Professionele Bachelor Toegepaste Informatica

Cubigo Community Platform: Partner Integration

Michiel Pawlenko

Promotoren:

Jelle Raus & Jeff Vaes — Cubigo NV
Wesley Hendrikx — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Cubigo Community Platform: Partner Integration

Michiel Pawlenko

Promotoren:

Jelle Raus & Jeff Vaes — Cubigo NV
Wesley Hendrikx — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Ik wil vooral Cubigo bedanken, voor mij een kans te geven aan de stageopdracht te mogen werken, en mijn bedrijfspromotors, Jelle Raus en Jeff Vaes, om mij zo goed te helpen en op te volgen tijdens de stageopdracht en alle (moeilijke) onderdelen van de stageopdracht zo duidelijk uit te leggen.

Verder wil ik alle collega's van Cubigo bedanken voor mij altijd te hulp te staan als ik hulp nodig had, maar ook voor de goede sfeer die er op Cubigo altijd hangt.

Als laatste wil ik mijn hogeschoolpromotor, Wesley Hendrikx, nog bedanken voor de feedback die hij heeft gegeven, die me verder heeft kunnen helpen met mijn eindwerk.

Abstract

Cubigo NV is een kleinschalig bedrijf dat momenteel één product aanbiedt: het Cubigo *Community* Platform. Dit platform is een manier om het leven van oudere mensen die in gemeenschappen wonen, waar ze verzorgd worden en allerlei services aangeboden krijgen, te vergemakkelijken. Via deze applicatie kunnen de bewoners voeding bestellen, transport aanvragen, zich inschrijven voor georganiseerde activiteiten, etc.

Momenteel is Cubigo ermee bezig om services van een extern EHR-bedrijf, Point Click Care (PCC), deels in het Cubigo *Community* Platform te integreren. Cubigo wil in zijn platform patiëntgegevens ophalen uit PCC, om daaruit gebruikers in het *Community* Platform aan te maken. Ook zou er een '*master-slave'*-synchronisatie tussen de twee platformen moeten zijn: als er patiëntgegevens veranderd, toegevoegd of verwijderd worden in PCC, moeten deze veranderingen ook op de corresponderende gebruiker(s) toegepast worden in het *Community* Platform.

Het Cubigo *Community* Platform is opgebouwd uit een Angular-frontend, met een ASP.NET backend die gebruikt wordt voor microservices en een API-backend waarmee de frontend zijn data kan ophalen. De integratie met PCC wordt dus opgebouwd uit meerdere microservices, die dan in de backend terechtkomen.

Voor het onderzoeks-onderdeel van de stage wordt er onderzoek gedaan naar Business Intelligence. Cubigo bekijkt momenteel of Business Intelligence (BI) gebruikt kan worden om onder andere het welzijn van de gebruikers van hun platform op te meten. Dit welzijn is namelijk zeer belangrijk voor de gemeenschappen waarin het platform gebruikt wordt, omdat meer welzijn betekent dat de inwoners van deze gemeenschappen er langer blijven wonen, waardoor de gemeenschap, en dus ook Cubigo, er meer winst uithaalt.

Het onderzoek zal een vergelijking worden tussen drie zeer populaire BI-toolsets: Microsoft Power BI, Qlik Sense en Tableau. Er wordt onderzocht hoe de gegevens van de backend in de BI-tool gebracht kunnen worden, welke stappen er ondernomen moeten worden om de gegevens te visualiseren in tabellen, grafieken, etc. Dit wordt allemaal gedaan aan de hand van een Postgres-database, zodat er een gelijkaardige infrastructuur is aan die van het Cubigo *Community* Platform.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

Application Programming Interface	API
Business Intelligence	BI
Cubigo Community Platform	CCP
Electronic Health Record	EHR
NServiceBus	NSB
Point Click Care	PCC
Product Owner	PO
Proof of Concept	POC
Software-as-a-Service	SaaS
Vice President	VP
Virtual Machine	VM

Inleiding

In het huidige tijdperk, waar bijna iedereen een online aanwezigheid heeft, is er een hele resem aan gebruikersgegevens beschikbaar op de verschillende online platformen die dagelijks door miljoenen mensen gebruikt worden. Uit deze gegevens kunnen bedrijven verschillende dingen leren, vaak met als doel het product dat ze aanbieden te verbeteren, en omzet te verhogen.

Natuurlijk moet deze data verwerkt worden voordat ze gebruikt kan worden. Het verwerken en gebruiken van deze data om omzet te verhogen, wordt ook wel Business Intelligence (BI) genoemd.

Cubigo is momenteel op zoek naar de beste tool die ze kunnen gebruiken voor hun data te verwerken en verduidelijken, om zo te kijken hoe ze, op basis van deze gegevens de tevredenheid van hun gebruikers te verhogen.

Er zijn een groot aantal tools op de markt die gebruikt kunnen worden voor BI. In dit eindwerk zullen er drie prominente BI-tools vergeleken worden, op basis van bepaalde criteria, om zo te bepalen welke BI-tool het makkelijkst te gebruiken is en het meest voldoet aan de criteria van Cubigo.

Eens dat de vergelijking gemaakt is, wordt er in de conclusie per vergeleken onderdeel gekeken welke van de drie tools het beste scoort op dat onderdeel van de vergelijking.

Dankzij dit onderzoek wordt niet alleen duidelijk welke BI-tool het gemakkelijkste te gebruiken is, maar ook wat de BI-tools allemaal kunnen en wat de verschillende prijsklassen zijn van de BI-tools.

Dit onderzoek vormt het tweede deel van het eindwerk. Het eerste deel van het eindwerk is een verslag over de stageopdracht die ik heb uitgewerkt voor Cubigo. In dit verslag is te lezen welke technologieën er allemaal gebruikt zijn in de opdracht, een voorstelling van Cubigo zelf, en als belangrijkste, een beschrijving van de verschillende stappen van het ontwikkelproces van de stageopdracht.

1. Stageverslag

1.1 Bedrijfsvoorstelling

Cubigo is een scale-up IT-bedrijf, opgericht in 2011, met 30 werknemers in dienst. Het biedt een online platform, het Cubigo *Community* Platform, aan, dat gebruikt wordt binnen gesloten gemeenschappen van ouderen. Deze gemeenschappen bestaan uit residenties, waar de ouderen een appartement huren. Hier worden ze dan verzorgd door personeel dat in de residenties werkt, met als de doel de inwoners nog zo zelfstandig mogelijk te laten leven.

Het CCP biedt de inwoners de mogelijkheid om gebruik te maken van enkele comfortservices die in de residentie worden aangeboden, zoals maaltijden bestellen, transport regelen, deelnemen aan geplande activiteiten, etc. Deze applicatie sluit dan ook aan bij het doel van Cubigo: via dit platform deze gemeenschappen helpen en de levenscomfort van de inwoners te verbeteren.

Momenteel is Cubigo vooral actief in de Verenigde Staten, omdat daar de markt van deze residenties veel groter is dan de rest van de wereld. Ze hebben twee werknemers in dienst die in de Verenigde Staten deze residenties bezoeken en onderzoeken of deze residenties het CCP kunnen gebruiken. Ook hebben ze een kantoor in Boston, waar deze twee werknemers actief zijn.

Sinds vorig jaar is Cubigo ook actief in België. Hier zijn ze het CCP in enkele kleinere residenties aan het uitrollen. Dit geeft ook als extra voordeel dat het veel makkelijker is voor Cubigo om te polsen hoe de inwoners van deze residenties het CCP ervaren.

Cubigo maakt momenteel geen winst met het CCP. Dit komt omdat het geldvermogen van Cubigo uit een groep van investeerders komt. Daarom kan Cubigo het CCP aanbieden voor een relatief lage prijs. Naar de toekomst toe, als Cubigo het CCP al bij een grotere aantal residenties heeft uitgerold en er naambekendheid is, gaat er wel geprobeerd worden om winst te maken.

Het *Community* Platform is volledig ontwikkeld door Cubigo, en dit is dan ook waar ze zich in specialiseren: *full-stack* applicaties bouwen. Dit platform is een relatief uniek product, en momenteel zijn er nog geen echte concurrenten of gelijkaardige platforms die op het niveau zitten van het *Community* platform. Momenteel onderhoudt Cubigo enkel het *Community* platform, en ontwikkelen ze geen andere producten meer. Ze breiden enkel het *Community* platform nog uit en passen het aan naar de wens van de klant. Vroeger werden er wel nog andere projecten gemaakt, maar dit is sinds enkele jaren al niet meer.

Cubigo profileert zich vooral als een Belgisch bedrijf dat het levenscomfort van de inwoners van deze residenties wil verbeteren, doormiddel van het product dat ze aanbieden. Hoewel ze vooral in de Verenigde Staten actief zijn, leggen ze toch wel altijd de nadruk dat het een Belgisch bedrijf is, en dat de software in België gemaakt is.

1.2 Uitwerking Stageopdracht

De uitwerking van de stageopdracht gebeurde in twee grote onderdelen: een analysefase, en de uitwerkingsfase. Tijdens de uitwerkingsfase vonden er meerdere meetings plaats, waarbij de bedrijfspromotor, de Product Owner en de Product Vicepresident polsten hoe ver de stageopdracht stond, wat er nog allemaal moest gebeuren en wat er best als volgende onderdeel uitgewerkt moest worden.

1.2.1 Probleemstelling

Cubigo wil een integratie van een externe partner, PCC, in hun *Community* Platform. PCC biedt medische data aan over patiënten in de residenties waarin het CCP actief is. Cubigo zou deze patiëntengegevens willen importeren naar het *Community* Platform, om in het CCP gebruikers te genereren op basis van deze gegevens, en dus eigenlijk de patiënten uit PCC te 'importeren' naar het CCP.

Nadat de patiënten succesvol geïmporteerd zijn uit PCC, moeten de gegevens binnen het CCP up-to-date gehouden worden, wat mogelijk is via een notificatiesysteem van PCC.

Deze integratie kan natuurlijk niet zomaar gebeuren: er moet rekening gehouden worden met technicaliteiten, zoals verschillen tussen het PCC-ecosysteem en het CCP-ecosysteem, en de eisen die PCC stelt aan applicaties die hun patiëntgegevens gebruiken. Ook moet er rekening gehouden worden met het *GDPR* en *HIPAA* (Amerikaanse versie van het *GDPR*) wanneer er medische gegevens gebruikt worden.

Er zijn al integraties met externe services gebeurd in het CCP, maar nog niet op het niveau zoals in deze opdracht verwacht wordt.

Er zijn op voorhand al enkele use cases uitgewerkt voor de integratie. Een *Application Programming Interface* (API) met testdata is ook al beschikbaar gemaakt door PCC, samen met een POC die Cubigo gemaakt heeft om met deze API te verbinden.

Als de integratie voltooid is, wordt er een technische validatie gedaan door het ontwikkelingsteam van PCC. Als deze validatie succesvol voltooid wordt, komt Cubigo en het CCP terecht op de '*PCC-marketplace*'. Hierdoor kan Cubigo ook door andere residenties die PCC gebruiken gezien worden, en zo mogelijk ook in deze residenties binnen geraken.

Deze integratie is ook een voorbereiding op enkele toekomstige use cases. In de toekomst worden er niet alleen patiëntgegevens geïmporteerd uit PCC, maar ook specifiekere informatie, zoals allergieën en mobiliteitsgegevens. Deze kunnen dan gebruikt worden wanneer bijvoorbeeld een maaltijd besteld wordt, of als er transport aangevraagd wordt.

1.2.2 Doelstelling

Het eindproduct van de stage is een integratie met PCC binnen het CCP zijn. Hieronder wordt verstaan dat alle patiëntengegevens uit de PCC-API geïmporteerd worden naar het CCP. Binnen het CCP wordt dan voor elke patiënt een account aangemaakt. De gegevens van dit account moeten overeenstemmen met de patiëntengegevens uit PCC, en moeten ook up-to-date .

Eerst moet er een verdere analyse gebeuren rond de huidige use cases. Dit komt dan vooral neer op *user stories* maken onder deze verschillende use cases, en kijken welke technicaliteiten er komen kijken bij deze use cases, bijvoorbeeld welke aanpassingen er gemaakt moeten worden aan het *Community* Platform om PCC er in te integreren.

De nieuwe code wordt geschreven volgends de richtlijnen van Cubigo. Dit komt erop neer dat alles als een *microservice* geschreven wordt en dat er gebruikt gemaakt wordt van *message-queues*, *actors* en de nodige services die hiervoor nodig zijn (Microsoft Orleans bijvoorbeeld). Dit alles wordt geschreven in C#, binnen het ASP.NET-*framework*.

1.2.3 Gebruikte Technologieën

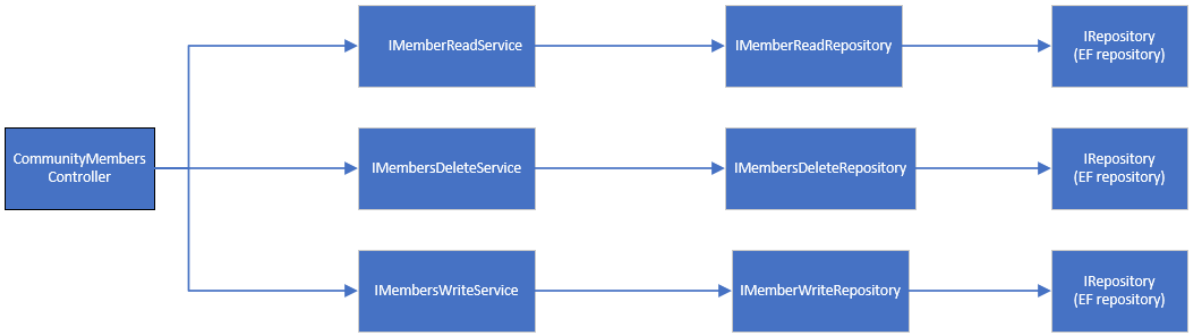
1.2.3.1 ASP.NET

De API-backend van het CCP is geschreven in C#, en maakt gebruik van het ASP.NET-*framework*. Deze backend maakt gebruik van het MVC-principe, samen met het principe van microservices om het systeem snel te houden en zorgen dat de onderdelen onafhankelijk van elkaar blijven.

1.2.3.2 Microservices

In zowel de frontend van het CCP als de API-backend ervan wordt gebruikt gemaakt van microservices. Het doel van deze microservices is zoveel mogelijk abstractielagen te creëren, zodat de code makkelijk getest of uitgebreid kan worden, maar ook om te zorgen dat de code mooi en leesbaar blijft.

Hieronder is een klein schema gemaakt, dat een voorbeeld van deze microservices structuur weergeeft.



Figuur 1 Microservices in de 'CommunityMembers'-controller.

De *CommunityMembers*-controller is verantwoordelijk voor gebruikersaccounts in de Cubigo-API aan te maken, up-to-date te houden en ze te verwijderen wanneer dit nodig is. Hierbij worden er verschillende services gemaakt om deze functionaliteit te ondersteunen. Deze services krijgen dan op hun beurt een *repository* service toegewezen. Deze *repositories* hebben dan zelf nog een *IRepository*-interface. Deze interface is verantwoordelijk voor de effectieve communicatie met de database, door middel van Entity Framework.

Door op deze manier van alles een verschillende service te maken, is elk onderdeel individueel testbaar, en staan al deze onderdelen los van elkaar.

Cubigo plaatst ook verschillende onderdelen van hun applicatie in verschillende projecten, zodat niet alle code in een groot, onoverzichtelijk project staat.

1.2.3.3 Dependency Injection

Al deze services gebruiken *Dependency Injection*. Hiervoor wordt Microsoft Unity gebruikt, een *library* dat *Dependency Injection* in het .NET Framework toelaat.

1.2.3.4 Postgres & Entity Framework

Postgres is de SQL-variant die in de API-backend van het CCP wordt gebruikt. De syntax die Postgres gebruikt, is grotendeels hetzelfde als gewone SQL.

Entity Framework wordt gebruikt om effectief transacties uit te voeren met de database, en voorziet dus eigenlijk een abstractielaag. Hiermee kan een gebruiker dus het gebruik van gewone *SQL-queries* vermijden en kan een gebruiker gewoon de datastructuren en *LINQ*-functies van het .NET Framework gebruiken.

Als er een veranderingen aan de database gebeuren, aan de hand van een 'migratie' binnen het Entity Framework, worden deze veranderingen niet zomaar toegepast. Er wordt een SQL-script gegenereerd van de migratie. Dit script wordt dan doorgestuurd naar de database-expert van het ontwikkelingsteam, die het script optimaliseert waar nodig, en dan de veranderingen toepast.

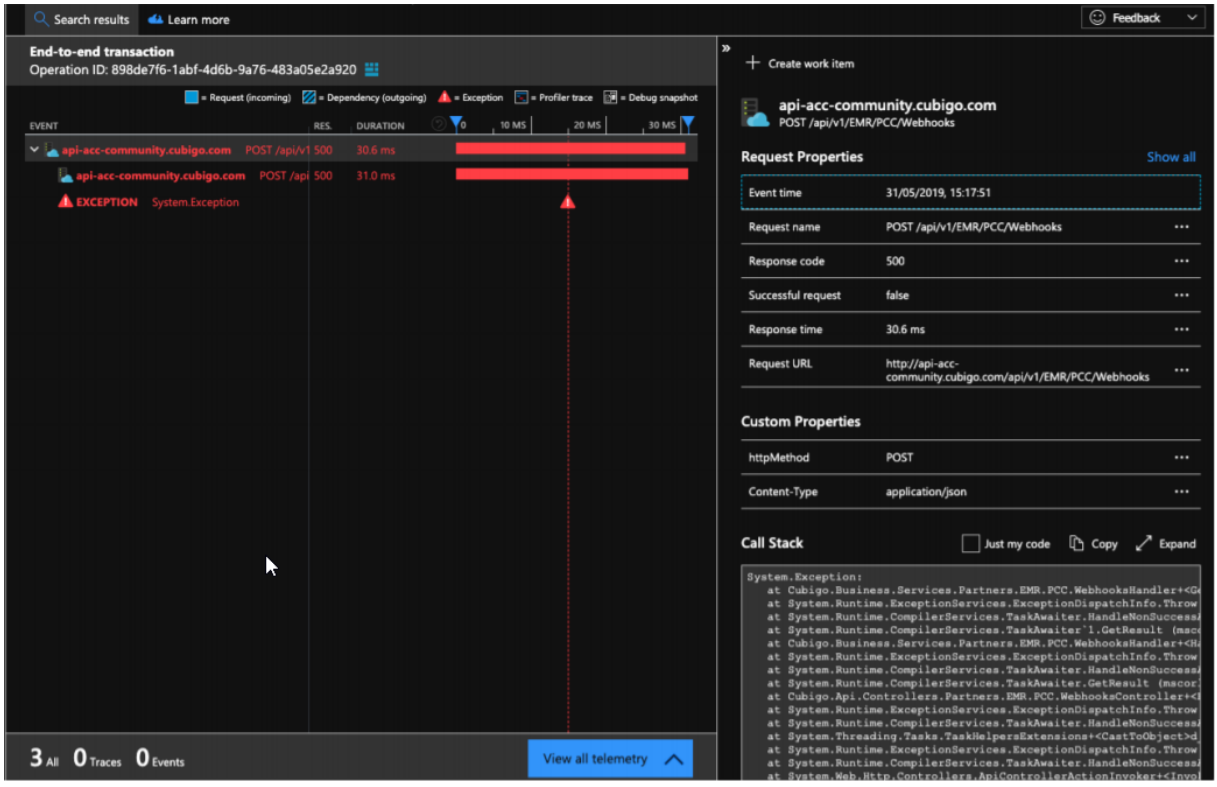
1.2.3.5 Microsoft Azure

Microsoft Azure wordt voor een aantal dingen gebruikt binnen Cubigo. Het belangrijkste waarvoor het gebruikt wordt is de *Application Gateway*, die ervoor zorgt dat alle binnenkomende aanvragen naar de juiste instantie van de Cubigo-API gestuurd worden. Ook wordt in de *Application Gateway* SSL-gerelateerde informatie bijgehouden.

Een andere heel belangrijke feature van Azure die gebruikt wordt binnen het CCP is *Application Insights*. Via *Application Insights* is het mogelijk om fouten die optreden in de backend van het CCP te loggen, en online op te slaan in Azure. Ook kunnen er grafieken gemaakt worden waarop een gebruiker kan zien hoeveel fouten er binnen een bepaalde tijdsperiode zijn opgetreden in het CCP.

Fouten loggen is niet alleen nuttig om gebruikers van het CCP beter te kunnen helpen, moesten er fouten optreden, maar ook tijdens het ontwikkelproces, omdat het niet altijd even makkelijk is om fouten op te sporen in een applicatie die zo goed als volledig asynchroon werkt.

Hieronder is een voorbeeld te zien van de een fout die gelogd is in *Application Insights:*



Figuur 2 Application Insights exception logging

De *message-queues* die gebruikt voor berichten te versturen tussen de verschillende onderdelen van het CCP, worden ook op Azure gehost.

Als laatste wordt ook nog het *DevOps*-onderdeel van Azure gebruikt voor het faciliteren van Agile. Ook de *repositories* waar de code van Cubigo inzit staan op Azure, zodat iedereen van het team er toegang tot heeft.

