API Strategie Algemeen (Nederlandse API Strategie I)

Geonovum Handreiking
Vastgestelde versie 09 maart 2022

Deze versie:

https://docs.geostandaarden.nl/api/def-hr-API-Strategie-20220309/

Laatst gepubliceerde versie:

https://docs.geostandaarden.nl/api/API-Strategie/

Vorige versie:

https://docs.geostandaarden.nl/api/cv-hr-API-Strategie-20220113/

Laatste werkversie:

https://geonovum.github.io/KP-APIs/API-strategie-algemeen/

Redacteurs:

Frank Terpstra, Geonovum

Jan van Gelder, Geonovum

Auteurs:

Lancelot Schellevis, Forum Standaardisatie

Han Zuidweg, Forum Standaardisatie

Friso Penninga, Geonovum

Matthias Snoei, Swis

Jasper Roes, Het Kadaster

Peter Haasnoot, Logius

Frank van Es, Logius

Dennis de Wit, Solventa

Doe mee:

GitHub geonovum/KP-APIs

Dien een melding in

Revisiehistorie

Pull requests

Rechtenbeleid:

Creative Commons Attribution 4.0 International Public License (CC-BY)

Samenvatting

Dit document beschrijft een API strategie voor de Nederlandse overheid.

Status van dit document

Deze paragraaf beschrijft de status van dit document ten tijde van publicatie. Het is mogelijk dat er actuelere versies van dit document bestaan. Een lijst van Geonovum publicaties en de laatste gepubliceerde versie van dit document zijn te vinden op https://www.geonovum.nl/geo-standaarden/alle-standaarden.

Dit is de definitieve versie van de handreiking. Wijzigingen naar aanleiding van consultaties zijn doorgevoerd.

Ten opzichte van de vorige versie van de API strategie (28-06-2021) is het hoofdstuk "API Strategie voor de overheid" aangepast.

Inhoudsopgave

1. Inleiding

Dit onderdeel is niet normatief.

Dit hoofdstuk geeft een inleiding op de Nederlandse API strategie

1.1 Status van de API strategie

In deze versie van de API strategie zijn de op en aanmerkingen uit de consultatie verwerkt. Op GitHub kan bekeken worden wat er precies is gedaan met de op en aanmerkingen.

1.2 Auteurs

Er worden slechts 6 auteurs genoemd, echter aan deze strategie is door veel meer mensen gewerkt. De genoemde auteurs zijn de 6 trekkers van de 5 werkgroepen API Strategie, Architectuur, Authenticatie en Autorisatie, Communicatie en Beleid, en Gebruikerswensen

1.3 Leeswijzer

De API strategie bestaat uit een drietal verschillende documenten.

Dit document bevat twee delen, waarbij het eerste deel "niet-technisch" en het tweede deel "technisch" van aard is. In het eerste deel zitten de hoofdstukken Communicatie en Beleid en Gebruikerswensen.

Het tweede deel bevat de hoofdstukken Standaarden en Architectuur, waarin een technische uitwerking van de eerste twee hoofdstukken staat.

2. API strategie voor de overheid

Dit onderdeel is niet normatief.

2.1 Visie

In onze visie zal de digitale overheid de komende jaren gegevens en data steeds meer veilig en efficiënt uitwisselen door applicaties zoveel mogelijk te scheiden van de data, en deze data bij de bron te bewaren. De voorzieningen van de digitale overheden dienen de data zoveel mogelijk actueel te gaan halen - in een veilige infrastructuur -zodra het voor een zaak nodig is. Hiermee biedt de overheid een gedemocratiseerde toegang tot evenwichtige en consistente data sets, onafhankelijk van de complexiteit van achterliggende systemen, voor burgers, bedrijven en ketenpartners van de overheid. Wij leven bovendien in een maatschappij waarin ICT-platformen een steeds grotere rol spelen in het bij elkaar brengen van vraag en aanbod van producten en diensten. Denk aan AirBnB, Über, Funda en Thuisbezorgd. Maar ook aan publieke platformen zoals OVChipkaart.nl en Publieke Dienstverlening Op Kaart. Een belangrijk kenmerk van deze platformen is het gemak en de flexibiliteit waarmee gebruikers diensten kunnen aanbieden en afnemen. Zo kan Funda een applicatie als Google Maps gebruiken om op een kaart te laten zien waar in Nederland huizen te koop staan. Burgers en bedrijven verwachten dit ook van de overheid. Daarbij mogen zij verwachten dat iedereen binnen de overheid gebruik maakt van dezelfde data sets. Volgens de Kamer van Koophandel is er geen weg meer terug uit deze platformeconomie. Ook de beleidslijnen vanuit de Europese Unie lopen in dezelfde richting, met onder meer het voorstel voor een Europese datastrategie en de EU richtlijn inzake open data en hergebruik van overheidsinformatie.

