Als je hiernaar surft zou je de applicatie moeten zien).

# Scripts

In dit document ga ik een overzicht geven welke scripts we allemaal geschreven hebben, waar je deze kan vinden en van elke script een kleine uitleg geven voor wat deze dient en hoe we deze hebben gebruikt.

## ExportOffice365UsersAndGroups3.ps1

Dit script is een file die we gebruikt hebben om al de nuttige data van de Active Directory (AD) van ‘Van Havermaet’ te halen en weg te schrijven in .csv files. Op deze active directory stond per gebruiker in groepen deze zitten en de rollen die elke gebruiker heeft. Aan de hand van deze gegevens, kon men ook in de active directory gaan zoeken welke rollen en groepen, welke permissies hebben. Deze gegevens hadden we nodig om deze te implementeren in onze tussenlaag, om deze hierna dan ook te gaan visualiseren.

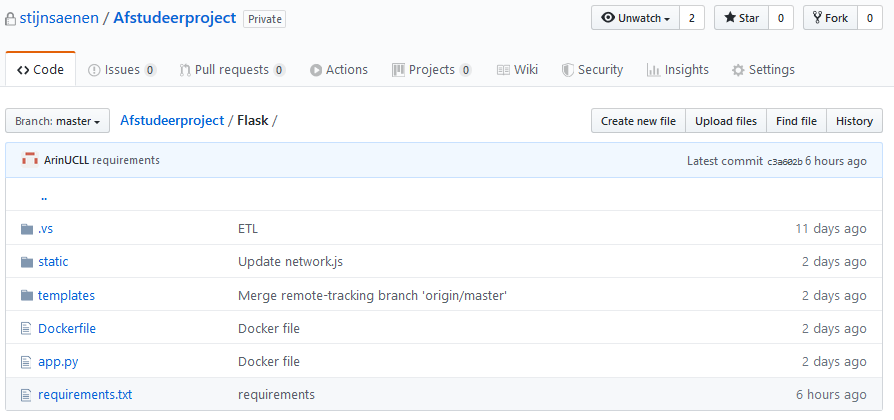
Dit script hebben we wel niet zelf geschreven, we hebben een script gebruikt dat we online hebben gevonden. (Zie bronvermelding) Het enige wat we moesten doen in het script om het te laten werken is de credentials aanpassen om in te loggen op de active directory en de namen van de tabellen aanpassen naar de namen van de tabellen die wij specifiek nodig hadden.

Meer uitleg is te vinden op in het script zelf in de vorm van comments.  
  
Bron:  
<https://confluence.atlassian.com/crowdkb/export-azure-active-directory-users-groups-and-import-into-crowd-952611012.html>

## Cvs-to-db.ktr & vhuat db to tussenlaag (ucllDB).ktr

Deze twee files zijn niet echt scripts. Ze zijn de bestanden die we gebruikt hebben om de .csv files van het eerste script om te zetten naar een tabel op de tussenlaag en de nuttige tabellen van de databank van ‘Van Havermaet’ te kopiëren naar de tussenlaag. Handleiding voor dit script is te vinden in de README op de repo (<https://github.com/stijnsaenen/Afstudeerproject>).

## App.py



Onze volledige back-end bestaand voornamelijk uit deze python file. Het maakt connectie met de databank (tussenlaag), doet query’s op de databank om specifieke data op te halen voor bijvoorbeeld: de visualisatie, de zoekbalk en de legende. Het transformeert de opgevraagde data ook om in de juiste json format zodat de library die we gebruiken voor de visualisatie, de json kan lezen en begrijpen. Het is de link tussen de tussenlaag databank en de front-end.

Meer uitleg is te vinden op in de file zelf in de vorm van comments.

## Searchbar.js

Deze JavaScript file zorgt voor alle dynamiek achter de zoekbalk. Het voert een methode uit, die in app.py staat, om alle personen op te halen die je kan visualiseren.

Als je een persoon ingeeft in de zoekbalk,geeft hij eerst een lijst die overeenkomst met u zoekopdracht. Als je op een specifieke persoon hebt geklikt, gaat hij de id van deze persoon ophalen en doorsturen naar de backend, zodat deze op zijn beurt de juiste visualisatie json terug kan geven naar de frontend. De file bevat ook een functie die ervoor zorgt dat er pas na 3 letters in het invoerveld resultaten worden getoond. We hebben dit op deze manier gecodeerd om ervoor te zorgen dat er geen hele lange lijst aan opties op de webpagina verschijnt, dit is niet gebruiksvriendelijk.