1.2.3.6 NServiceBus

NServiceBus (NSB) is een *messaging-framework* voor het .NET Framework. Met NSB kunnen er berichten op zogenaamde '*message-queues'* geplaatst worden, die dan ontvangen kunnen worden door andere onderdelen van de applicatie.

Voor het ontvangen van deze berichten wordt er een *'endpoint'* aangemaakt. Dit *endpoint* luistert op een *message-queue* voor berichten die een bepaalde type hebben (bijvoorbeeld welk soort klasse het bericht is), en handelt deze berichten dan af.

Binnen het CCP wordt NSB vooral gebruikt voor '*Fire & Forget'* taken. Dit zijn taken waarbij een bepaald onderdeel van het CCP een bericht stuurt naar een ander onderdeel van het CCP, dat dan een taak uitvoert, zonder dat de zender van het bericht zich zorgen moet maken of de taak succesvol voltooid is.

1.2.3.7 Microsoft Orleans

Microsoft Orleans is een *framework* voor .NET dat het zogenaamde *'actor-model'* gebruikt.

Het *actor*-model is een manier om een grote hoeveelheid kleine taken af te handelen. Een overkoepelend systeem ontvangt een aanvraag voor een *actor*-object aan te maken. Het systeem maakt dit object aan. Nu kan dit *actor*-object een aanvraag ontvangen en deze uitvoeren. Deze aanvragen uitvoeren en *actor*-objecten aanmaken gebeurt asynchroon. Hier ontstaat het grootste voordeel van het *actor*-model: er kunnen zoveel extra *actor*-objecten aangemaakt worden als het systeem aankan. Hierdoor kan er een heel groot aantal kleine taken, tegelijk en onafhankelijk van elkaar afgehandeld worden.

Elk *actor*-object houdt ook zijn eigen status bij, zodat er gekeken kan worden wat de *actor* aan het doen is. Zo kunnen actors ook opgeslagen worden en later terug hergebruikt worden, zonder dat er extra objecten aangemaakt moeten worden.

Omdat de *actors* los staan van elkaar, blijft het systeem ook gewoon doorwerken als er een fout optreed.

In Microsoft Orleans wordt dit systeem toegepast aan de hand van '*silo's*' en '*grains*'. Een *silo* is het overkoepelende systeem dat de verschillende *actor*-objecten beheert. De *grains* zijn de eigenlijke *actor*-objecten.

In de code kan er een *grain*-interface aangemaakt worden. Als deze *grain* wordt aangeroepen, gaat de silo een instantie van de *grain* aanmaken, om de taak uit te voeren waarvoor de *grain* opgeroepen wordt.

De silo zorgt ervoor dat er genoeg *grain*-instanties worden aangemaakt om de binnenkomende aanvragen uit te voeren.

Het resultaat is dat het hele systeem zeer schaalbaar is, samen met ingebouwde *error-handling* die ervoor zorgt dat het systeem nooit volledig crasht.

1.2.4 Analysefase

De analyse is van start gegaan met een meeting met de PO op het einde van de eerste week van de stage. Tijdens deze meeting heeft hij de verschillende use cases van de stageopdracht, die hij had uitgeschreven, uitgelegd. Er werd ook duidelijk gemaakt dat niet alle use cases afgewerkt moesten worden, en dat enkel verwacht werd dat de basisfunctionaliteit afgewerkt zou worden.

1.2.4.1 Use Cases verder onderverdelen

Nadat de use cases duidelijk waren, werden deze opgedeeld in *epics*, *user stories* en *features*. Hiermee heeft de PO geholpen, omdat hij voor een groot stuk verantwoordelijk is voor deze onderdelen uit te werken voor het team van ontwikkelaars. Deze verschillende onderdelen moesten allemaal aangemaakt worden in Azure *DevOps*, het platform dat Cubigo gebruikt voor alle Agile-gerelateerde onderdelen van het ontwikkelingsproces.

Wanneer een user story geschreven wordt, moet ook rekening gehouden worden met het “Given-When-Then” systeem, wat Cubigo gebruikt om criteria te maken waarmee gekeken kan worden of een user story naar toebehoren is uitgewerkt. Het “Given-When-Then” systeem bestaat uit 3 stappen:

- Given: wat een gebruiker wil dat er gebeurt;
- When: de actie die het proces in gang moeten zetten;
- Then: het resultaat van de actie die de gebruiker heeft uitgevoerd;

Een voorbeeld van een GWT ziet er zo uit:

GIVEN webhooks detects a notification in the ADT01 queue
WHEN the notification is of type cancelAdmin, Discharge or transfer
THEN the according user account is disabled using this API call:
/api/v1/CommunityMembers/BulkChangeStatus - status 2

GIVEN webhooks detects a notification in the ADT01 queue
WHEN the notification is of type admin, readmit, cancelDischarge, cancelTransfer or updateResidentInfo
THEN the user account is sent to the sync function that creates or updates the CC profile

Figuur 3 Voorbeeld van Given-When-Then

Als deze stappen succesvol doorlopen worden, is een user story uitgewerkt naar toebehoren.

Nadat alle *epics*, *features* en *user stories*, aangemaakt waren, is het ontwikkelproces gestart.

1.2.5 Uitwerkingsfase

Tijdens de uitwerkingsfase wordt er in sprints van twee weken lang gewerkt.. Er worden geen *sprint retrospectives* gehouden. In plaats daarvan wordt er wekelijks opgevolgd wat de vooruitgang is van de huidige sprint.

1.2.6 Use case 1: Importeren van PCC patiëntgegevens

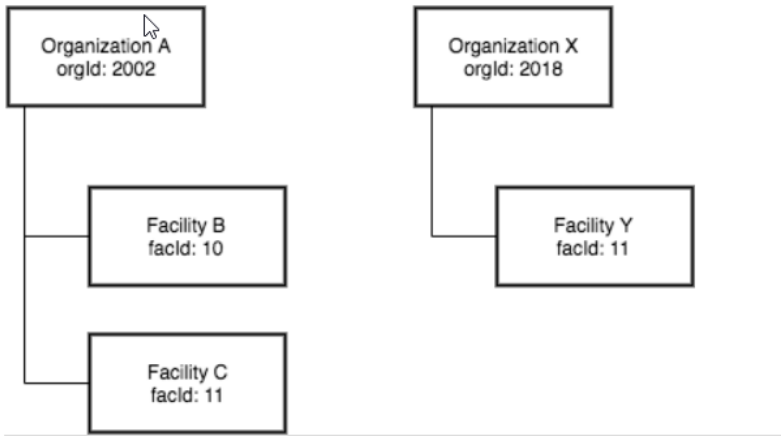
De eerste use case is de essentiële functionaliteit van de integratie: patiëntengegevens moeten uit de API van PCC gehaald worden, om vervolgens op basis van deze gegevens gebruikers aan te maken in het CCP.

1.2.6.1 Opbouw van het CCP

Binnen het CCP wordt gesproken over verschillende '*communities'*. Zo een *community* komt overeen met een residentie waar het CCP gebruikt wordt. Binnen deze *community* kunnen dan gebruikersaccounts aangemaakt worden, voor zowel inwoners van de residentie als het personeel dat actief is binnen de residentie. Hiervoor wordt een rollensysteem gebruikt, zodat meteen duidelijk is wie voor wat verantwoordelijk is binnen een *community*. In het geval van inwoners van de residentie bepaalt de rol welk zorgniveau ze nodig hebben.

1.2.6.2 Opbouw van PCC

PCC gebruikt een systeem met 'organisaties' en 'faciliteiten'. De organisatie is het hoogste niveau binnen de PCC-hiërarchie, en stelt de organisatie voor die alle faciliteiten managet. Deze organisatie komt overeen met een *community* binnen het CCP.



Figuur 4 Hiërarchie van PCC

De faciliteiten, die onder de organisatie vallen, stellen de effectieve afdelingen van een zorgcentrum voor waarin de patiënten verblijven. Elke patiënt die opgehaald wordt uit de PCC-API heeft dus niet enkel een patiënt ID, maar ook een faciliteit-ID en een organisatie-ID.

Als de PCC-API wordt aangesproken, wordt er telkens op organisatieniveau gewerkt: hiermee krijgt degene die de API aanspreekt toegang tot gegevens van alle faciliteiten die onder deze organisatie vallen, en de patiënten die in deze faciliteiten verblijven.

De PCC-API wordt natuurlijk niet zomaar opengesteld voor iedereen. Om toegang te verkrijgen tot de API van PCC, moet er eerst een validatieproces van PCC doorlopen worden. Hierbij wordt gekeken

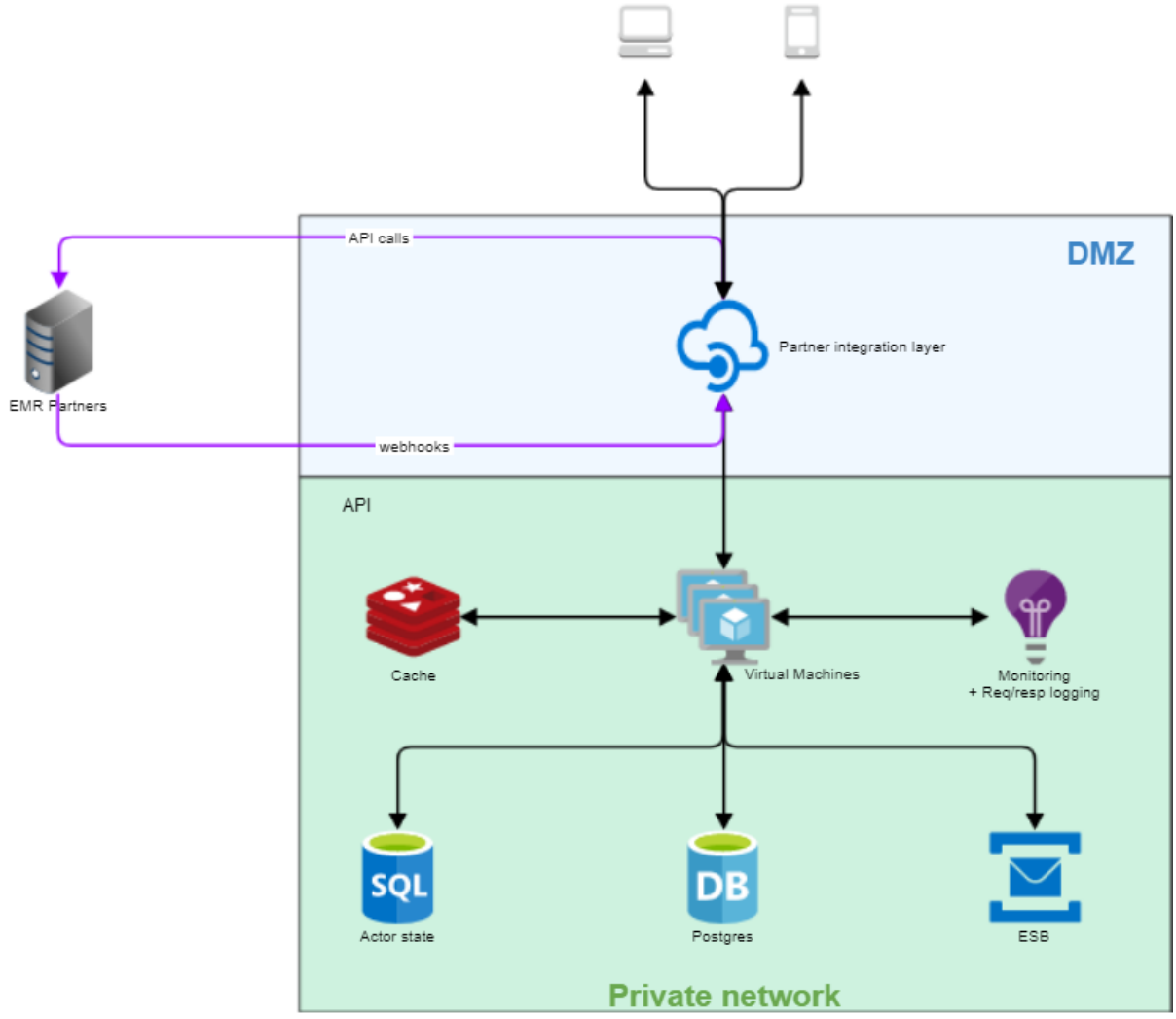
wat de applicatie precies is die gebruik wil maken van de PCC-API, maar ook welke API-functies er gebruikt gaan worden in de applicatie.

Eens deze validatie voltooid is, wordt er een 'app' aangemaakt binnen PCC. Deze app bevat een aantal gegevens over de applicatie die gebruik gaat maken van de PCC-API. Het belangrijkste uit deze gegevens is de client-ID en client-secret. Met deze velden kan de app zich autoriseren bij de PCC-API.

Deze app-gegevens kunnen bekeken worden via de website van PCC, aan de hand van een *developer-account*.

1.2.6.3 Flow van use case 1

De eerste stap van deze use case was een schema maken van de verschillende onderdelen van dit proces, en de verschillende technologieën die erbij komen kijken. In eerste instantie gaf de bedrijfspromotor wat uitleg over al de verschillende technologieën die gebruikt moesten worden in de use case. Vervolgens vond er een meeting plaats met de bedrijfspromotor en de softwarearchitect, die dan een schema heeft gemaakt van de verschillende technologieën. Hieronder is dit schema te zien:



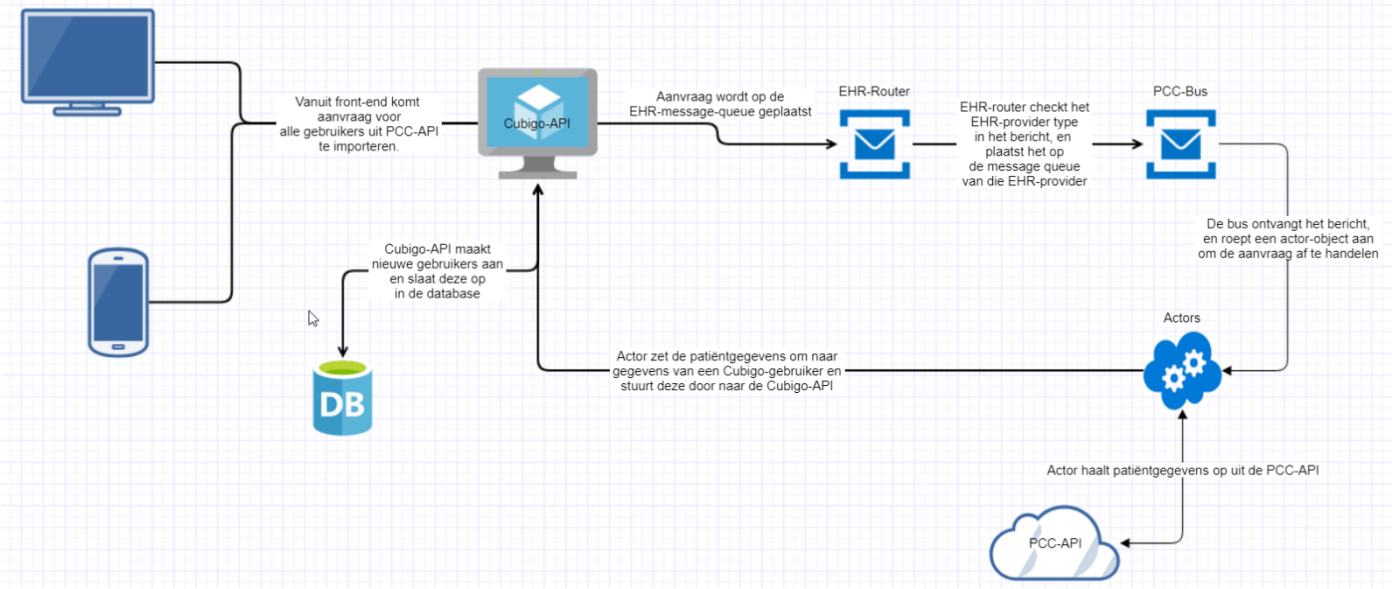
Figuur 5 Technologieschema

De DMZ in dit schema stelt de Azure Gateway door, die requests ontvangt en deze doorverwijst naar de juiste VM binnen het private-network van Cubigo. Op deze VM loopt de API dan ook.

De rode cache stelt een Redis-cache voor, die gebruikt wordt om frontend-gerelateerde gegevens tijdelijk in een cache op te slaan.

De uitvoering van de use case zou uiteindelijk zo moeten verlopen: er wordt een aanvraag gedaan om alle patiënten uit de PCC-API te importeren naar een *community* in het CCP. Deze aanvraag wordt vanuit de frontend van het CCP naar de Cubigo-API verstuurd. Deze API maakt dan een bericht aan, dat verstuurd wordt op de EHR-*message-queue*. In dit bericht zitten gegevens over de *community* waaruit het verstuurd is, en welke EHR-provider er gebruik wordt in de *community*.

Een router-programma luistert op deze *message-queue* voor berichten. Als de router een bericht ontvangt uit de queue, kijkt deze naar welke EHR-provider er in het bericht zit. Vervolgens gaat de router dan het bericht verder sturen, op de *message-queue* die bedoeld is voor deze EHR-provider, in dit geval PCC. De router stuurt het bericht verder op de PCC-*message-queue*. De 'PCC-bus' ontvangt dit bericht. Deze bus laat de aanvraag uit het bericht dan afhandelen door de *actors*. Hieronder is de flow uitgewerkt in een schema, om het allemaal wat te verduidelijken (flow begint links):



Figuur 6 Flow van eerste use case

Nadat dit schema gemaakt was, is het ontwikkelingsproces van de stageopdracht begonnen. De softwarearchitect had al een kleine POC gemaakt. Deze bevatte een basic API-*call* naar de API van PCC, met een werkende *authenticator*-klasse om te zorgen dat er toegang verkregen kon worden naar de PCC-API.

1.2.6.4 Verbinden met de PCC-API

Op basis van deze POC is er eerst een generische klasse gemaakt om de PCC-API aan te spreken. Deze klasse wordt gebruikt om zowel patiëntgegevens en gegevens over de faciliteiten uit de PCC-API op te halen.

De API-klasse maakt gebruikt van een *authenticator*-klasse. Deze *authenticator*-klasse haalt de *client-ID* en *client-secret* op uit de instellingen van het project waarin hij is opgeslagen. Deze worden dan gebruikt om een aanvraag te sturen naar een speciaal adres binnen de PCC-API. De PCC-API stuurt

dan een authenticatietoken terug, dat gebruikt wordt door de API-klasse om aanvragen te versturen naar de PCC-API.

```
public AccessTokenModel GetAccessToken()
{
    var client = new RestClient(BaseUrl) { Authenticator = new HttpBasicAuthenticator(username: ClientId, password: ClientSecret) };

    var request = new RestRequest(resource: $"{AuthPath}/token", Method.POST);
    request.AddHeader(name: "content-type", value: "application/x-www-form-urlencoded");
    request.AddParameter(name: "grant_type", value: "client_credentials");

    var response = client.Execute<AccessTokenModel>(request);
    if (!response.IsSuccessful)
    {
        throw new AuthenticationException();
    }

    return response.Data;
}
```

Figuur 7 Functie voor een authenticatietoken aan te vragen

1.2.6.5 *Actors*

Na het bouwen van de klasse om de API op te roepen, wordt er een *grain-interface* gemaakt, een *EHR-grain*. De functies van deze interface zijn verantwoordelijk voor 2 taken: patiëntgegevens ophalen uit de PCC-API, en deze doorsturen naar de Cubigo-API.

Van deze EHR-*grain* wordt een PCC-specifieke implementatie gemaakt. Voor elk type aanvraag dat de *grain* moet kunnen afhandelen (importeer één patiënt, updatet patiëntgegevens van deze gebruiker, importeer alle patiënten uit de PCC-API, etc.) wordt een methode geschreven.

Hieronder is de *'UpdateUser'*-methode te zien:

```
    public async Task UpdateUser(UpdateUserRequest updateUserRequest)
    {
        try
        {
            Patient patientToUpdate =
                await _patientApiService.GetPatientById(
                    updateUserRequest.OrganizationId,
                    updateUserRequest.PatientId);

            Facility patientFacility =
                await _facilityApiService.GetFacilityById(updateUserRequest.OrganizationId, patientToUpdate.FacId);

            var client = _cubigoRestClient.CreateCommunityMembersClient(Settings.Default.CubigoApiEndpoint,
                updateUserRequest.UserUuid);

            await client.UpdateEMRUserAsync(
                await PatientToCommunityMemberMapper.MapEMRPatientToUpdateModel(
                    patientToUpdate,
                    patientFacility,
                    updateUserRequest,
                    defaultPassword: "cubigo123"),
                patientToUpdate.PatientId,
                updateUserRequest.CommunityUuid.ToString(),
                accept_Language: "iv"
            );
        }
        catch (Exception ex)
        {
            throw new Exception(
                message: "An error occurred while updating the user. See inner exception for details.", ex);
        }
    }
```

Figuur 8 UpdateUser functie

Binnen deze functie worden eerst de patiëntgegevens van de Cubigo-gebruiker die geüpdatet moet worden, opgehaald, samen met de faciliteit waarin de patiënt verblijft.