Door standaardisatie van gegevensuitwisseling, het ontsluiten van data sets bij de bron en het hergebruik van de data en informatieproducten, neemt de operabiliteit van de overheid enorm toe en is de overheid in staat om sneller dienstverlening aan het publiek te leveren. De overheid gaat zich meer als een platform gedragen, Government as a platform. Daar waar de overheid traditioneel is georganiseerd als silo’s, worden grenzen doorbroken door het hergebruik van informatieproducten over de silo’s heen te faciliteren. Dienstverlening kan op deze manier meer rond de burger en ondernemer worden georganiseerd, in plaats van volgens de structuur van de organisaties binnen de overheid.

Platformen hebben gestandaardiseerde ICT koppelvlakken die applicaties en gegevensbronnen aan elkaar verbinden. Hiervoor gebruiken zij APIs. Een application programming interface (API) is een gestructureerd en gedocumenteerd koppelvlak voor communicatie tussen applicaties. Je kan een API zien als een digitale stekkerdoos die applicaties met elkaar verbindt. APIs bestaan al

zo lang er computers zijn. De meeste platformen gebruiken APIs onder de motorkap zonder dat de eindgebruiker dat ziet. Als je bijvoorbeeld een app opent voor het weer, dan merk je niet dat deze app gegevens ophaalt bij het KNMI via een API. Degene die het meest met een API te maken krijgt is de ontwikkelaar van de applicatie die de API gebruikt. Zij moet begrijpen hoe de API werkt en hoe je deze vanuit de applicatie moet bevragen. Ook de overheid gebruikt steeds vaker APIs om applicaties en gegevensbronnen te ontsluiten en met elkaar te koppelen. Bijvoorbeeld in initiatieven zoals Haal Centraal, Common Ground en het Digitaal Stelsel Omgevingswet. De overheid heeft door ervaring van deze initiatieven voldoende kennis en expertise voor het gebruik van APIs in een moderene informatiehuishouding.

Toch heeft de overheid ook veel ICT applicaties in gebruik die niet als platformen werken, maar als gesloten systemen waarin de data helemaal met de applicatie verweven zijn. Vaak werken deze applicaties met kopieën van cruciale data sets zoals basisregistraties. Dat brengt een aantal nadelen met zich mee:

1. De kans op fouten en datalekken neemt toe bij het kopiëren en verspreid opslaan van data sets.
2. Het kopiëren en synchroniseren van gegevens tussen systemen is vaak inefficiënt en kost meer geld dan nodig.
3. Bij nieuw beleid dat nieuwe gegevensuitwisselingen nodig maakt, moeten op veel plaatsen maatwerk koppelvlakken worden gebouwd, en geïmplementeerd, waardoor vernieuwing in beleid lange doorlooptijden van invoering kennen.
4. Personen en bedrijven hebben geen regie over hun gegevens als deze bij verschillende organisaties verspreid staan.
5. Applicaties met maatwerk koppelvlakken kunnen afhankelijkheid van leveranciers in de hand werken.

De platformeconomie is actueel en maatschappelijk relevant of het we het nu hebben over de Informatiehuishouding op orde (rapport ‘Ongekend onrecht’ de kinderopvangtoeslag problematiek en ‘Werk aan uitvoering’ de problemen bij uitvoeringsorganisaties), over de Interbestuurlijke Datastrategie, over de toekomstvisie op het stelsel van Basisregistraties, of de over doorontwikkeling van de GDI (meerjarig geprogrammeerd en onder architectuur): ze veronderstellen een modernisering van de binnenkant van de gegevensuitwisseling. Platformen, API’s en afgedwongen standaarden zijn daarin onmisbaar.