Meer uitleg is te vinden op in de file zelf in de vorm van comments

## Network.js

Deze JavaScript file heeft meerdele doelen. Eerst en vooral, en volgens ons ook de belangrijkste functie, tekent het de effectieve visualisatie van de relaties en de nodes. Hiervoor gebruiken we een library genaamd “D3.js”. Het herkent in de json de links en de nodes en gaat hier adhv id’s relaties tussen leggen. Het geeft ook elke type node of relatie een verschillende kleur.

De tweede functie dat dit bestand heeft is wanneer je met je muis over een node gaat, er in een ander vierkant informatie over deze node wordt getoont. Dit gebeurt doordat deze file een methode van app.py oproept. De derde functie is het tonen van een kleurlegende waarin je kan zien welke type nodes en type van relaties in de visualisatie vookomen. Ook toont het welke specifieke kleur deze hebben doordat de tekst in de legende de kleur van de juiste node of relatie bevat. Het laatste is het tonen van de richting van de relatie. Welke relaties er zijn wordt weergegeven in de kleurlegende maar de specifieke richting word aangeduid in het laatste vierkant.

De network.js file staat dus vooral in voor de visualisatie van het netwerk, de informatie en de legende.

Meer uitleg is te vinden op in de file zelf in de vorm van comments

Bron: <http://bl.ocks.org/thomasdobber/9b78824119136778052f64a967c070e0>

# Verslagen van meetings

## Donderdag 13/02: Meeting bij klant

|  |  |
| --- | --- |
| AANWEZIGEN: Stijn Saenen, Stef Michiels, Pieter-Jan Van Der Schueren, Arin Er, Benny Westaedt, Dimitri Joosten, Tom Eversdijk, Johan Strypsteen  VERONTSCHULDIGD: / VERSLAGGEVER: Arin Er DATUM VOLGENDE VERGADERING: 18/02/2020  AGENDA:   * Introductie Van Havermaet * Voorstelling Gantt chart * Algemene bespreking van het project |  |

*Dit document dient doorgestuurd te worden naar de klant en de begeleider van het afstudeerwerk na invullen*

KORT VERSLAG:

Voor dat het gesprek met Van Havermaet plaats vond hebben we zo veel mogelijk zaken onderzocht die te maken hadden met het project. Er waren echter onduidelijkheden, die we verduidelijk moesten worden. Hierdoor kwam het gesprek op het perfecte moment.

Na de algemene introductie kon elke groep hun project bespreken en de Gantt chart voorstellen. De Gantt chart van onze groep was de dag op voorhand afgewerkt om deze zo up-to-date mogelijk te houden. Na de voorstelling van het project en de Gantt chart volgde er feedback van Benny Westaedt en Dimitri Joosten.

De info die we als groep hebben onthouden:

* Er zijn 3 soorten klanten
  + Een persoon/individu
  + Een familie
  + Een Bedrijf
* De data op dit moment komt van de SQL DB, Azure AD en Sharepoint. Deze kan echter elk moment aangevuld worden met data van groep 2, waardoor alle 3 projecten met elkaar verbonden kunnen worden.
* ETL tooling kan gebruikt worden i.v.m. de tussenlaag. Data opvragen om daarna uniform te maken.
* Nuttig om de bron info te tonen bij de relaties.
* Niet alle relaties zijn even belangrijk. De belangrijkste relaties moeten eerder getoond worden dan de minder relevante.

De dataset met de relevante tabellen zijn toegelicht met de bijhorende uitleg. We zullen later nog een officieel document doorgestuurd krijgen van Dimitri met daarin de documentatie van de databank. Veel van de bovenstaande zaken worden vertaald in acties die we moeten ondernemen in ons project.

ACTIES:

Tegen 18/02/2020 zijn de functionele/technische analyses in orde. Waarin we de probleemstelling nogmaals definiëren samen met onze oplossing. In deze documenten verdelen we ook de rollen in onze groep (project lead, scrum master, …).

Het documenteren van alles dat niet werkt zoals gewenst.