Hierna wordt er een instantie van de Cubigo-API-client gemaakt. Dit client-object laat toe om aanvragen naar de Cubigo-API te sturen.

Met deze client wordt dan een functie aangeroepen, om de gebruikersgegevens in de Cubigo-API, die gelinkt zijn aan die patiënt, te vernieuwen. Eerst worden de patiëntgegevens omgezet naar de gegevens van een Cubigo-gebruiker, door middel van een *mapper*-klasse.

Vervolgens worden ook nog de *community-ID* en *patiënt-ID* doorgestuurd, zodat de Cubigo-API weet in welke *community* de gebruiker zit, en wat zijn/haar patiënt-ID is.

1.2.6.6 Mapping PCC Patient – Cubigo User

Bij het ophalen van de patiëntgegevens, moeten de gegevens van een patiënt omgezet worden naar die van een Cubigo-gebruiker. Veel van de informatievelden van een patiënt en Cubigo-gebruiker, zoals naam, e-mail, geboortedatum, etc. komen overeen tussen de twee systemen.

Het enige veld dat een probleem vormt is de rol die een Cubigo-gebruiker toegewezen krijgt. Deze rol bepaalt, bij inwoners van een residentie, welk soort verzorging ze nodig hebben. Helaas zitten deze rollen niet in de patiëntgegevens van een PCC-patiënt.

Dit probleem is opgelost door te kijken naar de faciliteit waarin een PCC-patiënt verblijft. Deze faciliteit is het zorgcentrum waarin de patiënt verzorgd wordt. Deze faciliteiten hebben wel een '*Line of Business'-veld*, waaruit afgeleid kan worden welk type patiënten er verzorgd worden. Via deze "Line Of Business" is er dan een *mapping* gemaakt met de rollen van het CCP.

De exacte mapping tussen de gegevens van een patiënt en een Cubigo-gebruiker is hier te zien:

Cubigo field	PCC field
FirstName	FirstName (patient)
LastName	LastName (patient)
BirthDate	BirthDate (patient)
Phone	Phone (patient)
Email	Email (patient)
Address1	AddressLine1 (facility)
Address2	RoomDesc (patient)
City	City (facility)
Country	Country (facility)
PostalCode	PostalCode (facility)
RegionCode	RegionCode (facility)
GenderType	Gender (patient)
CultureCode	LanguageCode (patient) + Country (facility)
Level0ExternalId	OrgUuid (patient)
Level1ExternalId	FacId (patient)
Level2ExternalId	PatientId (patient)

Figuur 9 Verhouding patiënt-gebruiker

1.2.6.7 Aanpassingen aan de Cubigo API

Uiteindelijk moet er nog altijd gecommuniceerd worden met de Cubigo-API, omdat hier de gebruikers effectief worden aangemaakt en opgeslagen in de database.

Eerst en vooral is er een nieuwe controller gemaakt, die alle EHR-gerelateerde API-taken afhandelt.

Om de Cubigo-API aan te roepen vanuit andere projecten, wordt gebruikt gemaakt van een API-client. Deze laat toe om een client-object aan te maken. Via dit object kan de API aangesproken worden. Zo wordt ook vermeden dat er verkeerde parameters naar de API gestuurd worden: de API-client volgt namelijk de *typing*-principes van .NET op.

Een gebruiker aanmaken in het CCP via de Cubigo-API is al mogelijk. Maar gebruikers die aangemaakt zijn op basis van EHR-patiëntengegevens moeten in de Cubigo API een speciale aanduiding krijgen, om ze te differentiëren van een gewoon gebruiker.

1.2.6.8 Aanpassingen aan de Cubigo API: nieuwe entiteiten

Hiervoor is beslist om twee nieuwe entiteiten aan te maken binnen de Cubigo-API: een *EHR-user* en *EHR-provider* entiteit. Deze entiteiten zijn generisch gemaakt, zodat ze ook met andere EHR-systemen gebruikt kunnen worden, en niet enkel PCC. De EHR-user entiteit bevat alle nodige EHR-gerelateerde informatie over de patiëntgegevens waarop een Cubigo-gebruiker is aangemaakt, zoals zijn patiënt-ID binnen het EHR-systeem waar zijn patiëntgegevens in zitten, de ID van de faciliteit waar de patiënt verblijft en de ID van de EHR-organisatie waartoe ze behoren. Deze ID's zijn nodig om gegevens van een patiënt op te halen uit de PCC-API.

Deze EHR-gebruiker entiteit wordt gelinkt aan een Cubigo-gebruiker entiteit, en aan een EHR-provider entiteit. Op deze manier wordt gekeken of een Cubigo-gebruiker een gelinkte EHR-gebruiker entiteit heeft, en aan welke EHR-provider deze EHR-user gelinkt is.

Naar het einde van de use case toe is er beslist om nog een extra entiteit aan te maken binnen de Cubigo-API: de EHR-facility. Deze entiteit is vooral aangemaakt omdat, tot nu toe, er altijd vanuit werd gegaan dat bij de initiële import uit de PCC-API, waarbij de gegevens van alle patiënten uit een PCC-organisatie opgehaald worden, patiëntgegevens uit elke faciliteit binnen de organisatie opgehaald moesten worden. De PO heeft dan beslist om een manier toe te voegen dat er gekozen kan worden uit welke faciliteiten binnen een PCC-organisatie er gebruikers geïmporteerd worden.

Als er bijvoorbeeld enkel een EHR-facility entiteit met ID 22 in een Cubigo-*community* zit, worden enkel patiëntgegevens uit de faciliteit met ID 22 opgehaald, wanneer er een import van patiëntgegevens wordt gedaan.

1.2.6.9 Aanpassingen aan de Cubigo API: nieuwe *community*-instellingen

Als laatste aanpassing binnen de Cubigo-API zijn er nieuwe instellingen aangemaakt. Elke *community* binnen het CCP heeft een aantal instellingen: bijvoorbeeld welke services er allemaal actief zijn in de *community*, welke functionaliteit aan en uit staat, de standaard taal van de *community*, etc.

Bij deze instellingen zijn er twee extra instellingen aangemaakt: een instelling om de EHR-functionaliteit van een *community* aan of uit te zetten, en een andere instelling die de naam van de EHR-provider van de *community* bevat. Deze laatste is ook gelinkt aan een EHR-provider entiteit.

1.2.6.10 Communicatie tussen de Cubigo-API en de *actors*

Om de Cubigo-API te laten communiceren met de *actors*, worden er gebruikt gemaakt van *message-queues*, zoals beschreven in de flow van de use case.

Hiervoor wordt eerst een soort 'router' gemaakt. Deze luistert op een EHR-*message-queue*. Als er een bericht binnenkomt dat afgeleid is van de klasse *'EHRBaseMessage'*, wordt deze door de router ontvangen. De router gaat dan vervolgens kijken welke EHR-Provider er in het bericht staat.

Vervolgens wordt het bericht verder gestuurt op een andere *'message-queue'*. Deze queue is specifiek gemaakt voor de EHR-provider die in het bericht staat, in dit geval de *'PCC-message-queue'.*

Op deze *message-queue* luistert dan de PCC-bus. Deze ontvangt het bericht en maakt een *grain*-client aan. Deze *grain* client wordt dan gebruikt om de *actors* aan te spreken, om een taak uit te voeren. Het type taak hangt af van het bericht type. Hieronder is een functie te zien van de '*message-handler'* in de PCC-bus:

```
public class MessageHandler :
    IHandleMessages<ImportSingleUserCommand>,
    IHandleMessages<ImportAllUsersFromFacilityCommand>,
    IHandleMessages<ImportAllUsersFromOrganizationCommand>,
    IHandleMessages<UpdateUserCommand>,
    IHandleMessages<SyncUserCommand>,
    IHandleMessages<SyncAllUsersFromOrganizationCommand>,
    IHandleMessages<SubscribeToWebhooksCommand>
{

    private IActorService _actorService;

    public MessageHandler(IActorService actorService)
    {
        _actorService = actorService;
    }

    public async Task Handle(ImportSingleUserCommand message, IMessageHandlerContext context)
    {
        ImportSingleUserRequest actorRequest =
            CommandToGrainRequestMapper.ConvertSingleUserCommand(message);

        var grain = await _actorService.GetPCCGrain();

        await grain.ImportUser(actorRequest);
    }
```

Figuur 10 Code van PCC-bus message-handler

De klasse die de berichten afhandelt, implementeer de '*IHandleMessages'*-interface, met een type van bericht meegegeven. Vervolgens wordt er per type een '*Handle'*-functie aangemaakt. Hier is de *handle*-functie te zien die één gebruiker importeert. Eerst wordt het bericht omgezet naar een *'grain-request'*. Vervolgens wordt er een *grain* aangeroepen, die de effectieve import gaat uitvoeren.

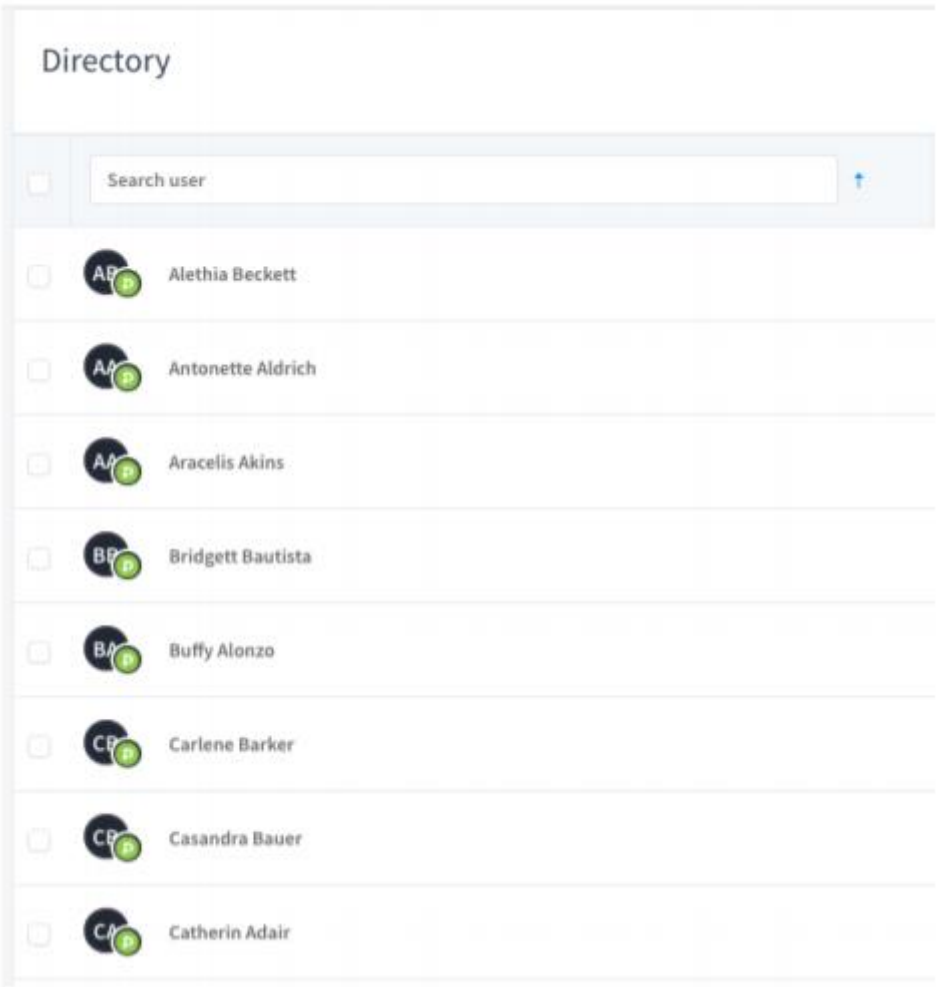
Dankzij dit systeem kan er vanuit de Cubigo-API gewoon een bericht geplaatst worden op de *EHR-message-queue.* Op deze manier hoeft de Cubigo-API zich geen zorgen te maken over wat er met het bericht gebeurd.

1.2.6.11 Aanpassing aan de frontend: weergave van een gebruiker

Net zoals een Cubigo-gebruiker in de Cubigo-API een manier moet hebben om herkend te worden als een geïmporteerde patiënt, moest dit ook in de frontend van het CCP gebeuren.

In de frontend wordt een gebruiker die een gelinkte EHR-gebruiker heeft aangeduid met enkele speciale markeringen:

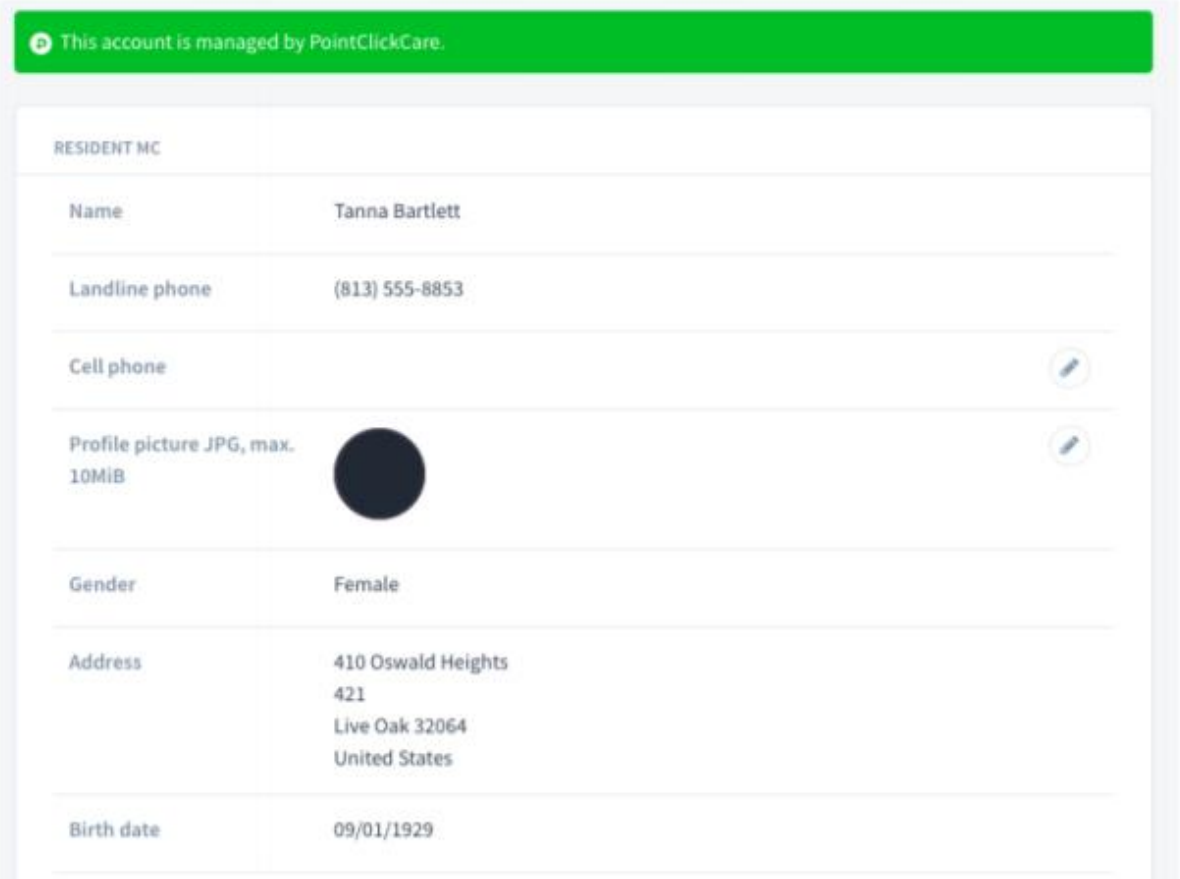
- Op het *user-management* scherm van het CCP worden geïmporteerde gebruikers aangeduid met een klein icoontje onder hun profielfoto. Dit icoontje is het logo van de EHR-provider waaruit hun patiëntengegevens opgehaald zijn, zoals hieronder te zien



Figuur 11 Aanduiding geïmporteerde gebruiker

- Op de profielpagina van een Cubigo-gebruiker, komt 'This account is managed by provider.' te staan, waar 'provider' wordt ingevuld door de naam van de EHR-provider van de Cubigo-gebruiker.

Hieronder is een te zien hoe de balk eruit zou zien als een gebruiker uit PCC geïmporteerd is:



Figuur 12 PCC markering op gebruikersprofiel

Op hun profielpagina kunnen gebruikers enkele van hun gegevens, zoals naam, geboortedatum, telefoonnummer, etc. aanpassen. Deze mogelijkheid tot aanpassen wordt uitgezet bij een geïmporteerde gebruiker, omdat de relatie tussen de PCC-API en het CCP een '*master-slave*' relatie is: het CCP neemt gegevens op uit de PCC-API, maar stuurt nooit gegevens terug. Daarom mogen deze velden dan ook niet aanpasbaar zijn, omdat anders de relatie geschonden wordt.

Om al deze functionaliteit werkend te krijgen, moet er eerst een verandering gemaakt worden in twee data-modellen die de frontend ophaalt uit de Cubigo-API: de *community*-instellingen, en het model dat de data van een gebruikersprofiel bevat.

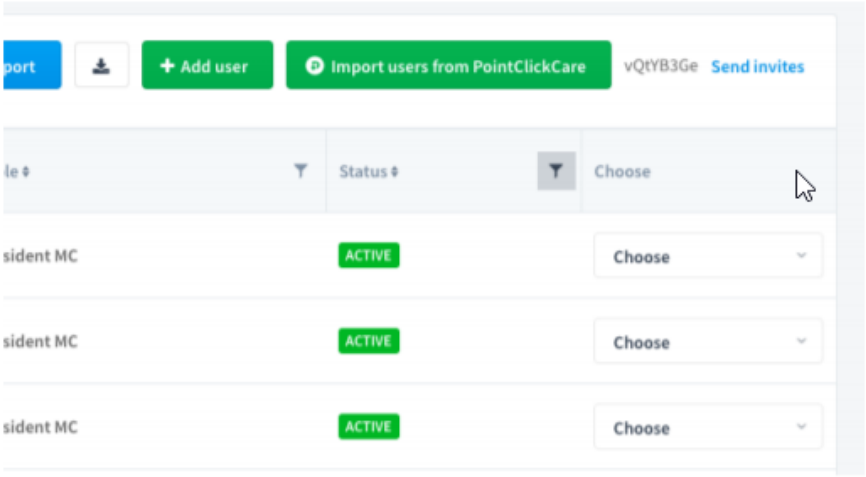
In het *community*-instellingen-model zijn 2 extra velden bijgevoegd, om de nieuwe EHR-instelling die aangemaakt zijn in de Cubigo-API ook door te voeren naar de frontend.

In het model van de gebruikersprofiel is een nieuw veld ingevoegd, dat de naam van de EHR-provider van de gebruiker bevat. Als een gebruiker geïmporteerd is uit de PCC-API, dan wordt dit veld ingevuld met 'PCC'.

Met deze extra instellingen weet de frontend wanneer de EHR-functionaliteit zichtbaar of onzichtbaar moet zijn.

1.2.6.12 Aanpassing aan de frontend: functioneel

Om de eerste use case af te sluiten, moet er nog een functie toegevoegd worden aan de frontend: een manier om het importeren van gebruikers uit de PCC-API te starten. Hiervoor werd een knop toegevoegd op de user-management pagina. De knop is enkel zichtbaar voor gebruikers met de rol '*General Admin'*. Deze rol wordt enkel gebruikt bij de initiële configuratie van een *community*, en is dus enkel zichtbaar voor de persoon die de initiële configuratie van de *community* heeft gedaan.



Figuur 13 Knop voor het importeren te starten

Bij het klikken op deze knop wordt de Cubigo-API aangeroepen, die dan het volledige import-proces start. Ook wordt er een *pop-up*bericht getoond aan de gebruiker, dat het importproces gestart is.