2.2 Intentieovereenkomst

Bij het realiseren van deze visie volgt de overheid een aantal bestaande afspraken (1 /tm 3) van het Forum Standaardisatie deze zijn al bekrachtigd in het Overheidsbreed Beleidsoverleg Digitale Overheid (OBDO) :

1. Waar de overheid REST APIs inzet, gebeurt dat volgens de REST API Design Rules. Zo biedt de overheid haar data en diensten via REST APIs op een standaard manier en goed gedocumenteerd aan.
2. Veilige autorisatie van toegang tot data via REST APIs realiseert de overheid met het NL GOV Assurance Profile for OAuth 2.0. Dit profiel op de authorisatie standaard OAuth is speciaal ontwikkeld voor de Nederlandse overheid.
3. Voor het realiseren van veilige koppelvlakken gebruikt de overheid de standaard DigiKoppeling. Deze standaard maakt het ook mogelijk om een REST API aan te bieden op een veilige manier die voldoet aan punten 1 en 2 hierboven.
4. De overheid registreert haar extern toegankelijke REST APIs bij developer.overheid.nl. Zo kunnen ontwikkelaars de APIs van de overheid altijd vinden.

Het aantal samenwerkende partijen binnen het Kennisplatform API's wordt verder uitgebreid. Gezamelijk wordt een concrete strategie opgesteld om de visie te realiseren.

2.3 Praktijkvoorbeelden

2.3.1 De API op de Basisregistratie Personen (BRP) van Haal Centraal

Persoonsgegevens uit de Basisregistratie Personen (BRP) worden nu vaak gesynchroniseerd met een lokale kopiedatabase bij een gemeente. In de processen van de gemeente wordt dan die kopiedatabase bevraagd in processen waar persoonsgegevens nodig zijn. Binnen het programma Haal Centraal experimenteert de Rijksdienst voor Identiteitsgegevens (RvIG), de vijf grote gemeenten (Amsterdam, Den Haag, Eindhoven, Rotterdam en Utrecht) en VNG Realisatie met een API waarmee gemeenten in hun werkprocessen direct de Basisregistratie Personen (BRP) kunnen bevragen. In totaal 10 gemeenten gaan deze API gebruiken en beproeven in een pilot.

Over de voordelen van de BRP API heeft Haal Centraal een heldere video gemaakt. De specificatie van de API uit de pilot is te vinden op Github repository van VNG. Wanneer de pilot slaagt, komt mogelijk een API beschikbaar voor alle gemeenten en andere overheidsorganisaties die de BRP nu via het huidige berichtenverkeer raadplegen.

2.3.2 De API voor Zaakgericht Werken van Common Ground

Veel gemeenten werken zaakgericht. Dit betekent dat informatie over dienstverlening aan inwoners in zaken is geordend in een zaaksysteem. Deze manier van werken richt zich op het transparant afhandelen van een samenhangende hoeveelheid werk (een 'zaak'), welke een duidelijk omschreven aanleiding kent (bijvoorbeeld een aanvraag) en leidt tot een duidelijk omschreven resultaat.

ICT systemen ondersteunen zaakgericht werken met als voordelen het automatische termijnbewaking en routeren van taken naar betrokken medewerkers of systemen. Met daarbij ook het parallel kunnen uitvoeren van samenhangende taken, geautomatiseerd toekennen van metadata en het duurzaam toegankelijk bewaren van het zaakdossier. De API-standaarden voor zaakgericht werken helpen bij het (ont)koppelen van zaaksystemen en de registers waar documenten en overige informatie in zijn opgeslagen. Als alle gemeenten de API-standaarden voor zaakgericht werken gebruiken, wordt de uitwisseling van gegevens voor deze systemen efficiënter. Het is bovendien een belangrijke stap in het realiseren van Common Ground, iets waar alle gemeenten naar streven.

2.3.3 De API voor Basisregistratie Kadaster (BRK) - Publiekrechtelijke Beperkingen

De Wet kenbaarheid publiekrechtelijke beperkingen onroerende zaken (Wkpb) geeft inzicht in door de overheid opgelegde beperkingen op een stuk grond of een daarop staand gebouw. Sinds 2021 is de vernieuwde Wet kenbaarheid publiekrechtelijke beperkingen (Wkpb) van kracht. Deze vernieuwde wet schrijft wijzigingen voor in het systeem voor het kenbaar maken van publiekrechtelijke beperkingen. Inwoners, ondernemers en overige partijen kunnen alle informatie over beperkingen en bijbehorende brondocumenten bij één partij kunnen vinden: het Kadaster.