De bestaande planning moet ook aangepast worden op verschillende manieren:

* Milestones toevoegen
  + Deadlines die we onderling afspreken
  + Alle soorten meetings die plaatsvinden
  + Active directory mee in de planning steken, dit ontbrak in de huidige
* Tijdens heel het project kritisch zijn op de bestaande DB
* Onderzoek starten i.v.m. ETL zodat we een passende tool kunnen vinden

Story:

Story van de tussenlaag en netwerk zijn gelinked aan deze meeting.

## Donderdag 20/02: Meeting via Microsoft Teams

|  |  |
| --- | --- |
| AANWEZIGEN: Stijn Saenen, Stef Michiels, Pieter-Jan Van Der Schueren, Arin Er, Benny Westaedt, Johan Strypsteen  VERONTSCHULDIGD: Dimitri Joosten VERSLAGGEVER: Arin Er DATUM VOLGENDE VERGADERING: 27/02/2020  AGENDA:   * Bespreking projectplan (Gantt chart) * Algemene bespreking (vragen & benodigdheden) * Demo van project tot nu toe |  |

*Dit document dient doorgestuurd te worden naar de klant en de begeleider van het afstudeerwerk na invullen*

KORT VERSLAG:

De wekelijkse meeting van 20/02/2020 diende als algemene check up van het project. Sinds de vorige meeting ging het project in het algemeen veel vlotter. Dit kwam doordat de doelstellingen veel duidelijker waren. Het verdelen van de taken ging hierdoor efficiënter.

De meeting ging door met Benny omdat Dimitri niet aanwezig kon zijn. Dit zal de komende weken ook het geval zijn.

Er waren twee algemene vragen die op voorhand klaarstonden voor Benny:

* Er was te weinig data in de Databank van Van Havermaet om een mooie visualisering te tonen in de demo applicatie.

Benny had twee oplossingen voor dit idee. De eerste oplossing voor dit probleem was het werken op de productie databank. De tweede optie was werken op een kopie van de productie databank die op de testomgeving gezet ging worden. Om fouten te vermijden hebben we voor optie twee gekozen. Deze kopie gaat volgens Benny beschikbaar zijn op 21/02/2020.

* Vraag of er suggesties of vereisten waren, voor de UI van de pagina, voor de visualisatie.

Momenteel verschijnt er een kader met meer info over de node waarover de muis aan het hoveren is. Deze info moet nog uitgebreider zijn. Om een onoverzichtelijke kader te vermijden vroeg Benny achter een algemene kader die verschijnt naast het venster met de visualisatie. Deze zal updaten wanneer de muis hovert over een nieuwe node.

De data die getoond moet worden in de kader moet bestaat uit:

* + Nummer
  + Naam en voornaam
  + Rijksregisternummer
  + Type klant (business of private), komt deels uit ContactCaps (=JSON)

ACTIES:

* Nieuwe testdata gebruiken eens deze beschikbaar is op 21/02/2020
* Project plan (Gantt chart) updaten
  + Kijken wat er al in orde is
  + Kijken wat sneller/trager loopt dan verwacht
* Installatie documentatie (visuele en ETL)

Story:

Story van de nodeinfo is gelinked aan deze meeting.

## Donderdag 27/02: Meeting bij klant

|  |  |
| --- | --- |
| AANWEZIGEN: Stijn Saenen, Stef Michiels, Pieter-Jan Van Der Schueren, Arin Er, Benny Westaedt, Johan Strypsteen , Tom Eversdijk VERONTSCHULDIGD:  VERSLAGGEVER: Arin Er  AGENDA:   * Demo van project tot nu toe * Feedback |  |

*Dit document dient doorgestuurd te worden naar de klant en de begeleider van het afstudeerwerk na invullen*

KORT VERSLAG:

De wekelijkse meeting van 27/02/2020 diende, net zoals vorige week, als algemene check up van het project. Deze week was het meer gericht op de demo.

De week van 24/02 tot 28/02 was er een goede gestructureerde planning. Er waren echter veel meer drempels als verwacht, doordat we zagen hoeveel er bij kwam kijken om heel de flow te vervolledigen.

De demo was daarom niet zoals we hadden ingepland aan het begin van de week. De front-end was in orde en de link naar de back-end ook. Het converteren van de JSON naar D3 formaat was niet 100% in orde. De visualisering was daardoor hardcoded. Op het moment dat dit verslag is geschreven is het probleem al opgelost.