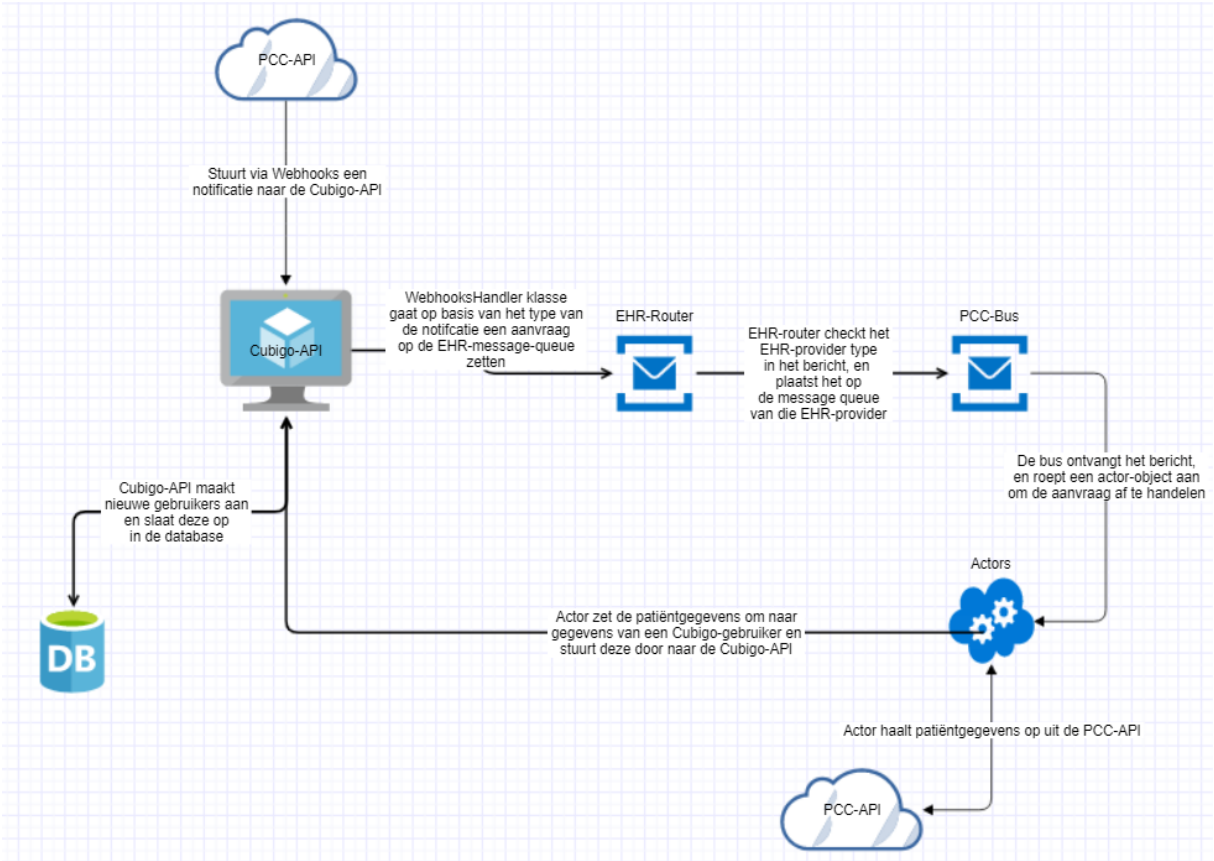
1.2.7 Use case 2: geïmporteerde gebruikers in sync houden met patiëntengegevens

De tweede use case van de stage opdracht is het synchroon houden van gebruikersgegevens van gebruikers die uit de PCC-API geïmporteerd zijn. Hier biedt de PCC-API zelf al een oplossing voor: een notificatiesysteem, dat gebruikt maakt van *Webhooks*.

Webhooks is een notificatiesysteem waarbij een ontvanger zich 'inschrijft' bij de zender van de notificaties. Bij dit proces geeft de ontvanger een '*callback*-URL' door aan de zender, samen met het type notificaties die de ontvanger wil krijgen. Eens de 'inschrijving' succesvol ontvanger is door de zender, begint deze notificaties te sturen naar de '*callback*-URL' van de ontvanger.

In het geval van de PCC-API worden er notificaties gestuurd als de status van een patiënt verandert, bijvoorbeeld als de patiënt verplaatst wordt naar een andere faciliteit of als hij/zij tijdelijk een faciliteit verlaat.

Hieronder is de flow van voor het ontvangen van *Webhooks* in een schema gestoken:



Figuur 14 Flow van tweede use case

Zoals te zien is het schema niet echt verschillend van de eerste use case. Het enige grote verschil is dat de Cubigo-API notificaties ontvangt vanuit de PCC-API, in plaats van een aanvraag vanuit de frontend

1.2.7.1 Ontvangen van *Webhooks*-notificaties

Bij het begin van de ontwikkeling van deze use case, heeft er eerst een meeting met de software-architect van Cubigo plaatsgevonden. Tijdens deze meeting heeft hij gekeken of er aanpassingen aan de architectuur moesten gemaakt worden om deze notificaties te ontvangen.

Tijdens het testen van de gehele flow voor het ontvangen van de notificaties, is er een belangrijk probleem opgedoken: voor het succesvol ontvangen van de notificaties, stelt PCC namelijk een aantal eisen omtrent de URL waar de notificaties naartoe gestuurd worden: deze moet namelijk een statisch IP-adres hebben, samen met een uniek SSL-certificaat. Sinds het wat tijd kost om te zorgen dat de Cubigo-API aan deze eisen voldoet, wordt er beslist om voorlopig de notificaties te simuleren, door deze aan de hand van een tool zelf te genereren en door te sturen naar de Cubigo-API.

Uiteindelijk loste de software-architect dit op door, via Azure, een statisch IP-adres aan te vragen, samen met nieuwe SSL-certificaten. Dit IP-adres wordt dan opengesteld via een unieke *Webhooks*-URL, die de binnenkomende notificaties dan doorstuurt naar de Cubigo-API.

1.2.7.2 Aanpassingen aan de Cubigo-API

Voor het ontvangen en afhandelen van de *Webhooks*-notificaties is een nieuwe *Webhooks*-controller aangemaakt binnen de Cubigo-API, die instaat voor het afhandelen van notificaties verstuurd vanuit de PCC-API.

Wanneer de controller een notificatie ontvangt, wordt deze doorgegeven aan een service, die vervolgens gaat kijken welk type notificatie het is. Afhankelijk van het type notificatie (patiënt wordt ontslagen uit zijn faciliteit, patiënt gaat tijdelijk op vakantie, patiëntgegevens zijn geüpdatet, etc.) genereert de service een bericht, dat dan naar de EHR-router wordt doorgestuurd, om uiteindelijk afgehandeld te worden door een EHR-*grain*.

1.2.7.3 Aanpassingen aan de *actors* en *message-queues*

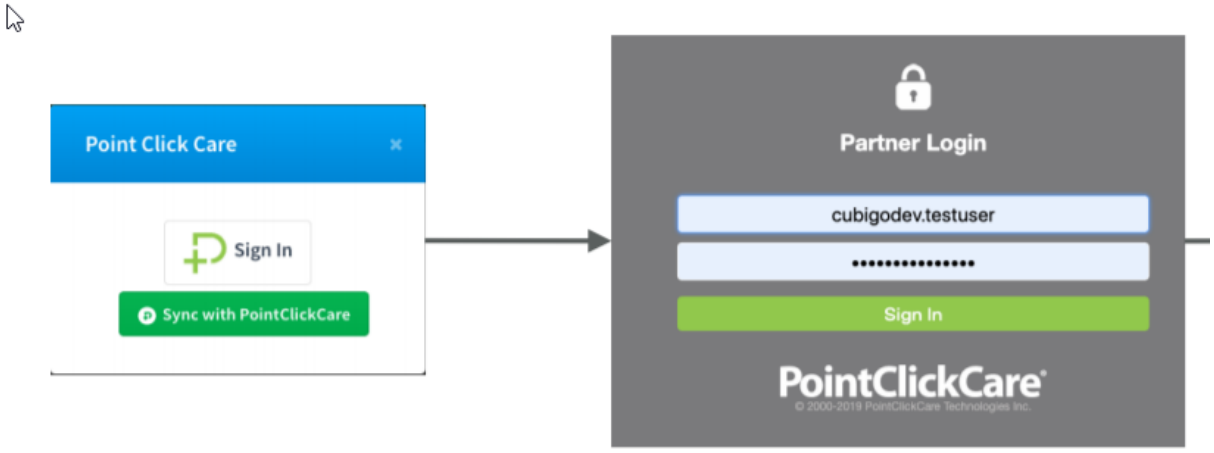
Bij het afhandelen van de *Webhooks*-notificaties, wordt ondervonden dat wanneer een patiënt zijn ID verandert binnen de PCC-API, hetzelfde type notificatie wordt verstuurd als wanneer er een nieuwe patiënt opgenomen wordt in een faciliteit.

Hiervoor moet er dus een nieuwe functie binnen de *actor*-klasse geschreven worden. Deze functie checkt of een patiënt-ID gelinkt is aan een bestaande Cubigo-gebruiker, en vervolgens de Cubigo API aanspreekt om deze user te updaten, of om een nieuwe Cubigo-gebruiker aan de maken, als er aan de patiënt-ID geen Cubigo-gebruiker gelinkt is.

Deze functie is ook in gebruik genomen bij de initiële import van alle gebruikers uit de PCC-API, zodat deze import ook gebruikt kan worden als er al enkele geïmporteerde gebruikers bestaan in een *community*, zodat dezelfde gebruikers niet opnieuw geïmporteerd worden.

1.2.8 Nice-to-have: *3-legged* authenticatie

Tijdens het uitwerken van de tweede use case is ook beslist om de zogenaamde '*3-legged* authenticatie' ook al uit te werken. Dit is een authenticatiesysteem dat PCC gebruikt, waarbij een gebruiker vanuit het CCP doorverwezen wordt naar een loginpagina van PCC. Hier moet de Cubigo-gebruiker inloggen met zijn/haar PCC-account, om vervolgens een authenticatietoken te verkrijgen, dat gebruikt kan worden om specifieke patiëntgegevens op te halen van de patiënt die gelinkt is aan dat PCC-account.



Figuur 15 Login knop en loginscherm van PCC

Deze vorm van authenticatie is voor de huidige use cases nog niet nodig, maar wel naar de toekomst toe: in enkele toekomstige use cases wil Cubigo specifieke patiëntgegevens uit de PCC-API halen, zoals allergie-informatie, gezondheidsinformatie, etc. Voor dit soort gegevens op te halen is deze '*3-legged* authenticatie' wel nodig.

1.2.8.1 Aanpassingen aan de frontend

In de frontend van het CCP is een nieuwe knop geïmplementeerd, op de user-management pagina: een *'Sign In'* knop, die de gebruiker doorverwijst naar de loginpagina van PCC. Hier moet de gebruiker zich inloggen met zijn PCC-account. Vervolgens wordt de gebruiker teruggebracht naar het user-management scherm. In de query-parameters van de URL zit nu een code-parameter, die de authenticatiecode van de gebruiker bevat.

Deze code wordt dan naar de Cubigo-API doorgestuurd, die het authenticatieproces verder afhandelt. De API stuurt dan een authenticatietoken terug, dat wordt opgeslagen in een tijdelijk cache en gebruikt kan worden voor de PCC-API operaties waarbij *3-legged* authenticatie nodig is.

1.2.8.2 Aanpassingen aan de Cubigo-API

In de Cubigo-API is er een nieuwe functie toegevoegd aan de EHR-controller. Deze functie ontvangt de authenticatiecode van de frontend, en geeft deze door aan een authenticatieservice.

Deze service stuurt de code dan, samen met de PCC-API-*credentials*, door naar het authenticatie-endpoint van de PCC-API, die dan een authenticatietoken teruggeeft. Dit token wordt dan doorgestuurd naar de frontend van het CCP.

1.3 Reflectie stageopdracht

De stageopdracht is een zeer leerzame en interessante opdracht geweest om aan te werken.

In het begin van de stage heb ik even wat moeilijkheden gehad met het begrijpen van al de verschillende technologieën die ik in de stage moest gebruiken, zeker omdat het allemaal in één keer op me afkwam. Vooral het gebruik van de messaging en de *actors* was in het begin allemaal redelijk moeilijk te begrijpen, maar door de bestaande code te bekijken, en voldoende uitleg te krijgen van de bedrijfspromotor, is het me uiteindelijk allemaal duidelijk geworden. Ik ben nu ook blij dat ik eens met messaging en *actors* gewerkt heb, omdat het toch wel interessante technologieën zijn. Vooral *actors*, waar ik voor de stage nog nooit van gehoord had.

Wat ik zelf zeer interessant vond was om eindelijk eens in een 'echte' omgeving te werken, en ook samen te werken met de andere leden van het ontwikkelingsteam.

Ik ben ook wel blij dat ik tijdens de stageopdracht ook geconfronteerd ben met mijn slechtere kantjes, waar ik dan ook werkpunten van wil maken. Waar ik vooral problemen mee heb gehad, was het vragen om hulp. Ik heb heel snel de neiging om gewoon volledig zelfstandig te werken, en te proberen zelf problemen op te lossen, zonder anderen om hulp te vragen. Ik denk altijd dat als ik anderen om hulp vraag, dat ik hen aan het storen ben, terwijl dat meestal niet het geval is. Zo kan ik wel eens snel tijd verliezen, door mezelf constant vast te bijten in een probleem dat ik wil oplossen.

De wekelijkse opvolging door Cubigo heeft me ook geholpen met een probleem dat ik nog wil aanpakken: uitstelgedrag. Dankzij deze wekelijkse opvolging kreeg ik op het einde van de week telkens een overzicht van wat er nog gedaan moest worden. Hierdoor kreeg ik een betere push om te zorgen dat ik bij deze wekelijkse meetings nooit in de problemen kwam met taken die te laat waren afgewerkt.

Wat ik ook wel leuk vond is dat er ook taken gedaan moesten worden die normaalgezien niet altijd door een applicatieontwikkelaar gedaan moeten worden, zoals een Jenkins-server instellen om automatisch *builds* aan te maken en zelf *user stories* en *features* te schrijven, wat normaal door de softwaremanagers wordt gedaan. Ik vind het persoonlijk belangrijk dat een applicatieontwikkelaar deze dingen ook kan.

2 Onderzoeksopdracht

2.1 Onderwerp

Business Intelligence is “een door technologie gedreven proces voor het analyseren van gegevens en het presenteren van actiegerichte informatie om werknemers, managers en andere zakelijke eindgebruikers te helpen met het nemen van geïnformeerde businessbeslissingen.” [1]

BI kan op veel manieren gebruikt worden, zowel uitgebreid als simpel. De eenvoudigste toepassing van BI is het visualiseren van gegevens en hieruit conclusies trekken. Dit is dan ook waarvoor het meestal gebruikt wordt.

Volgens Gartner, een van de grootste onderzoeksbedrijven actief in de IT-sector, zijn Power BI van Microsoft, Tableau van Tableau Software, en Qlik Sense van Qlik. Hieronder is dit te zien in het ‘*Magic Quadrant*’ van Gartner:

Figuur 16 Magic Quadrant van BI-tools [17]

2.2 Probleemstelling

Voor Cubigo komt de Return On Investment voornamelijk uit het welzijn van de residenten binnen de gemeenschappen waarin het CCP gebruikt wordt. Cubigo wil dus zo goed mogelijk het welzijn van de residenten kunnen monitoren en verbeteren waar dit mogelijk is, aan de hand van data die beschikbaar is binnen het CCP. Daarom zoekt Cubigo een manier om de data te analyseren en gebruiken om het welzijn van de residenten te verbeteren.

BI is een van de processen die kan gebruikt worden voor deze analyse.

Veel bedrijven hebben tegenwoordig nood aan BI om de grote hoeveelheid data die ze ter beschikken hebben te kunnen verwerken, om op basis van deze data beslissingen te kunnen nemen die het bedrijf positief beïnvloeden.

Een BI-tool is maar zo goed als de datasets waarmee het moet werken. In dit onderzoek wordt gewerkt met een dataset over de opkomst van studenten naar school. De opkomst is per dag gemeten, in percent, over een periode van drie jaar.

Deze datasets worden dan aan een BI-tool gegeven. BI-tools hebben meestal een groot aantal functies. Hieronder zijn enkele van de belangrijkste opgesomd:

- *Dataviews* (tabellen, grafieken, etc.) van een dataset maken;
- Dashboards aanmaken;
- Rapporten genereren, over bijvoorbeeld KPI's, opgegeven *metrics*, etc;
- *Queries* uitvoeren op datasets.

De drie tools die hier vergeleken gaan worden, zijn Power BI, Tableau en Qlik Sense. Deze zijn gekozen omdat ze de populairste spelers op de markt zijn.

2.3 Hoofdvraag

Welke BI-tool is het best geschikt voor Cubigo?

Om te kijken welke BI-tool het beste is voor Cubigo zijn er volgende criteria opgesteld:

- Gebruiksvriendelijkheid: kan de BI-tool ook gebruikt worden door een niet-IT'er? In principe zou de data die in de tool gebruikt wordt gefilterd worden door een IT'er, die de data vereenvoudigd en duidelijker maakt. Vervolgens zou een niet IT'er aan de slag moeten kunnen gaan met deze data.
- Deelbaarheid: de *dataviews* zouden gedeeld moeten kunnen worden met andere gebruikers. Tot welke mate is dit mogelijk met de BI-tools?
- Data connectiviteit: welke dataproviders ondersteunt de BI-tool?
- Prijs: liefst zou de BI-tool zo goedkoop mogelijk moeten zijn.
- Kwaliteit van documentatie: is er voldoende documentatie beschikbaar voor de tool? Wat is de kwaliteit ervan?

2.3.1 Deelvragen

- Wat moet een gebruiker doen in de BI-tool om een *dataview* en filter aan te maken?
- Kunnen de aangemaakte *dataviews* gedeeld worden met andere personen?
- Welke databronnen ondersteunt de BI-tool?
- Wat is de kwaliteit van de documentatie die beschikbaar is voor de BI-tool?
- Wat zijn de prijsverschillen tussen de tools?

2.4 Onderzoeksmethode

Er wordt een vergelijking gemaakt tussen deze drie tools: Microsoft Power BI, Qlik Sense en Tableau. Eerst zal er een literatuurstudie gehouden worden, waarbij er gekeken wordt wat de verschillende databronnen zijn die elke BI-tool ondersteunt, de verschillende prijsklassen en versies van de tools die er bestaan en de mogelijkheden om aangemaakte *dataviews* met anderen te delen.

Vervolgens zal er een *Proof-Of-Concept (POC)* opgesteld worden: er wordt een Postgres-database aangemaakt, met daarin een dataset die gegevens bevat over de opkomst van studenten naar school, verspreid over drie schooljaren. Deze data komt overeen met een van de doelen die Cubigo voor ogen heeft met de BI-tools: kijken hoeveel procent van de inwoners van een gemeenschap deelnemen aan een bepaalde activiteit, zodat er kan gekeken worden welke activiteiten populair zijn en welke niet.

Dan worden er in elke BI-tool enkele *dataviews* aangemaakt. Het proces van het aanmaken van deze *dataviews* wordt in stappen gedocumenteerd, met afbeeldingen. Zo wordt duidelijker hoe de BI-tool werkt en hoe de *userinterface* eruit ziet. Dit vormt dat een antwoord op de eerste deelvraag.

In de conclusie van het onderzoek worden de resultaten van elke deelvraag vergeleken, en, waar mogelijk, in een vergelijkingsmatrix gestoken, zodat er gekeken kan worden welke tool het beste is.

3 Uitwerking Onderzoek

3.1 Microsoft Power BI

Voor het POC-gedeelte van het onderzoek wordt de gratis versie van Power BI, genaamd Power BI Desktop, gebruikt.

3.1.1 Pricing [2]

Microsoft Power BI maakt gebruik van het '*Software-as-a-Service*'-model, wat wil zeggen dat gebruikers een maandelijkse premie moeten betalen voor Power BI te gebruiken. Microsoft biedt momenteel drie verschillende versies van Power BI aan: een Pro versie, een Premium versie en een gratis versie.

3.1.1.1 Power BI Desktop

De gratis versie van Power BI neemt de vorm aan van Power BI Desktop, een app die gratis gedownload kan worden. Via deze app kan gebruik gemaakt worden van de meeste features die Power BI aanbiedt, met als grote uitzondering dat gemaakte apps niet gedeeld kunnen worden met andere gebruikers.

3.1.1.2 Power BI Pro

Power BI Pro is de simpelste en goedkoopste versie die Microsoft aanbiedt. Microsoft raadt deze versie aan als een gebruiker enkel nood heeft aan een '*self-service*' BI-pakket. Power BI Pro kost $9,99 per maand, aangerekend per gebruiker. Deze versie van het Power BI-pakket zit ook inbegrepen in het Microsoft Office 365 Enterprise E5 pakket. Het grootste verschil met de gratis versie van Power BI is dat een gebruiker zijn aangemaakte rapporten kan delen met andere Power BI Pro-gebruikers. Deze versie ondersteunt datasets tot 1 GB groot, en stelt 10 GB opslag per gebruiker beschikbaar.

3.1.1.3 Power BI Premium

Power BI Premium is de duurste versie van Power BI, en is bedoeld voor grotere bedrijven waar veel werknemers gebruikmaken van Power BI. De prijs van deze versie wordt ook maandelijks aangerekend, maar wordt niet aangerekend per gebruiker: de prijs wordt aangerekend per *'node'* die het bedrijf gebruikt.

Deze *nodes* zijn servers waarop de Power BI service loopt. Elke *node* heeft een bepaalde capaciteit van dataopslag en is een geïsoleerd systeem, in tegenstelling tot Power BI Pro, waarbij verschillende gebruikers van verschillende organisaties allemaal op gedeelde servers werken. Deze versie biedt 100 TB aan opslag aan (per *node*), en ondersteunt datasets tot 10 GB groot. De prijs per *node* komt neer op $4995 per maand.

Microsoft biedt op de Power BI-website ook een optie aan om een consultatie met een Microsoft-medewerker te plannen, zodat bekeken kan worden hoeveel *nodes* een gebruiker nodig heeft.

3.1.2 Verbinden met data

Power BI laat gebruikers toe om met een gigantisch aantal dataproviders te verbinden. Daarom zijn de providers binnen Power BI verdeeld onder enkele categorieën.