In het oude systeem registreerden de gemeenten hun eigen gegevens terwijl andere overheden de basisregistratie van het Kadaster gebruikten. Deze twee langs elkaar werkende systemen brachten allerlei problemen met zich mee. Het was niet mogelijk om gemeentelijke brondocumenten bij het Kadaster op te vragen. En soms bleek de gemeentelijke situatie van een pand of object niet overeen te komen met de kenmerken zoals die in de basisregisters van het Kadaster zijn vastgelegd. Voorts konden overheden in het oude systeem alleen beperkingen opleggen op een kadastraal perceel, wat in veel gevallen niet aansloot op de praktijk. De vernieuwde Wkpb registreert en beheert alle publiekrechtelijke beperkingen in de Basisregistratie Kadaster publiekrechtelijke beperkingen (BRK-PB). Overheden en organisaties uit de private sector hebben zo een unieke bron waar ze alle gegevens kunnen raadplegen. En omdat beperkingen nu,

behalve op een perceel, ook gelegd kunnen worden op objecten uit de de basisregistraties Adressen en Gebouwen (BAG) en Grootschalige Topografie (BGT), of via een zelf vervaardigde contour, sluit de registratie veel beter aan op de uitvoeringspraktijk.

De BRK-PB kan al benaderd worden met de BRK-API. Het bij de bron registreren en beheren van gegevens heeft hier dus zichtbare meerwaarde, zowel voor de overheid als voor de private sector , zie ook dit artikel in Binnenlands Bestuur.

2.3.4 De API voor regie op gegevens

De Wet bescherming persoonsgegevens (Wrp) en z'n opvolger, de Algemene verordening gegevensbescherming (AVG) geven ons al bijna twintig jaar recht op inzage en correctie van de gegevens die de overheid over ons verzamelt en beheert. Dit betekent dat je als burger het recht hebt om te weten welke persoonsgegevens de overheid over je heeft vastgelegd en welke gegevens de overheid heeft gebruikt en uitgewisseld voor welk doel. Uit dit onderzoek uit 2017 van Berenschot (uitgevoerd in opdracht van BZK) blijkt dat volledig digitale inzage in alle relevante persoonsgegevens die onder de AVG vallen, in het huidige gegevenslandschap niet haalbaar is.

In de beleidsbrief regie op gegevens regie op gegevens van 2019 onderstreept het kabinet nog eens het beleidsvoornemen om serieus werk te maken van regie op gegevens. Maar dan er zal er dus iets moeten gebeuren in het gegevenslandschap om dit mogelijk te maken.

Om het inzicht in de verstrekking van persoonsgegevens veel eenvoudiger te maken, is er door VNG-Realisatie een model API opgesteld, die overheden in staat kan stellen de verwerking van persoonsgegevens op een toegankelijke manier te tonen. Deze API haakt aan bij wat conform de AVG al tot stand komt: de organisatie-brede registers van verwerkingen van persoonsgegevens. Als de overheid gegevens bij de bron gaat beheren en ontsluiten met APIs, wordt het gemakkelijker om bij de bron volledige inzage te geven. De overheid kan dan digitale inzage geven in een uniform gebruiksvriendelijk format, en ook de correctie van gegevens wordt eenvoudiger.

2.4 Samenvattend

De platformeconomie biedt nieuwe kansen voor de digitale overheid. Om deze kansen te benutten en tegelijkertijd de dienstverlening van de digitale overheid te verbeteren vindt er transitie plaats in het gegevenslandschap. Initiatieven zoals Common Ground, Haal Centraal en

het Digitaal Stelsel Omgevingswet (DSO) willen de digitale overheid efficiënter, slimmer, veiliger en beter beheersbaar maken door applicaties beter te scheiden van de gegevens en de gegevens alleen bij de bron te beheren. Hierdoor hoeven gegevens niet meer op grote schaal gedupliceerd en gesynchroniseerd te worden. In deze ambitie spelen API's als ondersteunende technologie een belangrijke rol. Nagenoeg alle organisaties binnen de overheid zijn de omslag van traditionele IT naar een API-ecosysteem aan het maken. Dit hoofdstuk biedt een kader aan waarbinnen de overheid zich kan committeren bij ontwikkeling en te realiseren beleid rondom API's. Om die reden krijgen organisaties die aan deze API Strategie hebben bijgedragen het verzoek om de intentieovereenkomst te ondertekenen.

2.5 Ondertekening

De hieronder vermelde organisaties hebben bijgedragen aan deze API strategie voor de overheid en onderschrijven de basisafspraken: ... | Forum Standaardisatie | Geonovum | ICTU | Logius | Ministerie van Binnenlandse Zaken en Koninkrijksrelaties | VNG Realisatie | ...

Dit is nog geen officiële lijst van ondertekenaars

3. Inspelen op gebruikerswensen: dé sleutel tot gebruik

Dit onderdeel is niet normatief.