De week van 2/03 mogen er niet heel veel set backs meer zijn. Anders zouden er problemen ontstaan i.v.m. de deadline van het project. Vervolgens hebben we de feedback van Benny omgezet naar acties die we moeten ondernemen in de week van 2/03.

ACTIES:

De acties zijn geordend op prioriteit

1. Dynamische search van relaties
2. Tussenlaag 100% operationeel maken, AD informatie hoort hier bij
3. Individuele nodes clickable maken zodat de relaties van deze nodes ook zichtbaar worden
4. De nodes en links identificeren aan de hand van kleuren
5. Legende maken voor het zichtbare netwerk

Story:

Story van de opties, ContactList en legende zijn gelinked aan deze meeting.

## Maandag 02/03: Meeting via Microsoft Teams

|  |  |
| --- | --- |
| AANWEZIGEN: Stijn Saenen, Stef Michiels, Pieter-Jan Van Der Schueren, Arin Er, Benny Westaedt, Johan Strypsteen , Tom Eversdijk VERONTSCHULDIGD:  VERSLAGGEVER: Arin Er  AGENDA:   * Demo van project tot nu toe * Feedback |  |

*Dit document dient doorgestuurd te worden naar de klant en de begeleider van het afstudeerwerk na invullen*

KORT VERSLAG:

De wekelijkse meeting van 27/02/2020 diende, net zoals vorige week, als algemene check up van het project. Deze week was het meer gericht op de demo.

De week van 24/02 tot 28/02 was er een goede gestructureerde planning. Er waren echter veel meer drempels als verwacht, doordat we zagen hoeveel er bij kwam kijken om heel de flow te vervolledigen.

De demo was daarom niet zoals we hadden ingepland aan het begin van de week. De front-end was in orde en de link naar de back-end ook. Het converteren van de JSON naar D3 formaat was niet 100% in orde. De visualisering was daardoor hardcoded. Op het moment dat dit verslag is geschreven is het probleem al opgelost.

De week van 2/03 mogen er niet heel veel set backs meer zijn. Anders zouden er problemen ontstaan i.v.m. de deadline van het project. Vervolgens hebben we de feedback van Benny omgezet naar acties die we moeten ondernemen in de week van 2/03.

ACTIES:

De acties zijn geordend op prioriteit

1. Dynamische search van relaties
2. Tussenlaag 100% operationeel maken, AD informatie hoort hier bij
3. Individuele nodes clickable maken zodat de relaties van deze nodes ook zichtbaar worden
4. De nodes en links identificeren aan de hand van kleuren
5. Legende maken voor het zichtbare netwerk

Story:

Story van de relatieinfo is gelinked aan deze meeting.

# Spike: Flask – Django

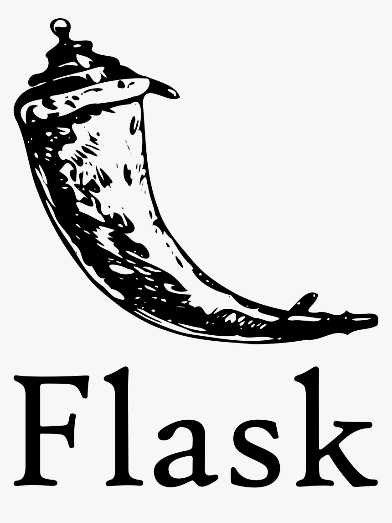
## Inleiding

We wisten vrij vroeg dat we python wilden gebruiken voor onze backend. Python biedt veel packages die relevant zouden kunnen zijn voor ons project (pyodbc, pandas, sqlalchemy, flask, django …), en het is een taal waarmee ja snel van start kunt gaan, ideaal voor een kort project als deze die geen monolitische backend nodig heeft.

Wat we echter nog niet wisten is welk web development framework we zouden gebruiken. Nadat we de limitaties van python zonder één van deze frameworks begonnen waar te nemen , zijn we beginnen zoeken naar een oplossing, de opties: Flask en Django.

Om er achter te komen welke technologie het meest geschikt is voor onze toepassing moeten we 2 vragen stellen: Wat moet mijn applicatie kunnen? En wat verwacht ik van het framework?

Django is een high level, open source web application framework. Het is geschreven in python en wordt gebruikt door oa. PBS, Mozilla, Instagram en The Washington Post. Het maakt gebruik van verschillende componenten voor zijn breed aanbod aan features.