De eerst categorie omvat de meest eenvoudige om met data te verbinden: een bestand in te laden. Dit kan een Excel-bestand, een CSV-bestand, JSON-bestand, etc. zijn.

Als tweede categorie zijn er de verschillende databases waarmee Power BI kan verbinden. Deze houden in:

Tabel 1 Lijst van databases die Power BI ondersteunt [3]

SQL Server Database	Access Database
SQL Server Analysis Services Database	Oracle Database
IBM DB2 Database	IBM Informix database (Beta)
IBM Netezza	MySQL Database
PostgreSQL Database	Sybase Database
Teradata Database	SAP HANA Database
SAP Business Warehouse Application Server	SAP Business Warehouse Message Server
Amazon Redshift	Impala
Google BigQuery	Vertica
Snowflake	Essbase
AtScale cubes (Beta)	BI Connector
Dremio	Exasol
Indexima (Beta)	InterSystems IRIS (Beta)
Jethro (Beta)	Kyligence Enterprise (Beta)
MarkLogic (Beta)	

De derde categorie omvat Power BI-datasets. In Power BI kunnen datasets aangemaakt worden en vervolgens ook opgeslagen worden. Deze kunnen dan ook als dataprovider gebruikt worden.

In de vierde categorie bevinden zich dataproviders uit Microsoft Azure. Deze houden in:

Tabel 2 Azure dataproviders die Power BI ondersteunt [3]

Azure SQL Database	Azure SQL Data Warehouse
Azure Analysis Services database	Azure Blob Storage
Azure Table Storage	Azure Cosmos DB (Beta)
Azure Data Lake Storage Gen1	Azure HDInsight (HDFS)
Azure HDInsight Spark	HDInsight Interactive Query
Azure Data Explorer (Kusto)	Azure Cost Management (Beta)

Als vijfde categorie zijn er nog de verschillende online services waarmee Power BI kan verbinden. Deze houden in:

Tabel 3 Online services die Power BI ondersteunt [3]

SharePoint Online List	Microsoft Exchange Online
Dynamics 365 (online)	Dynamics NAV
Dynamics 365 Business Central	Dynamics 365 Business Central (on-premises)
Common Data Service for Apps (Beta)	Microsoft Azure Consumption Insights (Beta)
Azure DevOps (Beta)	Azure DevOps Server (Beta)
Salesforce Objects	Salesforce Reports
Google Analytics	Adobe Analytics
appFigures (Beta)	Data.World - Get Dataset (Beta)
Facebook	GitHub (Beta)
MailChimp (Beta)	Marketo (Beta)
Mixpanel (Beta)	Planview Enterprise One - PRM (Beta)
Planview Projectplace (Beta)	QuickBooks Online (Beta)
Smart*sheet*	SparkPost (Beta)
Stripe (Beta)	SweetIQ (Beta)
Planview Enterprise One - CMT (Beta)	Twilio (Beta)
tyGraph (Beta)	Webtrends (Beta)
Zendesk (Beta)	Emigo Data Source (Beta)
IndustrialAppStore (Beta)	Microsoft Graph Security (Beta)
TeamDesk (Beta)	

De laatste categorie bevat enkele dataproviders die niet onder bovenstaande categorieën vallen, zoals Python-scripts en R-scripts. Deze zijn hieronder opgesomd:

Tabel 4 'Andere' dataproviders die Power BI ondersteunt

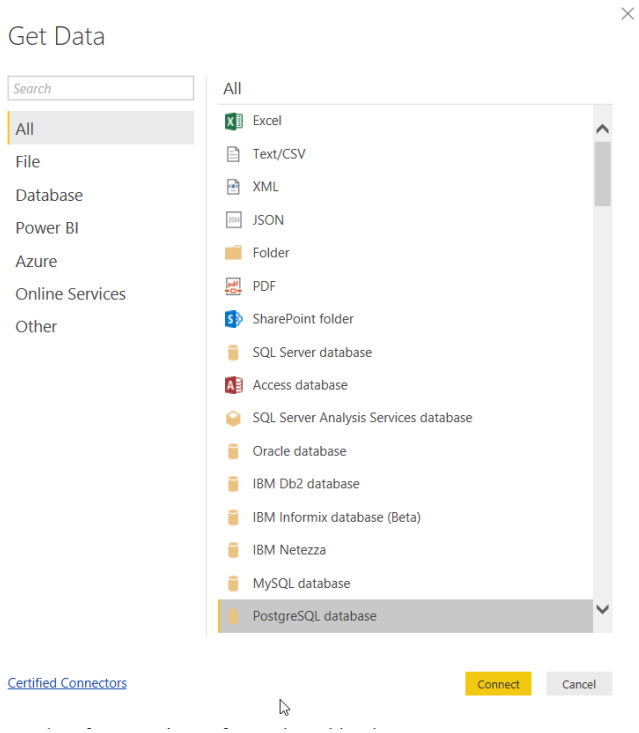
Web	SharePoint List
OData Feed	Active Directory
Microsoft Exchange	Hadoop File (HDFS)
Spark	R Script
Python script	ODBC
OLE DB	BI360 - Budgeting & Financial Reporting (Beta)
Denado	Information Grid (Beta)
Paxata	QubolePresto (Beta)
Quick Base (Beta)	Roamler (Beta)
SurveyMonkey (Beta)	Tenforce (Beta)
Workforce Dimensions (Beta)	Blank Query

3.1.3 Een *dataview* aanmaken

Bij het opstarten van Power BI, wordt er een nieuw rapport aangemaakt. Binnen dit rapport kan een gebruiker verschillende pagina's aanmaken, waarin *dataviews* geplaatst kunnen worden.

3.1.3.1 Data inladen

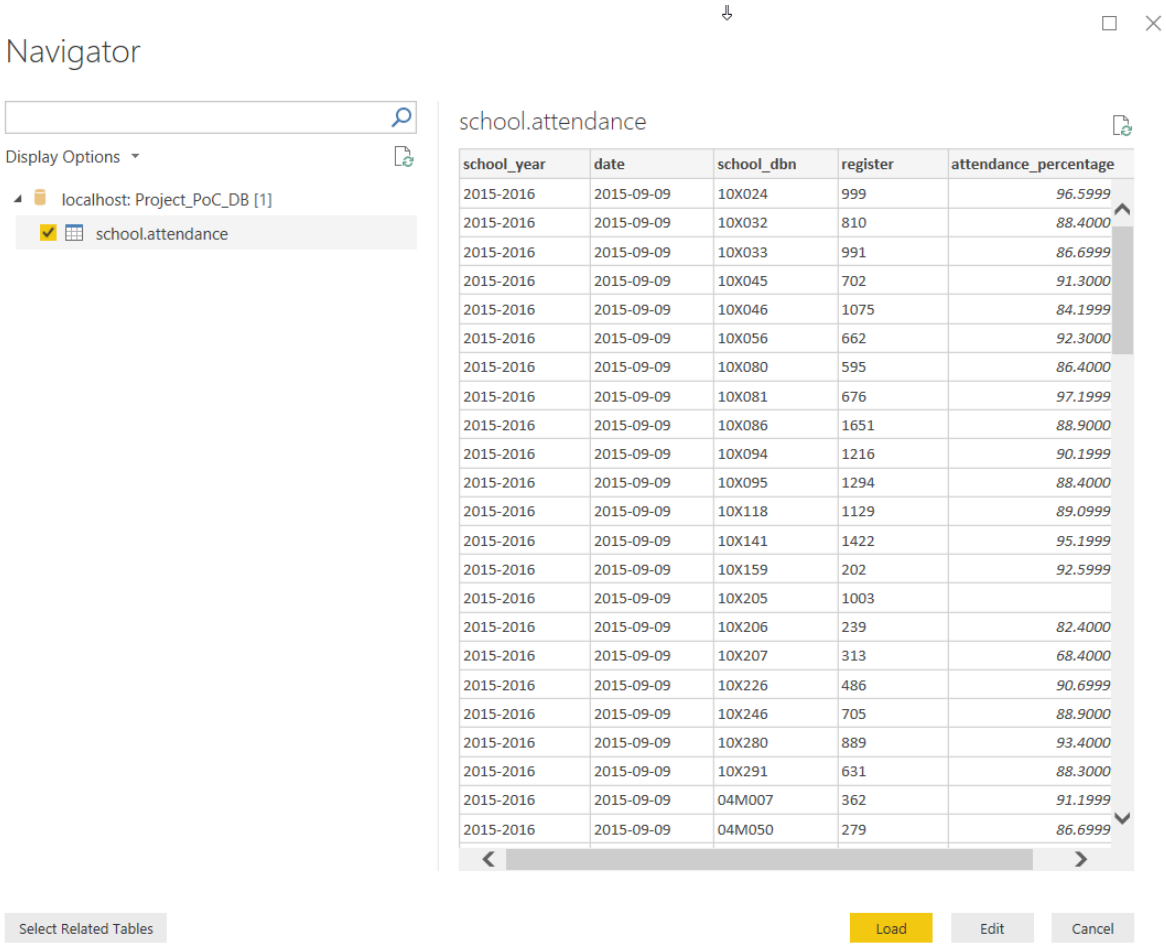
Vooraleer er iets gedaan kan worden in de BI-tool, moet er eerst data geïmporteerd worden. Door op de 'Get Data'-knop te klikken, krijgt de gebruiker volgend scherm te zien.



Figuur 17 Scherm voor dataprovider te selecteren

Hier wordt gekozen voor een Postgres-database. Vervolgens moet de gebruiker de inloggegevens van de Postgres-server invoeren.

Als er succesvol een verbinding is gemaakt met de server, krijgt de gebruiker het dataselectie-scherm te zien.

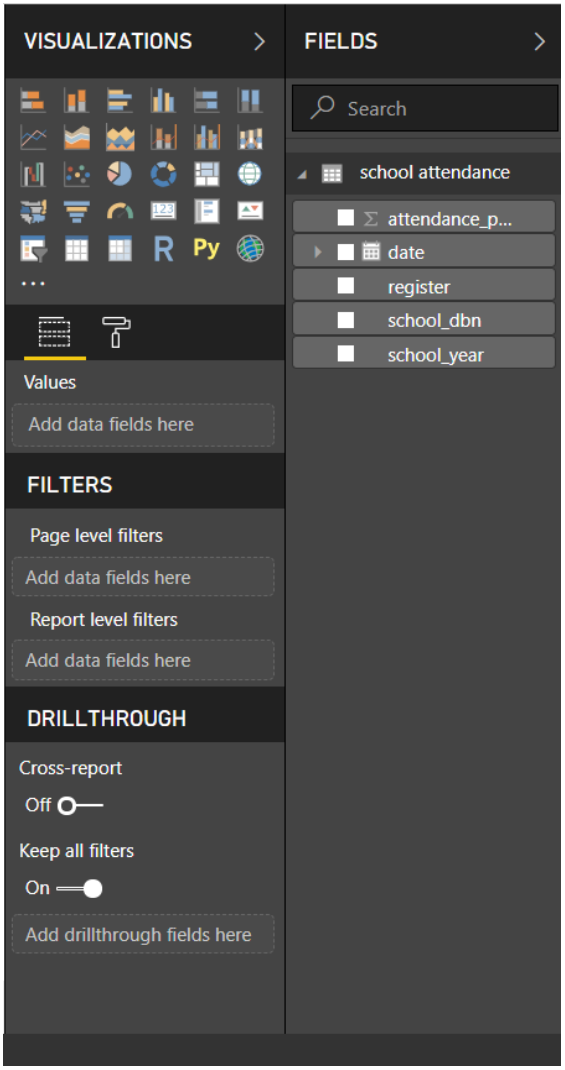


Figuur 18 Dataselectie in Power BI

Op dit scherm kan een gebruiker kiezen welke tabellen er geïmporteerd worden uit de database. Via de edit knop wordt Power Query geopend, waarmee bijvoorbeeld gekozen kan worden welke kolommen geïmporteerd moeten worden of bepaalde filters toegepast kunnen worden op de gegevens.

3.1.3.2 *Dataviews* aanmaken op een pagina

Nadat de tabellen geïmporteerd zijn, krijgt de gebruiker zijn pagina weer te zien. Rechts van de pagina zijn deze twee tabbladen te zien:



Figuur 19 Visualisaties en velden

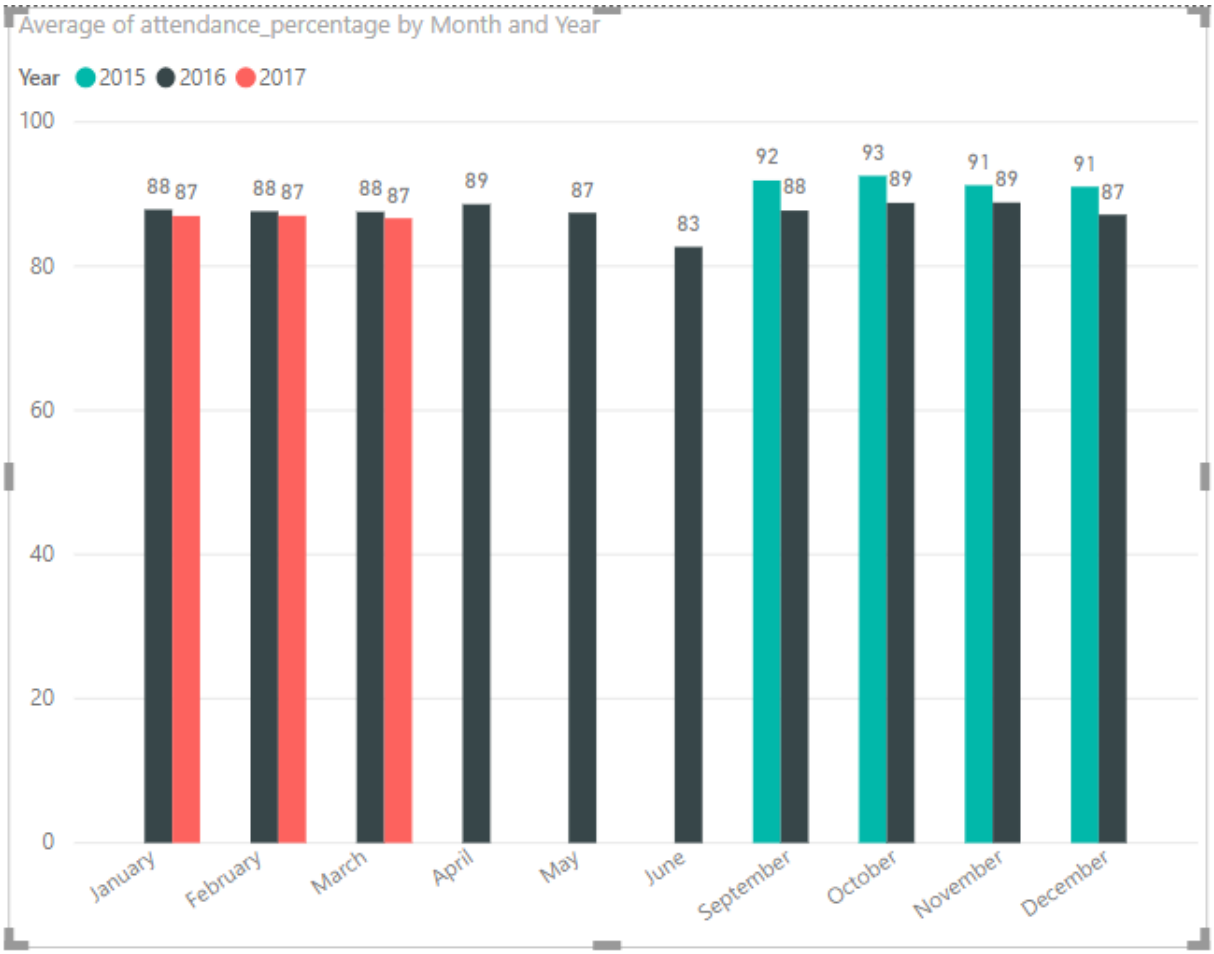
Onder visualisaties zijn de verschillende *dataviews* te zien die een gebruiker op een pagina kan zetten, met daaronder een tabblad waar filters ingesteld kunnen worden. Rechts ervan zijn de geïmporteerde tabellen en hun kolommen te zien.

Nu kan er een *dataview* aangemaakt worden. In deze POC wordt een staafdiagram aangemaakt, dat de gemiddelde schoolopkomst per maand laat zien.

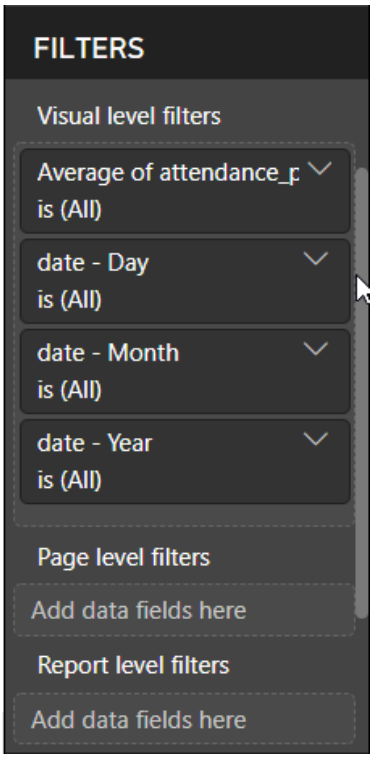
Door op een van de *dataviews* te klikken onder het visualisaties-tabblad, verschijnt deze automatisch op de pagina. Nu moeten de gegevens van de *dataview* nog aangevuld worden.

Een staafdiagram in Power BI heeft 3 velden die ingevuld moeten worden: een *axis*-veld, dat de x-as voorstelt, een *value*-veld, dat de y-as voorstelt. Als laatste is er nog een legende-veld, wat gebruikt kan worden om een legende bij de *dataview* te plaatsen. Op deze legende kan dan ook gefilterd worden. Voor het staafdiagram wordt op het axis-veld de datum-kolom geplaatst, op het *value*-veld het opkomstpercentage en op het legende-veld wordt de datum, gefilterd op jaar geplaatst.

Dit resultaat wordt dan bekomen:



Om nu filters toe te passen op deze *dataview*, moet er onder het filter-tabblad gekeken worden. Hier kunnen er op drie verschillende niveaus filters toegepast worden: op *dataview*niveau, op paginaniveau en op rapportniveau. Er worden automatisch al enkele filters gegenereerd, op *dataview*niveau:



Figuur 20 Toegepaste filters

Via deze filters kan er gefilterd worden op dag, maand en jaar.

Om het filteren nog makkelijker te maken, is het ook mogelijk om een *'slicer'* op de pagina te plaatsen. Deze ziet er zo uit:

Figuur 21 Slicer om data te filteren

Deze *slicer* krijgt een kolom toegewezen om op te filteren, de datum-kolom in dit geval. Er kan een exacte datum ingevoerd worden, of de twee bolletjes kunnen gebruikt worden om op een bepaalde periode te filteren.

3.1.4 Rapport delen met andere gebruikers

De gratis versie van Power BI Desktop heeft twee mogelijkheden om een rapport te delen: het rapport kan openbaar gemaakt worden. Hierbij wordt een URL aangemaakt die gedeeld kan worden met anderen. Via deze URL kan het rapport enkel bekeken worden, en niet aangepast.

De andere mogelijk is het rapport opslaan, als een PBIX-bestand, en het zo door te sturen naar de andere gebruikers.

Met Power BI Pro is er wel de mogelijkheid om het rapport met anderen te delen en hen het ook te laten bewerken. Hierbij wordt gebruik gemaakt van '*workspaces*'. In zo'n *workspace* kunnen gebruikers toegevoegd worden. Deze gebruikers kunnen hun rapporten publiceren naar de *workspace*, zodat de anderen er ook aan kunnen. [4]

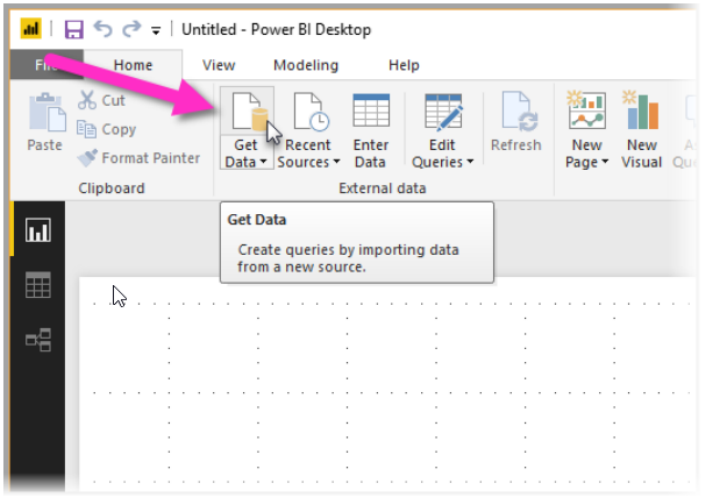
3.1.5 Documentatie van Power BI

Power BI heeft, zoals de meeste Microsoft producten, zeer goede documentatie ter beschikking. De documentatie is zeer duidelijk geschreven, en er worden zo goed als altijd schermafbeeldingen gebruikt om concepten nog duidelijker te maken. Hieronder is een pagina uit de Power BI documentatie te zien:

Connect to data

With **Power BI Desktop** you can connect to many different types of data. You can connect to basic data sources such as a Microsoft Excel file, and you can connect to online services that contain all sorts of data such as Salesforce, Microsoft Dynamics, Azure Blob Storage, and many more.