3.1 Inleiding

Overheden bezitten kwalitatief hoogwaardige data en bieden deze aan via API's. Vanuit het open data-perspectief maken overheden het hiermee mogelijk dat anderen -binnen én buiten de overheid- zinvolle toepassingen kunnen ontwikkelen, die maatschappelijke meerwaarde bieden. Maar de overheid gaat nog verder: bij initiatieven als Common Ground is het gebruik van API's zelfs randvoorwaardelijk om werkprocessen zodanig in te richten, dat data altijd bij de bron bevraagd en gewijzigd kan worden. Een belangrijke succesfactor in deze ontwikkelingen is de mate waarin overheden erin slagen om drempels voor het gebruik van hun API's weg te nemen. Om het gebruik zo laagdrempelig mogelijk te maken, is het essentieel dat er aandacht is voor de wensen van gebruikers. En om te begrijpen welke wensen ze hebben, is het essentieel om te begrijpen wie die gebruikers zijn. In generieke zin zijn de gebruikers van API's developers: mensen die toepassingen ontwikkelen. Deze developers fungeren als intermediair tussen de data- of dienstaanbieder enerzijds en de eindgebruiker anderzijds. Neem bijvoorbeeld een gemeente die het mogelijk wil maken dat burgers makkelijk melding kunnen maken van een defecte lantarenpaal, losliggende stoeptegel of zwerfvuil, De burger is dan de eindgebruiker, maar de ontwikkelaar van de Melding Openbare Ruimte app is de gebruiker van de API's die de gemeente aanbiedt om bijvoorbeeld een kaartondergrond op te halen of een melding aan te bieden aan de gemeente. De gebruikerswensen liggen niet alleen op het "wat" (wat voor data krijg ik?), maar zeker ook op het "hoe" (hoe makkelijk kan ik aan de slag met deze API?). Bij de eerste kennismaking met de API moet de gebruiker verleidt worden om de API te gaan gebruiken en in alle volgende fases van gebruik (implementeren, in productie nemen en in productie houden) moet de gebruikerservaring zodanig zijn, dat de gebruiker geen enkele reden heeft om af te haken. Sterker nog: idealiter wordt de gebruiker zó enthousiast over de API en de bijbehorende ondersteuning, dat de gebruiker onderdeel wordt van de gebruikerscommunity en actief gaat bijdragen aan verdere ontwikkeling en promotie van die API. Merk op dat dit effect alleen bereikt kan worden, wanneer de totaalbeleving klopt: ook al is het product (de data) zelf nog zo goed, als de gebruikservaring belabberd is, dan zul je niet veel potentiële gebruikers verleiden om met je data zinvolle toepassingen te gaan ontwikkelen. En nog erger: gebruikers die tóch met de API aan de slag gaan, zullen de helpdesk van de aanbieder zwaar gaan belasten, de data mogelijk verkeerd gaan interpreteren en de ICT-infrastructuur onnodig zwaar belasten, wanneer zij slechts omslachtig (met veel API-calls) de gewenste gegevens kunnen krijgen.

3.2 Overkoepelende aanbeveling: biedt een goede ‘developer experience (DX)’

De belangrijkste aanbeveling aan aanbieders van API’s is om zich te richten op die gebruikservaring; op een goede ‘developer experience (DX)’. Goede DX komt voort uit stapeling van aandacht voor *functionality* (functionaliteit - wat moet de API doen?), voor *usability* (hoe bruikbaar is de API voor de developer?) en daar bovenop voor de *experience* (hoe voelt de developer zich als die de API gebruikt?). Dat laatste aspect -hoe voelt de developer zich- klinkt wellicht wat vaag, maar het is in essentie de optelsom van ervaringen in de interactie tussen een developer en een API aanbieder. Stel: een developer, Johan (26 jaar – ZZP-er), heeft moeite om een bepaalde API te vinden, vervolgens blijkt het registratieproces lastig en krijgt hij de API key pas een week later, daarna blijkt de documentatie in voor Johan deels onbegrijpelijk jargon geschreven (én is het ook nog eens een PDF van 352 pagina’s), dan heeft Johan meerdere calls nodig om de gewenste data te krijgen en uiteindelijk is data helemaal niet in het door Johan gewenst formaat. Elke kleine irritatie is op zichzelf niet onoverkomelijk, maar de optelsom maakt dat Johan concludeert: “Dit is niet te doen, ik zoek wel wat anders!”.

