# Lessons learned (obv github)

1. **API beschikbaarheid en API bruikbaarheid**

Aanvankelijk was de gedachten bij de start van het project, dat een ruime beschikbaarheid van API’s zou zorgen voor een brede beschikbaarheid van data. Het juist combineren van deze data zou een schat aan informatie opleveren.

Echter, in tegenstelling blijkt dat API’s niet ruim beschikbaar zijn. De BAG beschikt over een goed opgezette API die zowel relationele als geografische data biedt. De BAG API onderscheidt zich daar dan ook mee.

Meer dan 80% van de gebruikte data wordt beschikbaar gesteld als kaart laag, die zich niet als API laat gebruiken. Daarbij is de BAG API ook de enige API die zich geografisch laat bevragen.

Daarnaast stelt een API niet per definitie alle data beschikbaar. De BAG API stelt bijvoorbeeld het gebruiksdoel van een verblijfsobject niet beschikbaar via de API.

1. **API als datasilo**

Eén constatering ten aanzien van API’s lijkt te zijn dat een API gezien kan worden als data silo. De API deelt de data en bevat veelal geen enkele relatie naar data (of een dataset) buiten de API. Gebruikelijk lijkt te zijn, dat API’s die meerdere requests aan bieden (een url om bepaalde data aan te bieden). Naar elkaar verwijzen. Zo verwijst de data over een verblijfsobject naar de API request om pand data op te vragen.

Om data uit de datasilo te halen, zou er verwezen moeten worden naar identifiers. Hierdoor kan er generiek om gegaan worden met het op het ophalen van data van verschillende API’s.

Daarnaast zouden API’s kunnen verwijzen naar identifiers van andere API’s. Zo beschrijft de data van de Rijksdienst voor Cultureel Erfgoed (RCE) adressen op monumenten, maar deze laten zich lastig vergelijken en brengt grove foutmarges met zich mee. Idealiter zou de RCE API (ook) verwijzen naar een adres identifier beschikbaar binnen het BAG.

1. **Eén en tweezijdige verwijzingen en afwijkingen**Datamodellen leggen restricties op. Zo legt bijvoorbeeld het BAG datamodel vast dat een verblijfsobject naar een pand verwijst, maar een pand niet een verblijfsobject. Echter de API doet dit wel en wijkt daarmee af van het datamodel. Dit maakt het gebruik van een API erg specifiek.
2. **Wisselende API resultaten**

Een API kan wisselende resultaten terug geven. Een veelvoorkomende situatie is het teruggeven van één of meerdere resultaten. Dit houdt bijvoorbeeld als het resultaat één instantie bevat wordt deze direct terug gegeven in de resultaten. Echter als het resultaat meerdere instanties teruggeeft, worden deze gebundeld in bijvoorbeeld een JSON array. Normaliter is dit niet persé een probleem. In dit project vormt dit wel een probleem. De resultaten van een API worden omgezet naar een semantisch formaat. Dit wordt gedaan door middel van een mapping. Hoe de instanties gebundeld worden moet in de mapping verwerkt worden, anders kan deze niet goed uitgevoerd worden.

1. **Adres gegevens onvergelijkbaar**

Adres gegevens blijken moeilijke gegevens te zijn. In de verschillende datasets worden adres gegevens gebruikt, echter laten deze zich moeilijk vergelijken. Aannemende dat het Kadaster het meest complexe model op adres gegevens hanteert lijkt dit ook de meest nauwkeurige en daardoor meest bruikbare. Een adres bestaat uit een aantal facetten (plaats, straat, huisnummer, huisnummer toevoeging, huisnummer letter). Echter een groot deel van de datasets/API’s hanteren deze data door elkaar. Hierdoor is het lastig om adres data te vergelijken.

1. **Stelselcatalogus geen relatie met bron**

Om de semantische relatie tussen de data van de verschillende API’s te ontdekken, moet in de semantische orchestratie laag gedefinieerd worden hoe data zich tot elkaar verhoudt. Dit wordt prima gedaan door de al reeds bestaande stelselcatalogus. Deze catalogus bevat enkel de basisregistratie, maar dezelfde structuur wordt toegepast op de registratie die hier niet in zitten.

Een deel van de data is ook al beschikbaar als semantische data bij de bron. Echter is er geen relatie tussen de semantische relaties bij de bron en de equivalente relaties in de stelselcatalogus. Hierdoor is het onmogelijk om orchestratie laag te gebruiken om de al reeds semantische data op te vragen en aan de demonstrator toe te voegen.

1. **Wat als dat van meerdere bronnen komt?**

Tijdens het ontwikkelen van de Disgeo demonstrator is de aanname gedaan dat een data geleverd wordt door één enkele API. Data omtrent een verblijfsobject zal altijd van het kadaster komen. Op het moment data deze aanname ongeldig wordt, treedt het probleem op dat het haast onmogelijk is om te achterhalen waar een bepaald object opgevraagd moet worden.

1. **Herkomst van data**

Een API biedt in geen van de gevallen meta data over het object. Het is daardoor onmogelijk om de actualiteit, nauwkeurigheid en betrouwbaarheid van data te valideren.

1. **Configuratie last**

Om de API’s aan elkaar te kunnen relateren, de resultaten van een API om te zetten in semantisch formaat en om de API configuratie te beschrijven is een enorme configuratie last nodig. Voor de beperkte API’s o dit moment hebben hier al ruim 4000 regels aan configuratie nodig. Ook het onderhoud van deze configuratie zal een redelijke last met zich mee brengen.

1. **Geografische vraag kenmerkt zich door missende relatie (en waar naar te zoeken)**

Bij de geografische vraag is gaat de demonstrator op zoek naar objecten die geen relatie hebben tot elkaar, deze objecten moet binnen een bepaalde straal van elkaar liggen. Echter zou dit mogelijk zijn op alle objecten die gedefinieerd zijn in de Stelselcatalogus waarvan de gedefinieerde API geografische query’s ondersteund. Echter zou dit betekenen dat met een groeiend aantal API’s dit een flinke belasting op de performance wordt. Daarnaast betekent dit, dat bijvoorbeeld alle panden worden opgevraagd in een bepaalde straal. Dit levert ontzettend veel data op, die in veel gevallen waarschijnlijk niet de vraag van de gebruiker zal beantwoorden. Op dit moment is gekozen om in de configuratie vast te leggen op welke objecten geografisch gezocht moeten worden vanuit een bepaald start object. Hierdoor kan er sturing plaats vinden.

1. **Geografische relatie obv GeoSPARQl**

Geografische data laadt zich goed lenen om met elkaar te gebruiken. Geografische data wordt in alle gevallen aangeboden in een Geo standaard. Hierdoor kunnen verschillende tools en libraries eenvoudig omgaan met geografische data ongeacht de standaard die toegepast is. Ook tools die semantisch werken kunnen goed met deze data omgaan om dat het eenvoudig te gebruiken is met Semantische Geo standaarden.