To connect to data, from the **Home** ribbon select **Get Data**.



The **Get Data** window appears, where you can choose from the many different data sources to which **Power BI Desktop** can connect. In this quickstart we use the Excel workbook that you downloaded, described in the *Prerequisites* section at the beginning of this article.

Figuur 22 Documentatie Power BI [5]

Ook belangrijk om te vermelden is dat Microsoft een 'Guided Learning'-cursus aanbiedt, waarin alle concepten binnen Power BI worden uitgelegd, aan de hand van documentatie en videos. Deze cursus gaat ook over de andere services binnen Power BI, en niet enkel over Power BI Desktop. [6]

3.2 Qlik Sense

Voor de POC van dit onderzoek wordt de gratis versie van Qlik Sense gebruikt: Qlik Sense Desktop.

3.2.1 Pricing

Ook Qlik maakt gebruik van het *SaaS*-model voor de producten die ze aanbieden. De prijs wordt berekend op maandbasis, maar wordt wel jaarlijks aangerekend, tegenover maandelijks. Qlik biedt twee gratis versies aan van Qlik Sense, en twee betaalde versies.

3.2.1.1 Qlik Sense Desktop

Het meest simpele pakket heet Qlik Sense Desktop, en is volledig gratis. Desktop neemt de vorm van een klassieke desktopapplicatie, en laat een gebruiker toe de meeste 'klassieke' BI-functies te gebruiken. Deze versie is eerder bedoeld voor individuele gebruikers, sinds apps niet gepubliceerd kunnen worden en alle gegevens van de gebruiker offline staan, wat wil zeggen dat de gebruiker zijn gegevens en apps zelf moet kopiëren als hij bijvoorbeeld een nieuwe computer wil gebruiken.

3.2.1.2 Qlik Sense Cloud

De andere gratis versie van Qlik Sense is de cloudversie. Deze versie biedt ongeveer dezelfde features aan als de Desktop versie van Qlik Sense, maar slaat alles op in de cloud, en kan enkel gebruikt worden via een webbrowser. Via de cloudversie kunnen apps die een gebruiker heeft aangemaakt gepubliceerd worden en gedeeld worden met andere gebruikers. De gratis versie van Cloud voorziet 250 MB opslagruimte per gebruiker, en laat apps tot 25 MB groot toe.

3.2.1.3 Qlik Sense Business

Er bestaat ook een betaalde versie van Qlik Sense Cloud, genaamd Qlik Sense Business. Deze versie is een 'premium' versie van de gewone cloudversie. Deze versie biedt voornamelijk meer opslagruimte aan (500 GB) en laat een gebruiker toe grotere apps (tot 250 MB) te publiceren, maar geeft een gebruiker ook toegang tot het volledige gamma aan dataproviders die beschikbaar zijn in Qlik Sense Cloud. [7] Qlik Sense Business kost $15 per maand, per gebruiker. Dit bedrag wordt aangerekend op jaarbasis. [8]

3.2.1.4 Qlik Sense Enterprise

Als laatste blijft er dan nog Qlik Sense Enterprise over, de meest uitgebreide en dure versie van Qlik Sense. Deze versie bestaat uit een server-client-systeem, waarbij een bedrijf de mogelijkheid heeft om een server te hosten, of te laten hosten door Qlik zelf. Met dit pakket heeft een bedrijf eigenlijk het volledige bestuur van het systeem in handen.

Met deze server kunnen alle Qlik Sense-apps die gemaakt zijn door de gebruikers gehost worden op een centrale server, zodat alle gebruikers (die toestemming hebben) deze apps kunnen gebruiken.

Qlik biedt voor deze versie geen algemene prijs aan. Een bedrijf moet Qlik zelf contacteren om een Enterprise pakket te laten samenstellen en op basis hier van een prijs schatting te maken.

Uit deze recensie [9] blijkt dat de Enterprise versie werkt met een token systeem, waarbij er een aantal tokens aangekocht worden. Deze tokens laten dan een gebruiker toe om in te loggen op de Enterprise Server. De schrijver van de recensie kreeg in zijn consultatie met Qlik een aanbod van 1500$ per token, met een jaarlijkse onderhoudskost van 3000$ voor de Enterprise Server. De tokens moeten wel maar eenmalig betaald worden.

3.2.2 Verbinden met data

Data importeren naar Qlik Sense Desktop kan op twee manieren: een databestand laden, of verbinden met een database aan de hand van een ODBC.

Qlik Sense ondersteunt Excel-bestanden en CSV-bestanden, maar ook HTML-bestanden en XML-bestanden. In het geval van HTML-bestanden ziet het programma elk '<table>' element als een tabel aan.

Naast deze bekende formaten, heeft Qlik Sense ook een aantal zelfgemaakte bestandsformaten. Deze zijn QVD-, QVX- en KML-bestanden. Deze formaten kunnen enkel gebruikt worden in Qlik Sense Desktop, maar zouden wel sneller werken dan alle andere beschikbare formaten.

Als alternatief voor een bestand in te laden, kan er ook een verbinding gemaakt worden met een database. Qlik Sense ondersteunt de volgende databases:

Tabel 5 Databases die Qlik Sense ondersteunt [10]

Amazon Redshift	Apache Drill
Apache Hive	Apache Phoenix
Apache Spark	Azure SQL
Cloudera Impala	Google BigQuery
IBM DB2	Microsoft SQL Server
MongoDB	MySQL Enterprise
Oracle	PostgreSQL
Presto	Sybase ASE
Teradata	

Qlik Sense Desktop heeft ook de mogelijkheid om data op te halen uit een REST-API.

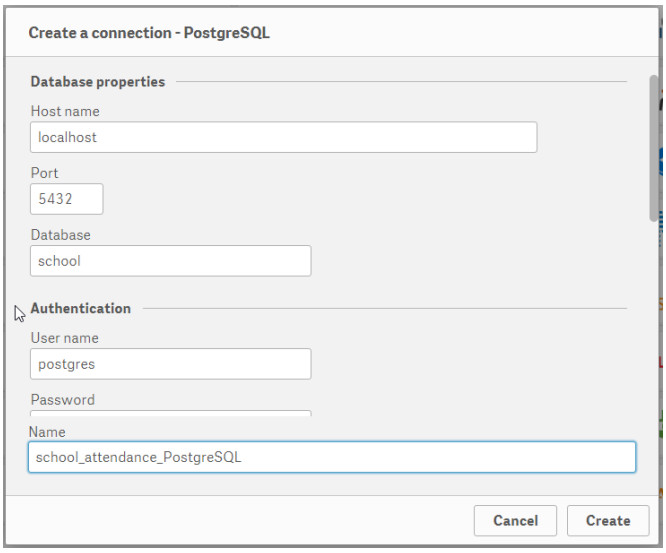
Qlik biedt ook een Software Development Kit aan voor zelf *database-connectors* te maken.

3.2.3 Een *dataview* aanmaken

Om te beginnen met het maken van een *dataview*, moet er eerst een app gemaakt worden. Een app binnen Qlik Sense is “een verzameling van herbruikbare data items, *sheets* en stories”. [11] Deze apps kunnen dan ook makkelijk gedeeld worden met anderen. Bij het opstarten van Qlik Sense Desktop moet een gebruiker dus eerst een nieuwe app aanmaken.

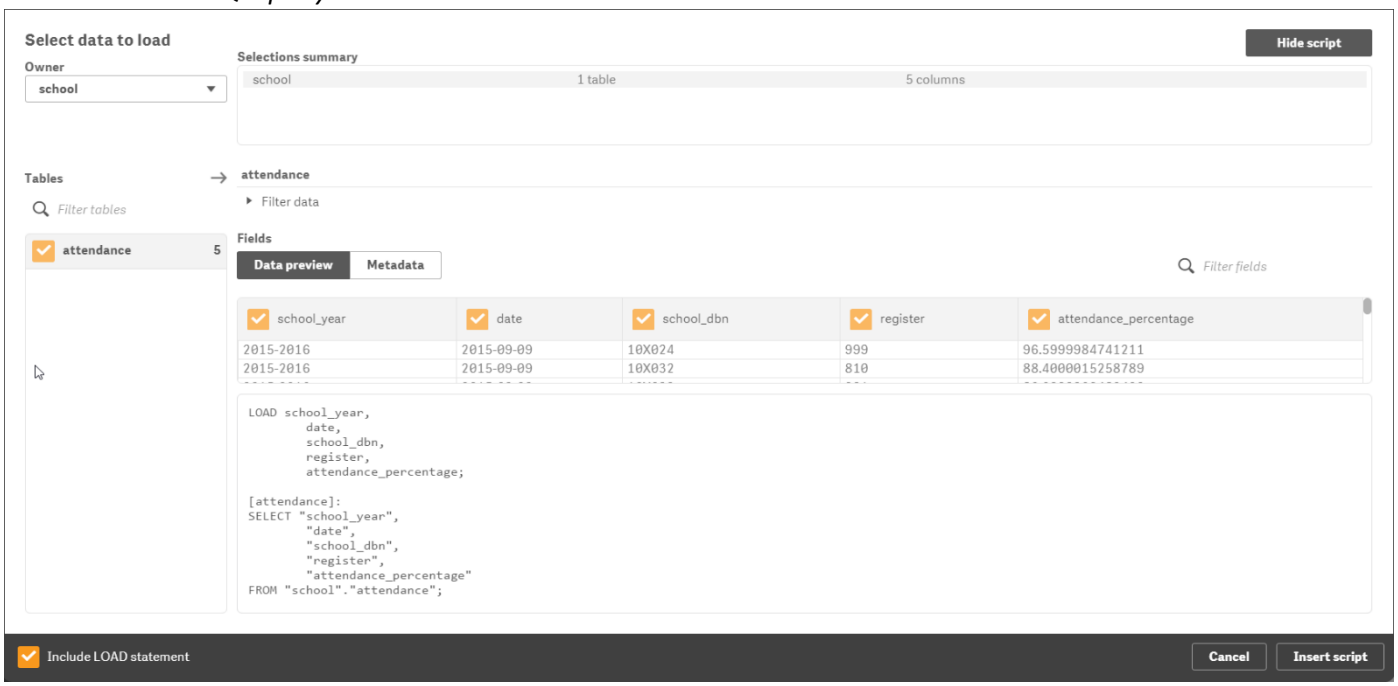
3.2.3.1 Data inladen

Nadat de app is aangemaakt, moet er een verbinding met de database gemaakt worden. Hier wordt gekozen voor een Postgres-database. Vervolgens moet de gebruiker de gegevens van de Postgres-server invullen.



Figuur 23 Verbinden met een Postgres-database

Als er succesvol verbinding is gemaakt met de database, krijgt de gebruiker een overzicht van de verschillende tabellen uit de database. Hier kan dan gekozen worden welke tabellen ingeladen moeten worden. Binnen de tabellen kan ook nog gekozen worden welke kolommen worden ingeladen. Ook kan er gekozen worden om een specifieke filter toe te passen, gelijkaardig aan een *'WHERE'-SQL-query*.



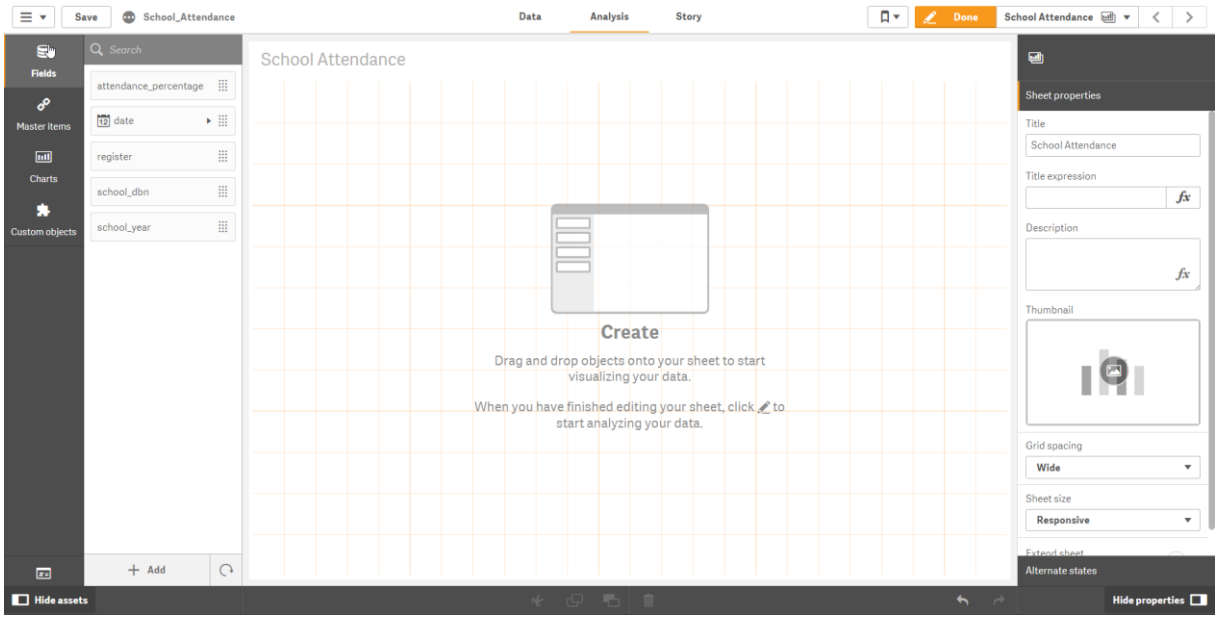
Figuur 24 Data selectie scherm

3.2.3.2 Aanmaken van *dataviews*

Na het selecteren van de data komt de gebruiker terecht op het analysescherm. Hier kan een gebruiker beginnen met het aanmaken van *dataviews*. Qlik Sense bied ook de *Insights*-optie. Deze optie gaat automatisch enkele *dataviews* aanmaken, op basis van de ingeladen data.

Om te beginnen met het aanmaken van *dataviews*, moet een gebruiker klikken op de *edit*-knop, waarna er een nieuwe *'sheet'* wordt aangemaakt. Een *sheet* is een verzameling van verschillende *dataviews*.

Aan de linkerkant kan een gebruiker views kiezen om in de *sheet* te plaatsen. Buiten tabellen, grafieken, etc. kunnen er ook gewoon velden uit de data geplaatst worden in de *sheet*, om bijvoorbeeld het totaal of gemiddelde van een kolom weer te geven.

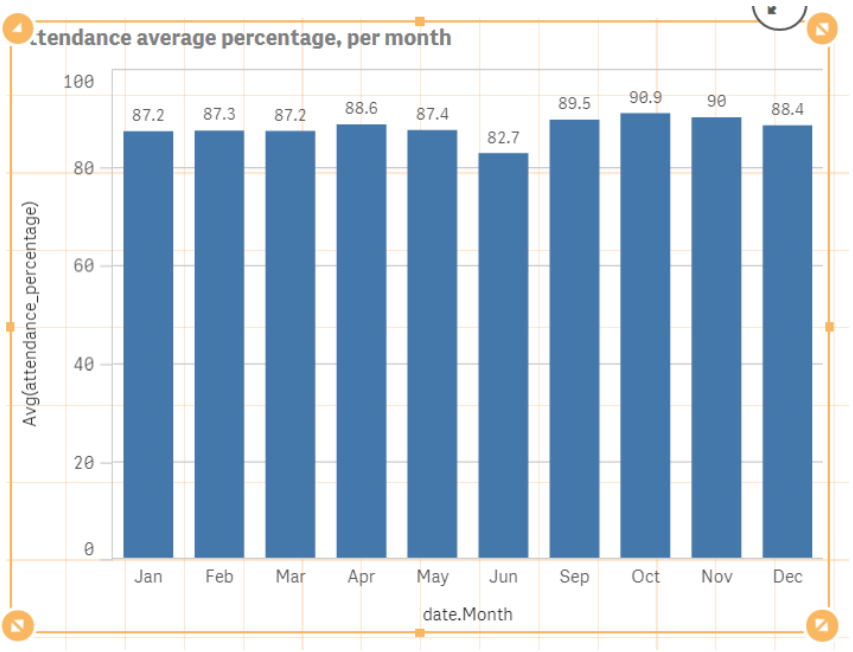


Figuur 25 Aanpassen van een sheet

Als er een nieuwe *dataview* aangemaakt wordt, moet er eerst een 'dimensie' en een 'maatstaf' gekozen worden. De dimensie en maatstaf stellen respectievelijk de x-as en y-as van de *dataview* voor.

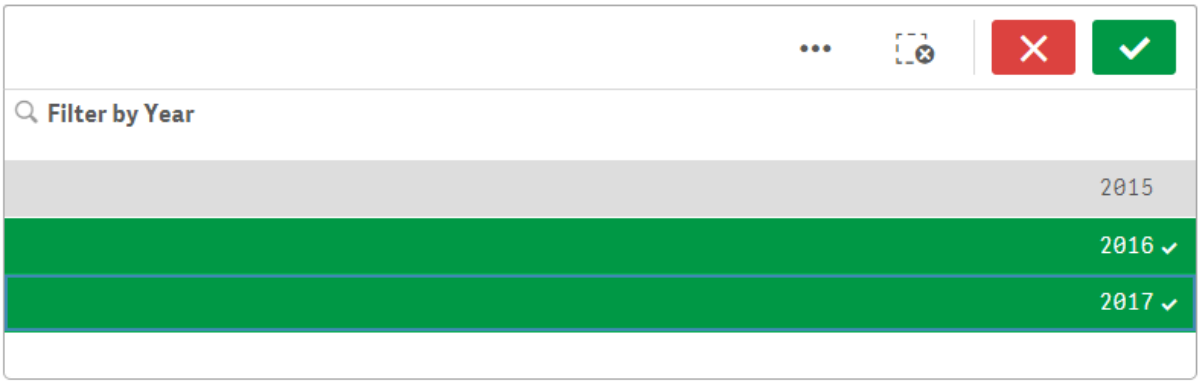
De waarden die hiervoor gekozen worden, kunnen ook gefilterd worden met functies. Zo kan er bijvoorbeeld de som van een kolom of het gemiddelde van een kolom gekozen worden als maatstaf.

In dit voorbeeld wordt er een staafdiagram gemaakt, zoals in onderstaande figuur te zien is, met het percentage van opkomst als maatstaf, en de datum kolom als dimensie. Sinds Qlik Sense automatisch detecteert dat de datumkolom datums bevat, geeft het automatisch ook opties om bijvoorbeeld het jaartal, maand, dag, kwartaal, etc. als dimensie te gebruiken.



Figuur 26 Dataview: staafdiagram

Nu moet er ook nog een datafilter aangemaakt worden, zodat de data in de *dataviews* gefilterd kan worden. Hiervoor wordt een '*filter pane'* in de *sheet* geplaatst. Deze filter heeft enkel een dimensie nodig. Een voorbeeld van een dimensie die gekozen kan worden is het jaartal van de datums, of het ID van een specifieke school, zodat de data in de *dataviews* specifiek voor één school wordt weergegeven. Hieronder is te zien hoe deze filter eruit ziet op de *sheet*.



Figuur 27 Jaarfilter

De filter die hier is aangemaakt gaat in elke *dataview* op de *sheet* waar de filter opstaat, die de datum-kolom als dimensie gebruikt, filteren op jaar, zodat enkel de datums van dat jaar in de *dataview* gebruikt worden.

3.2.4 App delen met andere gebruikers

Helaas biedt Qlik Sense Desktop geen directe, ingebouwde functie om de aangemaakte app met andere gebruikers te delen. De enige manier om een aangemaakte app te delen met andere is het bestand waar de app in opgeslagen zit, door te sturen naar andere gebruikers, die de app dan kunnen openen in Qlik Sense Desktop, en deze vervolgens kunnen bekijken en aanpassen.