3.3 Gebruik: van ‘onboarding’ tot ‘in productie’

De zwakte van veel teksten over ‘developer experience’ is dat men zich vaak alleen richt op de use case van een technische developer die een API wil implementeren - denk aan Johan in de vorige paragraaf. Gebruik begint echter niet bij de poging tot implementatie; hier gaan nog stappen aan vooraf. Johan heeft zich al een aantal vragen gesteld, voordat hij besloot om die poging te wagen. Die vragen -en daarmee de stappen die hij doorloopt- zijn beter te introduceren aan de hand van het voorbeeld van een grotere softwareleverancier. Stel: Anne (43 jaar) werkt als product manager bij een commerciële softwareleverancier en heeft een aantal applicaties voor de gemeentelijke markt in haar portfolio. Ze ziet een API op een ontwikkelaarsportaal en beoordeelt aan de hand van de functionele specificaties of implementatie van de API meerwaarde biedt voor één van haar applicaties. Eén API trekt haar aandacht en Anne vraagt Steven (56 jaar, architect bij hetzelfde bedrijf) om te beoordelen of de API past binnen de software stack van het bedrijf en kritisch te kijken naar eventuele security en privacy issues. Steven ziet mogelijkheden en vervolgens krijgt Jeffrey (37 jaar, software engineer bij hetzelfde bedrijf) de opdracht om binnen een dag te beoordelen of hij de API succesvol kan integreren in de nieuwe versie van het product uit Anne’s portfolio. Na Jeffrey’s positieve oordeel besluit Gea, de manager, om tijd en geld vrij te maken voor implementatie. Zo komt de API uiteindelijk in de productieversie terecht: *"developer tries, business buys"*.

Dit eindresultaat is bereikt, doordat in alle fases de ‘onboarding’ -het proces dat Anne, Steven en Jeffrey doorliepen toen zij voor de eerste keer gebruik maakten van een digitale service (het

aanbieden van de API)- prettig verliep. Uit marketingonderzoeken blijkt een prettige eerste gebruikerservaring van een online platform (bijvoorbeeld het ontwikkelaarsportaal waarop Anne de API ontdekte en zij vervolgens samen met Steven en Jeffrey de API beoordeelde) veel invloed heeft op het verdere gebruik ervan. Oftewel: de kans is groot dat dit bedrijf vervolgens ook andere API's op dit ontwikkelaarsportaal gaat gebruiken.

De kern van dit verhaal is dat een API technisch gezien weliswaar een machine tot machine koppeling is, maar strategisch juist gezien moet worden als een product. En dat product moet gebruikers (zie ze als potentiële klanten!) verleiden tot gebruik. Dan zijn aspecten als de winkelervaring (in een API store of ontwikkelaarsportaal), de geboden service en de kwaliteit van het product de doorslaggevende factoren in de beslissing om tot gebruik over te gaan.

3.4 Specifieke aanbevelingen voor een goede DX

3.4.1 Aanbeveling 1: werk met (meerdere) persona's

Een API is weliswaar een machine tot machine koppeling, maar onthoud: API's ontwerp en bouw je niet voor een machine, maar voor een gebruiker: een mens! Om een goede DX te bieden, moet je dus eerst weten voor wie je ontwerpt en bouwt, voordat je dat goed kunt doen. Persona's zijn dan belangrijk, er is niet maar één type developer! Redeneer van buiten naar binnen: wie zijn mijn gebruikers, wat willen zij kunnen doen en wat moet ik doen om dat zo goed mogelijk te faciliteren. Het typeren van gebruikers en analyseren welke behoeften zij hebben, doe je op basis van persona's.

Belangrijk hierbij is om te onderkennen dat er verschillende niveaus of typen gebruik zijn. Onderscheid daarom -met de fases van 'onboarding' tot 'in productie' in het achterhoofd- minimaal de volgende persona's:

- een product manager / business developer: focus op het kunnen beoordelen van functionaliteit ("Is dit relevant voor mijn product / mijn doel?")
- een architect: focus op het beoordelen van het informatiemodel ("Hoe integreert dit met de rest van onze software?") en security- en privacy-aspecten.
- een technische developer: focus op het daadwerkelijk kunnen gebruiken ("Hoe krijg ik dit werkend?").

3.4.2 Aanbeveling 2: analyseer welke API's je aan moet bieden: welke informatievragen wil je beantwoorden?

De ene API is de andere niet. In veel modellen worden API's in drie categorieën onderverdeeld: de System API (die werkt op het niveau van de databron), de Process API (die doet aan *orchestration* door één of meerdere System API's aan te roepen) en de Convenience of Experience API (die één specifieke gebruikersvraag beantwoord). Vraag je altijd af welke informatievragen de gebruiker heeft – in veel gevallen hangen deze vragen niet 1:1 samen met het datamodel.