Flask is een microframework, geschreven in python, die je niet verplicht bepaalde tools of libraries te gebruiken, hiervoor kan je beroep doen op extensies. Het wordt gebruikt door oa. LinkedIn en Pinterest.

## Vergelijking van de mogelijkheden

## ORM

Object Relational Mapping is het mappen van het model met het database schema.

Django heeft ingebouwde ORM, die niet ideaal is voor geavanceerde queries. Django ondersteunt ook models migration.

Flask wordt niet geleverd met een ORM, je kan hiervoor zelf een tool uitkiezen, zoals SQLAlchemy of PonyORM.

Onze case: ORM is van belang.

## Templates engine

Templates engine is een tool die html documenten dynamisch genereert.

Flask gebruikt hiervoor Jinja2, een template engine die geïnspireerd is op de Django Template Engine en het uitbreidt met bijkomende features.

Django heeft zijn eigen templates engine, maar laat toe om externe template engines te gebruiken.

Onze case: Templates zullen in beperkte mate gebruikt worden.

## Forms

Forms dienen om data input op te halen van een user.

Django beidt complexe tools voor forms.

Flask beidt geen tools voor forms, externe libraries zijn nodig (Flask WTF).

Onze case: geen forms nodig.

## Admin panel

Tool die toe laat database content te managen, nuttig indien app beheerd moet worden door niet-technische individuen.

Django heeft een ingebouwde admin panel met authenticatie, permission levels en data validation.

Flask biedt geen admin panel, kan library gebruiken (Flask-Admin).

Onze case: geen admin panel nodig.

## Authenticatie

Django beidt bescherming tegen XSS, CSRF, SQL injection, Clickjacking.

Flask beidt bescherming tegen XSS. Je moet zelf aan beveiliging denken bij het gebruik van tools.

Onze case: security is belangrijk, maar er zijn weinig risico’s verbonden aan ons project, andere zaken helpen ook bij het verbeteren van de veiligheid van ons project (abstractie databank, SQLAlchemy).

## Performantie

Flask is sneller omdat het minder uitgebreid is. Echter zijn de verschillen niet al te groot in een productie omgeving.

## Doel

Django laat toe om, zonder al te veel kennis over hoe het werkt, volledige applicaties te maken.

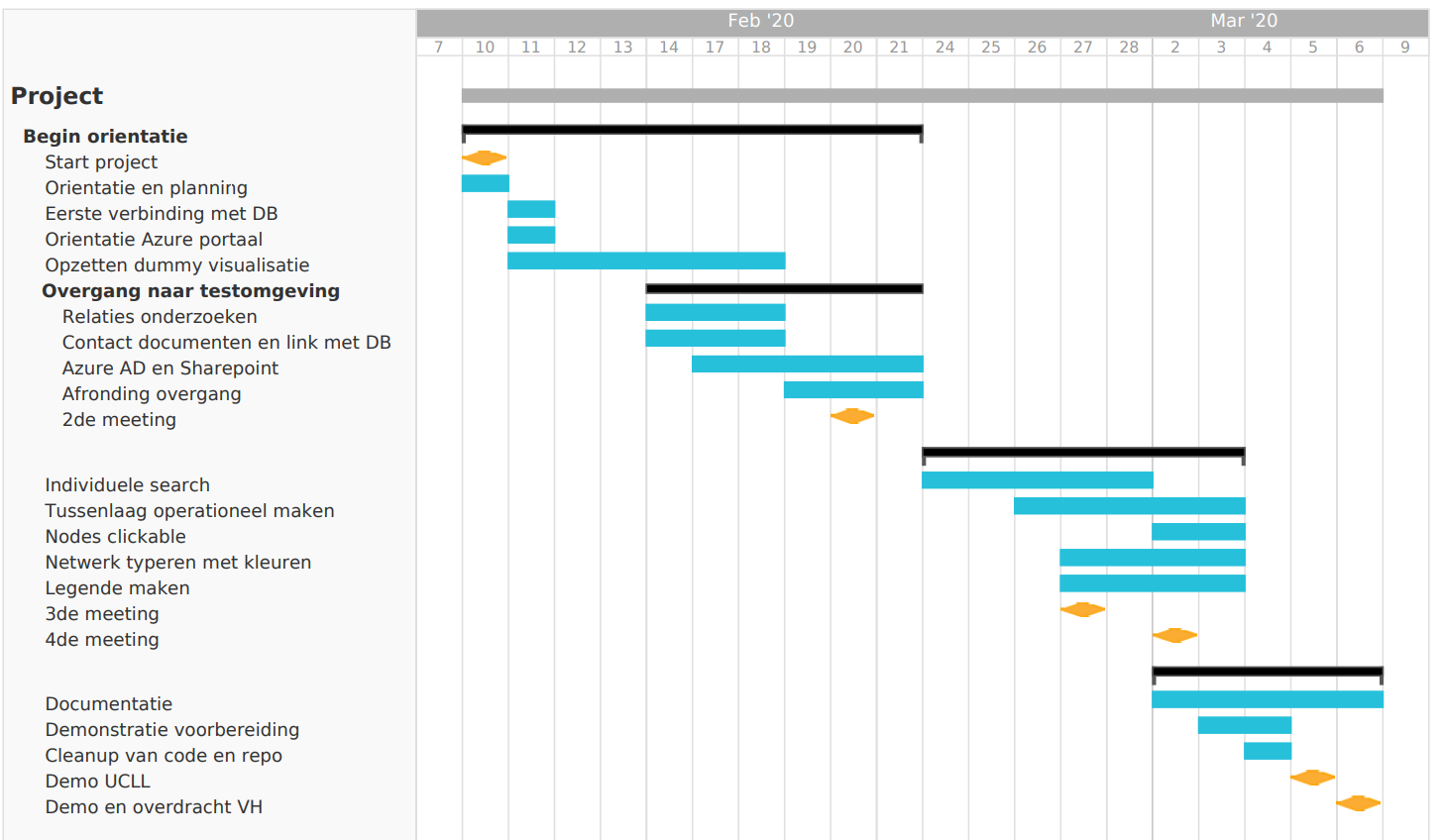
Flask is makkelijker te begrijpen en is perfect om minimalistische applicaties te maken.