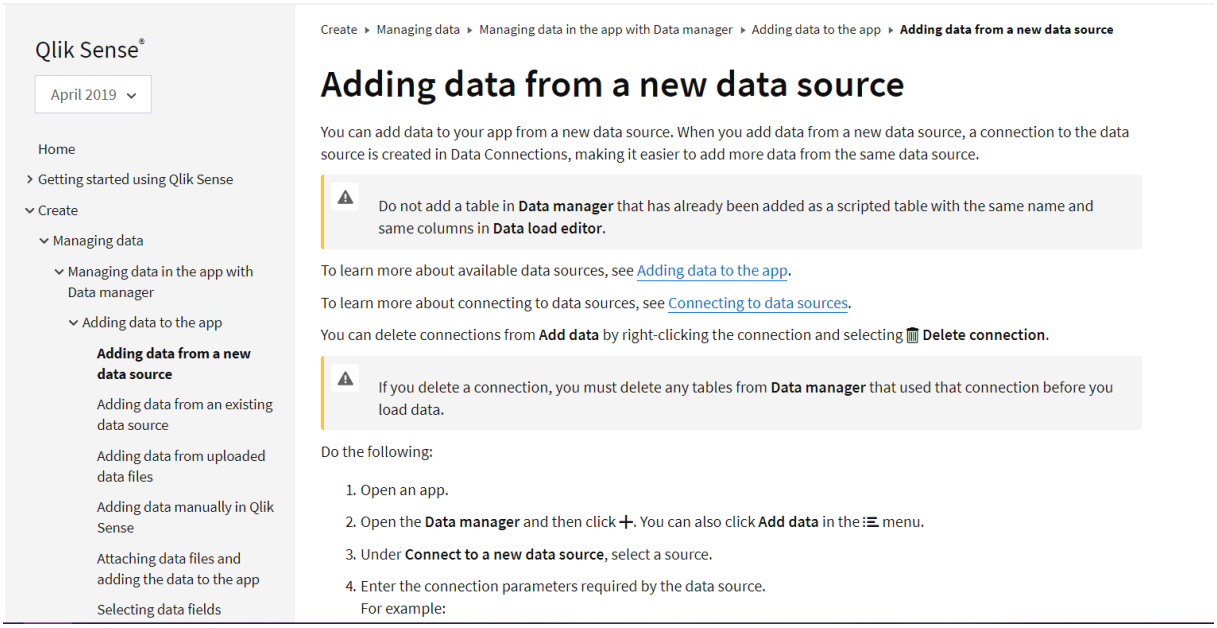
De apps worden standaard opgeslagen in een 'QlikSense'-map, in de documentenfolder van de gebruiker.

Qlik Sense Cloud, de cloud-gebaseerde variant van Qlik Sense Desktop, biedt wel enkele mogelijkheden aan om een app te delen: een gebruiker kan tot 5 anderen gebruikers uitnodigen via e-mail om de app te bekijken. Voor meer gebruikers is er Qlik Sense Business nodig. Deze gebruikers moeten dan enkel een Qlik Sense account hebben om via een webbrowser de app te bekijken.

3.2.5 Documentatie van Qlik Sense Desktop

Qlik biedt voor elk van zijn producten documentatie aan, die zeer duidelijk is: als iets uit tekst alleen niet duidelijk is, wordt er gebruik gemaakt van afbeeldingen.

Hieronder is een voorbeeld van Qlik Sense documentatie te zien:



Figuur 28 Documentatie van Qlik Sense [12]

Qlik heeft ook een Youtube-kanaal, waar er allerlei verschillende *videotutorials* geüpload worden.

3.3 Tableau

3.3.1 Pricing

Tableau biedt verschillende versies van hun product aan, en verdelen deze versies op basis van het aantal gebruikers die het product gaan gebruiken.

3.3.1.1 Tableau Creator

Het pakket dat Tableau aanbiedt voor individuele gebruikers is het Tableau *Creator* pakket. Dit pakket kost $70 per maand, aangerekend op jaarbasis. In dit pakket krijgt een gebruiker toegang tot Tableau Desktop, de primaire tool van Tableau waarmee een gebruiker de klassieke BI taken kan uitvoeren, Tableau Prep Builder, een tool om data te filteren en duidelijker te maken, zodat deze data dan in Tableau Desktop gebruikt kan worden, en één "*Creator*" licentie, waarmee een gebruiker toegang krijgt tot Tableau Online of Tableau Server. Dankzij deze 2 laatsten kan een gebruiker kiezen om de Tableau server on-premises te hosten (Tableau Server) of om de server te laten hosten door Tableau zelf (Tableau Online).

3.3.1.2 Tableau Team [13]

Voor meerdere gebruikers biedt Tableau een Team pakket aan, bestaande uit enkele verschillende tools, met als optie *on-premises* hosting te doen, aan de hand van Tableau Server, of om het pakket volledig te laten hosten door Tableau zelf, aan de hand van Tableau Online. De prijzen van de verschillende onderdelen van dit pakket verschillen op basis van de gekozen hosting optie.

Eerst en vooral krijgt een gebruiker één *Creator* pakket, met als bedoeling dat dit pakket gebruikt wordt om de initiële dataverwerking te doen en *dataviews* aan te maken.

Als tweede zitten er *Explorer* licenties bij het pakket. Met deze licentie kan een gebruiker op basis van vooraf aangemaakte data, *dataviews* maken en bekijken. Als de gebruiker dit pakket aankoopt, moeten er minstens vijf Explorer licenties gekocht worden. Een licentie kost $35 per maand met on-premises hosting, en $42 per maand met Tableau Online hosting, op jaarbasis aangerekend.

Als laatste bevat dit pakket Viewer licenties, die puur bedoeld zijn voor aangemaakt views te bekijken. Bij een Team pakket moeten er minstens 100 Viewer licenties gekocht worden. Deze licentie kost $12 per maand met on-premises hosting, en $15 per maand met Tableau Online hosting, ook weer op jaarbasis aangerekend.

3.3.2 Verbinden met data

In Tableau Desktop zijn er meerdere manieren om data te verkrijgen. Zoals Bij Qlik Sense Desktop zijn de manieren om te verbinden weer grotendeels te verdelen in twee categorieën: databestanden inladen, of verbinden met een database.

Tableau Desktop ondersteunt onderstaande dataproviders. Buiten degenen die hier opgesomd zijn, kunnen er ook zelf tools gemaakt worden om met nog andere providers te verbinden.

Tabel 6 Dataproviders die Tableau Desktop ondersteunt [14]

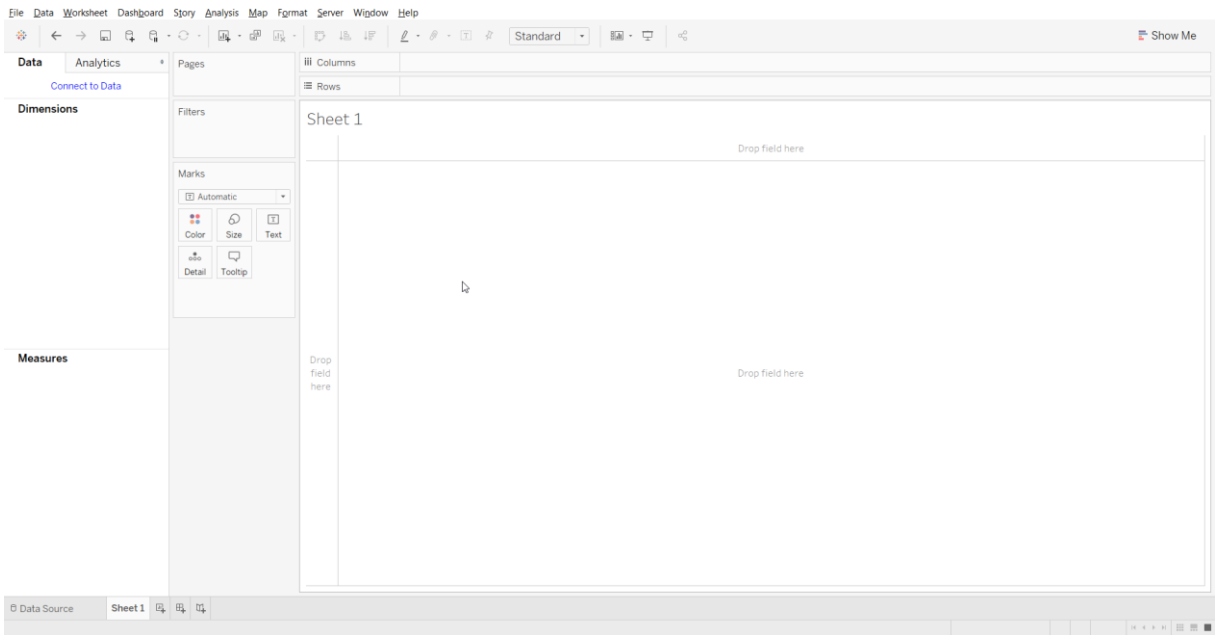
Actian Matrix	Actian Vector	Amazon Athena
Amazon Aurora	Amazon Elastic MapReduce	Amazon Redshift
Anaplan	Apache Drill	Aster Database
Box	Cloudera Hadoop Hive	Cloudera Impala
DataStax Enterprise	Denodo	Dropbox
Esri Shapefiles	Exasol	Firebird
GeoJSON	Google Ads	Google Analytics
Google BigQuery	Google Cloud SQL	Google *Sheets*
Hortonworks Hadoop Hive	HP Vertica	IBM BigInsights
IBM DB2	IBM PDA	JSON files
KML files	Kognitio	MapInfo Interchange Formats

MapInfo Tables	MapR Hadoop Hive	MariaDB
Marketo	MarkLogic	MemSQL
Microsoft Access	Microsoft Analysis Services	Microsoft Azure Data Warehouse
Microsoft Azure DB	Microsoft Excel	Microsoft OneDrive
Microsoft PowerPivot	Microsoft SharePoint Lists	Microsoft Spark on HDInisght
Microsoft SQL Server	Microsoft SQL Server PDW	MonetDB
MongoDB	MongoDB BI	MySQL
OData	Oracle	Oracle Eloqua
Oracle Essbase	PDF files	Pivotal Greenplum Database
PostgreSQL	Presto	Progress OpenEdge
Quickbooks Online	R files	Salesforce.com, including Force.com and Database.com
SAP BW	SAP HANA	SAP Sybase ASE
SAP Sybase IQ	SAS Files	ServiceNow ITSM
Snowflake	Spark SQL	Splunk
SPSS Files	Tableau Data Extract	Teradata
Teradata OLAP Connector	TIBCO® Data Virtualization	Text files—comma separated value (.csv) files

3.3.3 Een *dataview* aanmaken

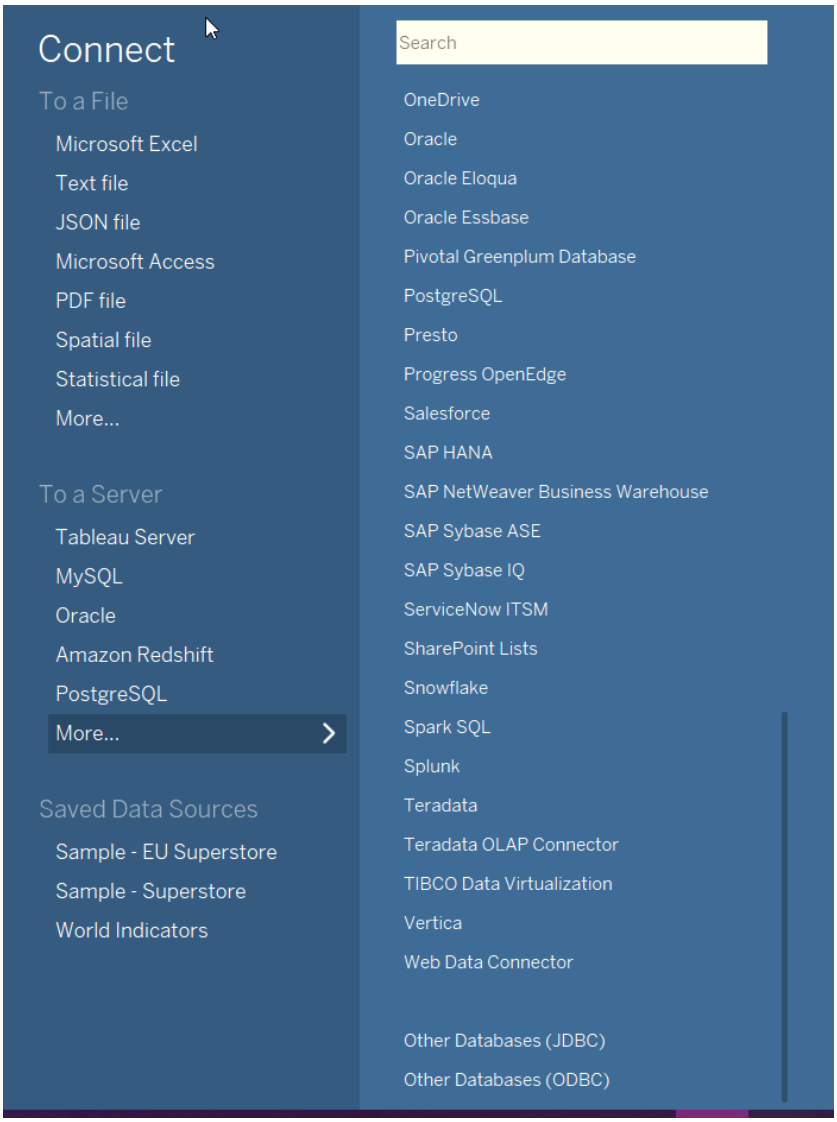
Tableau maakt gebruikt van '*Workbooks'*, die dezelfde functie hebben als de 'Apps' in Qlik Sense. Binnen deze *Workbooks* zitten dan meerdere *'sheets'*, die de *dataviews* bevatten. De *dataviews* die in deze *sheets* zitten, kunnen dan ook nog in een dashboard geplaatst worden, om zo een overzicht van de verschillende *dataviews* te creëren.

Eerst wordt er dus een nieuw *Workbook* aangemaakt. Hierbij wordt er ook automatisch een *sheet* aangemaakt, die een gebruiker dan te zien krijgt, zoals hieronder te zien.



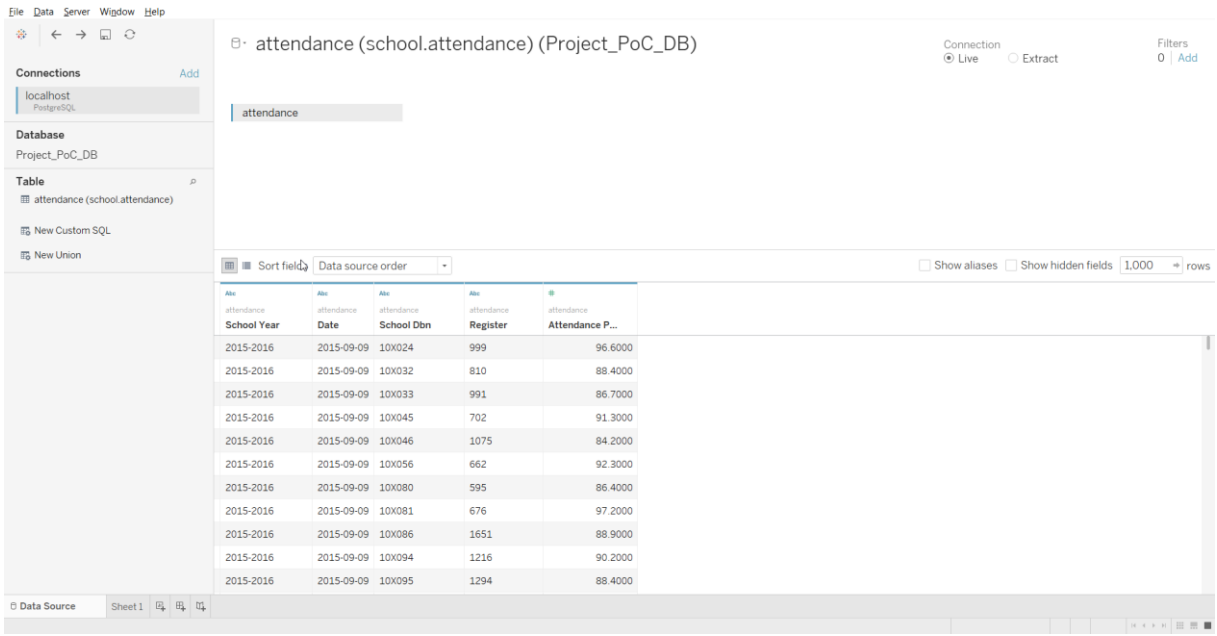
Figuur 29 Een sheet in Tableau Desktop

Vervolgens wordt er een verbinding gemaakt met de database te maken. Door te klikken op *'Connect To Data'*, krijgt de gebruiker een overzicht van de mogelijke databronnen waarmee verbonden kan worden, zoals hieronder te zien is.



Figuur 30 Overzicht van databronnen in Tableau Desktop

Er wordt verbinding gemaakt met de opgestelde Postgres-database. Na het invoeren van de gegevens van de Postgres-database server, krijgt de gebruiker een scherm te zien waarop gekozen kan worden welke tabellen en kolommen er precies ingeladen moet worden uit de Postgres-server. Hieronder is het dataselectie scherm te zien:

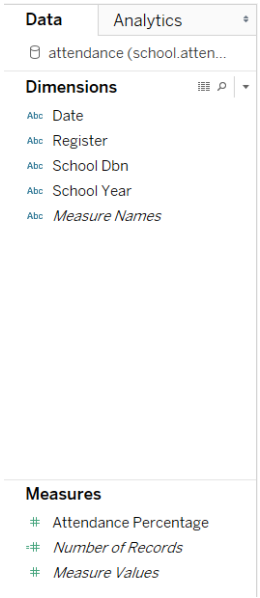


Figuur 31 Dataselectie in Tableau Desktop

Ook belangrijk is dat op deze pagina gekozen kan worden voor een *'live connection'* of een *'extract'* optie is. De eerste optie zorgt dat de data telkens live uit de database wordt gehaald, wat ervoor zorgt dat ge ze constant up-to-date blijft. De tweede optie haalt de dataset op uit de database, en slaat deze apart op, waardoor ze niet up-to-date blijft. Dit kan nuttig zijn als er bijvoorbeeld enkel data uit een specifieke periode opgehaald moet worden.

3.3.3.1 Aanmaken van *dataviews*

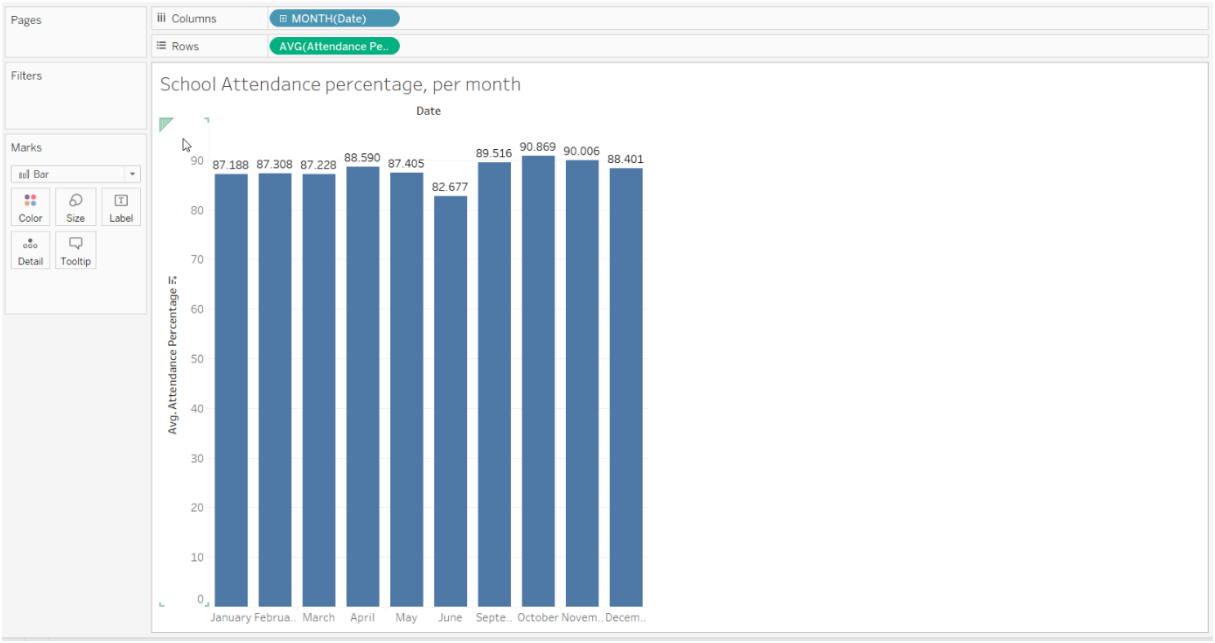
Na het inladen van de data, kan een gebruiker op de *sheet* beginnen met het aanmaken van *dataviews*. Op de *sheet* zijn nu aan de linkerkant de ingeladen tabellen te zien, samen met hun kolommen, zoals hieronder te zien is.



Figuur 32 Ingeladen tabellen en kolommen

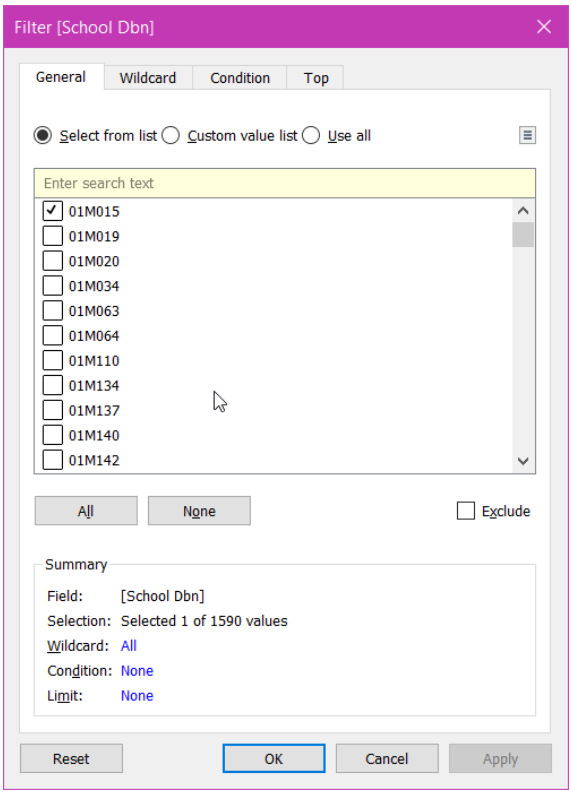
Tableau Desktop gebruikt een iets anders systeem voor het aanmaken van *dataviews*. Bovenaan de *sheet* kunnen kolommen en rijen ingesteld worden, die respectievelijk de x-as en de y-as van een *dataview* voorstellen.