Stel dat je de Basisregistratie Adressen en Gebouwen wil ontsluiten. De informatievraag "Geef me het volledige adres bij deze postcode" is alleen te beantwoorden door intern bij een Verblijfsobject de Openbare Ruimte naam, Nummeraanduiding en Woonplaats op te vragen en te combineren tot één adressering. Bij een System API moet je meerdere calls doen om dit adres op te bouwen, terwijl een Convenience API deze gangbare (maar complexe) vraag in één call kan beantwoorden. Het aanbieden van Convenience API's naast System API's is dus erg gebruiksvriendelijk: de 80% gebruikers die hiermee geholpen zijn, confronteer je met slechts 20% van de complexiteit. Vergelijk dit met de Toptaken-aanpak bij de websites van veel gemeenten: de meest aangevraagde producten (bijv. nieuw paspoort en rijbewijs) staan pontificaal op de homepage, terwijl de minder vaak gevraagde diensten op vervolgpagina's staan. Zo reduceer je de complexiteit voor een zo groot mogelijk deel van je gebruikers.

3.4.3 Aanbeveling 3: documenteer gericht op de gebruiker, biedt snel inzicht en gebruik OAS 3

Elk type gebruiker dat in aanbeveling 1 wordt genoemd (business developer, architect, technische developer) verdient zijn eigen 'Getting started' documentatie, gericht op het snel kunnen beoordelen en/of toepassen. Documentatie is nooit een lijvig document (wanneer veel documentatie nodig lijkt, is de functionaliteit vermoedelijk te complex) én nooit een PDF: laat je gebruikers makkelijk klikken naar relevante onderdelen binnen én buiten de documentatie! Referentie-implementaties kunnen ook zinvol zijn om snel een indruk te bieden van functionaliteit. Voor de technische developer is de documentatie conform de Open API Specifiction 3.0 (OAS 3); deze staat in Nederland op de 'Pas toe of leg uit-lijst' van het Forum Standaardisatie.

3.4.4 Aanbeveling 4: minimaliseer Time to First Call met een goede Sandbox

Zorg dat een developer snel een werkend voorbeeld heeft. Dit vraagt om een goed gedocumenteerde, realistische Sandbox: een experimenteeromgeving met testdata. Deze Sandbox dient alle aspecten van de API te ondersteunen en identiek gedrag aan de productieversie van de API te vertonen, bijvoorbeeld rond authenticatie. Daar waar mogelijk is het zeer wenselijk dat meerdere API's dezelfde dataset bieden als Sandbox, zodat ook het samenspel tussen verschillende API's getest kan worden.

3.4.5 Aanbeveling 5: borg ontwikkeling en beheer

Ook al is een API nog zo goed ontwikkeld, wanneer doorontwikkeling en beheer niet goed geregeld is, zal die API niet succesvol zijn. Essentieel hierin is dat je gebruikers duidelijkheid biedt:

3.4.5.1 Aanbeveling 5.1 Stel een SLA op

> Maak duidelijk welke verwachtingen een gebruiker mag hebben qua uptime, service window, beprijzing etc.

3.4.5.2 Aanbeveling 5.2 Biedt een roadmap aan

> Maak duidelijk of en zo ja, wanneer er eventuele wijzigingen te verwachten zijn en hoe lang de API minimaal beschikbaar blijft.

3.4.5.3 Aanbeveling 5.3 Doe aan versiebeheer

> Borg dat de toepassing van de gebruiker blijft werken, door te zorgen voor backward compatability. Grotere updates kunnen als nieuwe versie worden uitgebracht, waarbij oudere versies nog een gegarandeerde periode beschikbaar blijven. Het versienummer kan in elke call staan, bijv. GET /api/v1.0/... En versiebeheer slaat niet alleen op de API zelf, maar ook op de bijbehorende documentatie, Sandbox, etc.

3.4.5.4 Aanbeveling 5.4 Sluit de feedback-loop: betrek de community

> Om echt van buiten naar binnen te kunnen werken, is het betrekken van de community van gebruikers onmisbaar. De community kan vertellen hoe de developer experience tot nu toe is, welke verbeteringen wenselijk zijn, welke gebruikersvragen er sterk leven, maar nog onvoldoende ondersteund worden, etc., etc. Daarnaast wil je de community ook actief informeren over voorgenomen wijzigingen e.d. Het gebruik van API keys is een manier om je gebruikers te kennen.