Onze case: Minimalistische applicatie

## Conclusie

Aangezien ons project een beperkte scope heeft besluiten we dat Flask meer geschikt zal zijn. Veel features die Django beidt zijn niet nodig (forms, admin panel). Flask zal ook flexibeler zijn en makkelijker om op een beperkte tijdspanne ons project te laten draaien. De betere performantie van Flask kan onze web applicatie mogelijks sneller maken. Het nadeel is dat we voor ORM zelf een oplossing zullen moeten vinden (we kijken naar SQLAlchemy).

# Gannt-chart



# Bronnen

<https://d3js.org/>

<https://stackoverflow.com/>

<https://www.w3schools.com/>

…

En vele anderen die op verscheidene plaatsen in dit document al vermeld staan.  
Bv. Scripts uitleg, Gebruikershandleiding, …  
Er is in het algemeen ook gewoon veel gebruik gemaakt van google en youtube en daarom is het vaak moeilijk om alle specifieke bronnen hier neer te lijsten.

# Slot

Deze afgelopen 4 weken hebben we veel geleerd over het uitvoeren van een project voor een echte klant. We hebben gemerkt dat niet altijd alles zo vlot loopt als verwacht. Bij elke implementatie van een nieuwe stap in het plan kunnen er altijd complicaties ontstaan waardoor de planning achteruit loopt. Ook is het belangrijk om prioriteiten te stellen en meteen aan het werk te gaan met de meest belangrijke componenten. Zo zijn er een aantal onderdelen die we onderzocht hebben maar uiteindelijk, wegens tijdsgebrek, niet hebben kunnen implementeren. Ook is het belangrijk om zo veel mogelijk contact te houden met de klant, om te zorgen dat iedereen weet wat er verwacht wordt en zodat alles correct begrepen is. Tevens is het ook belangrijk om na te kijken dat alle gegeven data ook daadwerkelijk overeenkomt met wat er verwacht was. Zo was de Active Directory omgeving die we hadden gekregen niet overeenkomstig met de data van de database en konden we dat niet meer implementeren wegens tijdsgebrek.  
  
Al bij al denk ik dat we toch tevreden mogen zijn van wat we hebben afgeleverd, aangezien onze opdrachtgever ons toch een vrij positieve reactie had gegeven tijdens de demonstratie en overhandiging van het eindproduct.