Eens dat er kolom en rij is ingesteld, wordt er automatisch een *dataview* gegenereerd, op basis van de gekozen kolom(men) en rij(en). Links van de *sheet*, onder het tabblad '*marks*', kan de *dataview* aangepast worden: welk soort view het moet zijn (grafiek, staafdiagram, cirkeldiagram, etc.), de kleuren, of er labels getoond worden, etc. Als datum als kolom wordt ingesteld, en het opkomstpercentage als rij, wordt er deze *dataview* gegenereerd:



Figuur 33 Aangemaakte dataview

Om nu deze gegevens te filteren, kan er onder het 'filters' tabblad een van de kolommen uit de tabel geplaatst worden. In dit voorbeeld wordt er gefilterd op basis van het school-ID. Een gebruiker krijgt dit te zien wanneer hij een filter toepast.



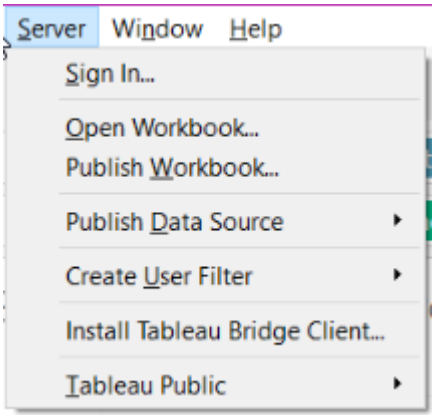
Figuur 34 Filteren op basis van school-ID

Er kan gekozen worden om op één of meerdere ID's te filteren. Als de gebruiker op OK klikt, wordt de filter automatisch toegepast.

3.3.4 *Workbook* delen met andere gebruikers

Binnen het Tableau-ecosysteem zijn kan een gebruiker *workbooks* delen via Tableau Online, Tableau Server of Tableau Public.

Bovenaan, onder de tab '*Server*' staan de opties voor te delen:



Figuur 35 Opties voor workbook te delen

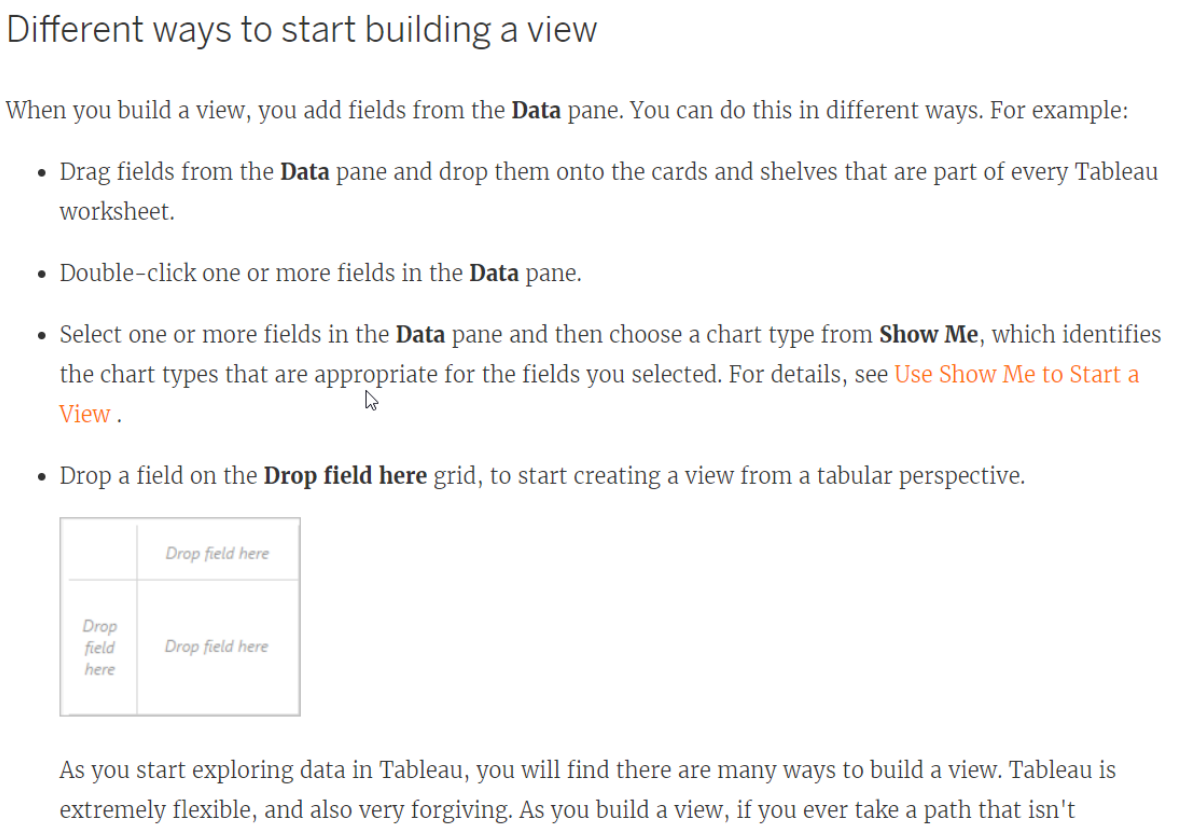
Als een gebruiker zijn/haar *workbook* deelt naar Tableau Server of Tableau Online, kunnen de andere gebruikers die in dezelfde organisatie zitten het *workbook* bekijken. Ook kunnen er permissies opgesteld worden op het *workbook*, zodat het bijvoorbeeld ook aangepast kan worden door bepaalde gebruikers. [15]

De *workbook* kan ook Tableau Public gepubliceerd worden. Hierbij is de *workbook* wel zichtbaar voor alle andere gebruikers van Tableau Public. Deze optie is dan ook niet echt bedoeld voor het delen van *workbooks* met mogelijk confidentiële informatie.

3.3.5 Documentatie van Tableau Desktop

Tableau biedt, net zoals de andere BI-tools, zeer uitgebreide en duidelijke documentatie aan. In de Tableau Desktop documentatie is de informatie altijd gepresenteerd in de context van een iets of wat 'realistische' situatie ('Je hebt verkoopcijfers opgehaald van de laatste 4 jaar, en wil nu X en Y maken voor je baas.'), waardoor lezers van de documentatie meteen een idee krijgen waarvoor ze Tableau Desktop kunnen gebruiken.

Hier is een stuk uit de Tableau documentatie te zien:



Figuur 36 Voorbeeld van Tableau Documentatie

Hoewel er in de algemene documentatie geen video's terug te vinden zijn, is er wel een pagina met '*training videos*', die bepaalde concepten verduidelijken of uitleggen. [16]

4 Conclusie van onderzoek

Hier worden de resultaten van het onderzoek op een rijtje gezet, om te kijken welke tool het beste scoort op welk criterium.

4.1 Gebruiksvriendelijkheid

Gebruiksvriendelijkheid is in dit onderzoek gemeten door middel van het maken van een POC in de drie verschillende BI-tools die onderzocht zijn, en te kijken hoe makkelijk het is om een *dataview* met een filter aan te maken. Toch blijft het moeilijk om een objectief oordeel te vellen over gebruiksvriendelijkheid.

Power BI Desktop en Qlik Sense Desktop hebben een zeer gelijkende gebruikersinterface, met in het midden van het scherm de dataviews en aan de rechterkant de eigenschappen van de dataviews. De views zelfs kunnen gewoon uit een lijst gesleept worden en zo neergezet worden.

Tableau Desktop daarentegen gebruikt een toch wel iets andere interface, met de kolommen en rijen die bovenaan ingesteld moeten worden, om er zo dataviews van aan te maken. Dit maakt het iets moeilijker om gewoon '*drag-and-drop'* te werken.

Qlik Sense Desktop en Power BI Desktop laten ook toe om op een dataview zelf een filter te plaatsen, wat in Tableau Desktop niet mogelijk is.

Voor een eenvoudige 'drag-and-drop' ervaring, zijn Qlik Sense Desktop en Power BI Desktop het beste.

4.2 Deelbaarheid van aangemaakte *dataviews* met andere gebruikers

Tabel 7 Vergelijkingstabel deelbaarheid

Power BI Desktop	**Tableau Desktop**	**Qlik Sense Desktop**
• Aangemaakte rapporten kunnen openbaar gemaakt worden, waardoor ze door iedereen gezien kunnen worden. • Rapport kan opgeslagen worden als PIBX bestand en doorgestuurd worden naar andere gebruikers. • Pro versie van Power BI Desktop laat gebruikers toe om workspaces aan te maken, waar rapporten gedeeld kunnen worden met anderen.	• *Workbooks* kunnen gedeeld worden via Tableau Server of Tableau Online. Zo krijgen andere uitgenodigd worden om de *workbooks* te bekijken of aan te passen. • Een *workbook* kan ook gewoon opgeslagen worden als een bestand, dat doorgestuurd kan worden naar andere gebruikers.	• Biedt geen mogelijkheden om te delen met andere gebruikers, buiten een aangemaakte app op te slaan als QVF bestand en door te sturen naar andere gebruikers. • Gratis cloudversie van Qlik Sense Desktop laat gebruikers toe om apps met maximum 5 gebruikers te delen. • Betaalde versie van cloud laat toe om met een ongelimiteerd aantal gebruikers te delen.

Als er gekeken wordt naar de versies die in de POC gebruikt zijn, is Qlik Sense Desktop de enige die geen ingebouwde manier heeft om te delen met andere gebruikers. Daar tegenover staat dan wel dat de cloudversie van Qlik Sense Desktop de enige gratis tool is die gebruikers toelaat om met elkaar te delen.

De gratis versie van Microsoft Power BI Desktop biedt ook geen manieren aan om rapporten met andere gebruikers te delen. Hiervoor is de Pro versie nodig.

Tableau Desktop biedt wel enkele manieren om te delen, aan de hand van Tableau Online/Server, maar van Tableau Desktop is natuurlijk geen gratis versie beschikbaar.

Als er een tool gekozen moet worden, die zou weinig mogelijk kost, maar wel toelaat om te delen met andere gebruikers-, is de cloudversie van Qlik Sense Desktop de winnaar: deze is gratis en laat gebruikers toe hun apps te delen met andere gebruikers.

4.3 Documentatie van de BI-tools

Over het algemeen bieden alle drie de tools goede, duidelijke documentatie aan. Elke tool zijn documentatie bevat zowel video's als afbeeldingen om concepten duidelijker te maken. Daarom wordt er vooral gekeken welke extra's er worden aangeboden in de documentatie om alles duidelijker te maken.

Tabel 8 Vergelijking documentatie

Power BI Desktop	**Qlik Sense Desktop**	**Tableau Desktop**
• Gratis 'Guided Learning'-cursus, om alle onderdelen van Power BI te leren kennen. Deze cursus gebruikt zowel tekstdocumentatie als video's bij elke hoofdstuk van de cursus.	• Heeft een YouTube-kanaal, waar er verschillende video's beschikbaar zijn over verschillende Qlik producten.	• Biedt gratis training video's aan. Deze video's zijn opgedeeld per product dat Tableau Software aanbiedt. Onderaan elke video staan er nog enkele links naar relevante documentatie omtrent dat hoofdstuk.

In deze categorie zijn Power BI Desktop en Tableau Desktop de winnaars. De training video's van Tableau en de cursus van Power BI zijn beiden zeer duidelijk en professioneel uitgewerkt. Beiden volgen een systeem waarbij er wordt verder gewerkt op de vorige video, zodat er een verhaal wordt gecreëerd.

4.4 Prijsklassen van de BI-tools

Tabel 9 Prijsvergelijking van de BI-tools

Power BI	**Qlik Sense**	**Tableau**
• Power BI Desktop: **gratis** versie van Power BI. • Power BI Pro: premium versie van Power BI Desktop. Geeft onder andere mogelijkheden tot het delen van rapporten met anderen. **Kost 9,99$ per maand.** • Power BI Premium: serveroplossing van Power BI. Servers worden op 'nodes' gehost, en kosten worden berekend per node. **Kost 4999$ per maand, per maand.**	• Qlik Sense Desktop: is volledig **gratis** te gebruiken. • Qlik Sense Cloud: een cloudversie van Qlik Sense Desktop. Is ook gratis te gebruiken. • Qlik Sense Business: een premium versie van Qlik Sense Cloud. **Kost 15$ per maand**. • Qlik Sense Enterprise: serveroplossing waarbij alle apps op een server gehost worden. Prijs staat niet vast, en moet in overleg met Qlik opgemaakt worden.	• Tableau Creator: **kost 70$ per maand**. Bevat Tableau Desktop, samen met een Creator-licentie voor Tableau Online/Server. • Tableau Team: teamoplossing van Tableau. Dit pakket bevat meerdere licenties, met verschillende prijzen op basis van de gekozen hosting optie: Tableau Server of Tableau Online. Refereer naar het hoofdstuk 'Tableau Team' voor een volledige uitleg van de kostberekening van dit pakket.

Als er naar de Desktop oplossingen gekeken wordt, is Power BI de winnaar van deze categorie: ze bieden een gratis versie van hun Desktop product aan, en de premium versie ervan is goedkoper dan die van Tableau en Qlik.

Op vlak van serveroplossingen is er geen duidelijke winnaar. De kosten hier hangen af van hoe groot het de serveroplossing moet zijn. Bij Power BI Premium liggen de kosten voor een server vast, maar bij de Qlik Sense Enterprise wordt er een prijs gemaakt op basis van de noden die een gebruiker heeft. Bij Tableau Team liggen de kosten per licentie van het pakket ook vast, maar dan moet een gebruiker nog altijd beslissen hoeveel licenties er precies nodig gaan zijn.

4.5 Mogelijkheden om met data te verbinden

Tabel 10 Vergelijking dataproviders

Power BI Desktop	Qlik Sense Desktop	Tableau Desktop
• 30 verschillende databases. • 12 verschillende Azure dataproviders. • 37 verschillende web services. • 22 'overige' dataproviders (bijvoorbeeld Python Script) • Datasets kunnen ook ingeladen worden uit bestanden, zoals Excel-*sheets*, CSV-bestanden, JSON, etc. • Datasets kunnen ook worden opgeslagen in een Power BI-formaat, en zo gebruikt worden.	• 17 verschillende databases. • Datasets kunnen ook ingeladen worden uit bestanden, zoals Excel-*sheets*, CSV-bestanden, JSON, etc. • Datasets kunnen in een speciaal Qlik-formaat worden opgeslagen, waardoor ze sneller ingeladen kunnen worden door Qlik Sense Desktop. • Er kan kunnen ook datasets uit een REST-API gehaald worden. • Qlik biedt een SDK aan voor het maken van database-connectors.	• 79 verschillende databases en services. • Datasets kunnen ook ingeladen worden uit bestanden, zoals Excel-*sheets*, CSV-bestanden, JSON, etc. • Tableau Desktop biedt gelimiteerde ondersteuning voor zelfgemaakte database- en web-connectors.

Als er puur gekeken wordt naar out-of-the-box ondersteuning, biedt Power BI de meeste mogelijkheden om met dataproviders te verbinden. Het aanbod van Power BI is gigantisch, en zorgt dus dat een gebruiker direct aan de slag kan, zonder zich echt zorgen te moeten maken of de gebruiker zijn dataprovider ondersteund is.

Op de tweeds plaats staat Tableau Desktop. Deze biedt iets minder out-of-the-box mogelijkheden aan, maar compenseert hiervoor met de ingebouwde mogelijkheid om zelf database- en web-connectors aan te maken.

Qlik Sense Desktop heeft een beduidend minder aantal ondersteunde dataproviders. Daarin tegen is Qlik Sense Desktop wel de enige die een ondersteuning biedt om te verbinden met een REST-API om data op te halen.

5 Reflectie onderzoek

Het onderzoek is voor mij een beetje stroever verlopen dan ik had gewild.

Ik heb tijdens de laatste weken van de stage niet veel aan het onderzoek kunnen werken, omdat ik alle tijd nodig had om mijn stageopdracht af te werken. Hierdoor heb ik na de laatste week van de stage nog veel werk gehad aan mijn onderzoek.

Een andere negatief puntje van mezelf dat me heeft tegengewerkt is mijn uitstelgedrag, waardoor ik met sommige stukken van het onderzoek later ben begonnen dan ik had gewild, wat me toch ook wel de nodig stress heeft opgebracht.

Ook heb ik moeite gehad met het definiëren van een duidelijke onderzoeksvraag met duidelijke deelvragen, omdat ik het niet zo makkelijk vond goede criteria te vinden om de tools op de beoordelen.

Buiten deze negatieve puntjes is het onderzoek wel interessant geweest voor te maken. Ik had enkel van Business Intelligence gehoord tijdens een seminarie tijdens mijn tweede schooljaar aan de PXL, en heb het sindsdien altijd wel een interessant onderwerp gevonden. Hoewel BI misschien niet direct iets is waar ontwikkelaars altijd mee in aanraking gaan komen, biedt het toch wel leuke mogelijkheden aan ontwikkelaars voor hun data in te kijken.

Bibliografie

[1] C. S. E. B. Margret Rouse, „Definition business intelligence (BI),” TechTarget, Februari 2019. [Online]. Available: https://searchbusinessanalytics.techtarget.com/definition/business-intelligence-BI. [Geopend 1 April 2019].

[2] „Pricing & Product Comparison,” Microsoft, [Online]. Available: https://powerbi.microsoft.com/en-us/pricing/. [Geopend 01 April 2019].

[3] „Data sources in Power BI Desktop,” Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/power-bi/desktop-data-sources. [Geopend 07 Juni 2019].

[4] „Create classic workspaces in Power BI,” Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/power-bi/service-create-workspaces. [Geopend 09 Juni 2019].

[5] „Quickstart: Connect to Data,” Microsoft, 08 05 2019. [Online]. Available: https://docs.microsoft.com/en-us/power-bi/desktop-quickstart-connect-to-data. [Geopend 08 Juni 2019].

[6] „Microsoft Power BI Guided Learning,” Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/power-bi/guided-learning/. [Geopend 06 Juni 2019].

[7] „Upgrading your cloud subscription,” Qlik, [Online]. Available: https://help.qlik.com/en-US/sense-cloud/Subsystems/CloudHub/Content/Sense_Hub/Cloud/upgrade-cloud.htm. [Geopend 01 April 2019].

[8] „Data Visualizations In The Cloud,” Qlik, [Online]. Available: https://www.qlik.com/us/products/qlik-sense/qlik-sense-cloud. [Geopend 01 April 2019].

[9] O. Rist, „Qlik Sense Enterprise Server,” PCMag, 30 Juni 2016. [Online]. Available: https://www.pcmag.com/review/338085/qlik-sense-enterprise-server. [Geopend 08 Juni 2019].

[10] „ODBC Connector Package - Qlik Connectors,” QlikTech International AB, [Online]. Available: https://help.qlik.com/en-US/connectors/Subsystems/ODBC_connector_help/Content/Connectors_ODBC/Introduction/ODBC-connector.htm. [Geopend 27 Juni 2019].

[11] „Creating Apps,” QlikTech International, April 2019. [Online]. Available: https://help.qlik.com/en-US/sense/April2019/Subsystems/Hub/Content/Sense_Hub/Visualizations/create-apps-visualizations.htm. [Geopend 14 Mei 2019].

[12] „Adding data from a new data source,” Qlik, [Online]. Available: https://help.qlik.com/en-US/sense/April2019/Subsystems/Hub/Content/Sense_Hub/LoadData/adding-data-new-data-source.htm. [Geopend 05 06 2019].

[13] „Tableau Pricing For Teams & Organizations," Tableau Software, [Online]. Available: https://www.tableau.com/pricing/teams-orgs. [Geopend 26 April 2019].

[14] „Tableau Desktop," Tableau Software, [Online]. Available: https://www.tableau.com/products/desktop#data-sources. [Geopend 08 Juni 2019].

[15] „Publish Data Sources and Workbooks," Tableau Software, [Online]. Available: https://onlinehelp.tableau.com/current/pro/desktop/en-us/publish_overview.htm. [Geopend 08 Juni 2019].

[16] „Tableau Training: View Training Courses," Tableau Software, [Online]. Available: https://www.tableau.com/learn/training. [Geopend 08 Juni 2019].

[17] „2019 Gartner Magic Quadrant," Microsoft, [Online]. Available: https://info.microsoft.com/ww-landing-gartner-mq-bi-analytics-2019.html. [Geopend 09 06 2019].

[18] Anna, „Microsoft Power BI Review: Is It Right Visualization Tool For You? Our 10 Point Scorecard," Openbridge, 15 November 2017. [Online]. Available: https://blog.openbridge.com/is-microsoft-power-bi-right-for-you-10-point-checklist-to-make-the-right-decision-c99eb9673405. [Geopend 05 April 2019].