3.4.6 Aanbeveling 6: maak duidelijk wat je data betekent

Wanneer je data openstelt voor derden, inclusief niet-specialisten, is het essentieel om eenduidig vast te leggen wat de data betekent, waarbij deze betekenis ook voor niet-specialisten begrijpelijk is. Het vastleggen van semantiek kan o.a. door definities en informatiemodellen goed online te ontsluiten (zoals bijvoorbeeld in de Stelselcatalogus Omgevingswet), maar ook door praktischere zaken als heldere naamgeving van variabelen etc.

3.4.7 Aanbeveling 7: wees vindbaar voor developers

Een goede, gebruiksgerichte API, die bovendien actief wordt beheerd en doorontwikkeld, kan nog steeds weinig gebruikt worden wanneer deze API niet goed vindbaar is. Zorg daarom dat je een goed developersportaal hebt, nationaal idealiter bij developer.overheid.nl. Als een API een product is, hoort dat product ook in een goede winkel te staan. En in een succesvolle winkel is altijd goed nagedacht over productpresentatie: presenteer breed toepasbare API’s en datasets prominenter dan specifiekere API’s en obscure datasets, wederom te vergelijken met de toptaken-aanpak bij gemeentelijke websites. Een goed developersportaal informeert niet alleen over beschikbare API’s, maar inspireert en verleidt zelfs developers om bepaalde API’s te gebruiken. Het vullen van het developersportaal mag daarom nooit sluitpost van een project zijn. Formuleer hierbij ook goed het ambitieniveau:

- API store: de one-stop-shop, waarin je niet alleen (het bestaan van) de API ontdekt, maar ook de documentatie vindt, de API kunt uitproberen, etc. Het runnen van een API store kan zelfs zover gaan, dat er centraal proxy's worden ontwikkeld op API's van andere aanbieders en documentatie geredigeerd, om zo voor de eigen gebruikers een zo uniform mogelijke gebruikerservaring te garanderen.
- API catalogus: uitsluitend gericht op het ontdekken van API's, waarna doorverwezen wordt naar de API store van de desbetreffende aanbieder.

- hybride oplossing: een combinatie van een catalogus en een store. Alle API's staan in de catalogus, een kleine subset (de high-value API's) wordt volledig aangeboden. Deze hybride oplossing zou een mooi ambitieniveau voor developer.overheid.nl kunnen zijn: alle overheids-API's zijn vindbaar en voor de meest generieke en populaire API's ben je direct op de juiste plaats.

Een bijzondere categorie die de overheid ook nodig gaat hebben, o.a. door de ontwikkelingen in Common Ground, is de API spec store: een plaats waarin API specificaties gepubliceerd kunnen worden. Leveranciers kunnen op basis van die specificaties zelf hun specificatie-conforme API's ontwikkelen, waarmee de interoperabiliteit van gegevens binnen het gemeentelijke applicatielandschap wordt vergroot.

In de context van API stores en API catalogi is het belangrijk om te realiseren dat developer experience niet alleen een externe focus heeft (gericht op laten ontdekken, evalueren, testen en gebruiken van API's), maar ook een interne focus, gericht op het activeren van potentiële API-aanbieders. Zij moeten getriggerd worden om API's te ontwikkelen, te ontsluiten in je API store of catalogus, te beschrijven conform bepaalde kwaliteitseisen, etc.

3.4.8 Aanbeveling 8: niet alles is een API!

Bedenkt altijd goed of een API de juiste oplossing is. In sommige gevallen is een bulk download nog steeds praktischer voor een gebruiker. Wanneer een API zinvol is? Hoe hoger de mutatiefrequentie van de data, hoe zinvoller een API wordt. En bij hoge mutatiefrequenties, is een API die was-wordt leveringen kan bieden zinvol. Denk hierbij weer aan 'API als een product': ga niet blind data ontsluiten, maar begin bij de vraag welke propositie je neer wil zetten.

4. Standaarden

Dit onderdeel is niet normatief.

Binnen de Nederlandse publieke sector zijn meerdere standaarden die betrekking hebben op APIs. In dit hoofdstuk geven we verwijzingen naar de standaarden die zijn voortgekomen uit het Kennisplatform APIs. In een volgende versie van dit document zal hier een compleet overzicht met relevante standaarden komen.

4.1 API Designrules (Nederlandse API Strategie IIa)

Deze sectie beschrijft de status en uitgangspunten voor de standaard "API Designrules". De standaard is een op zichzelfstaand document en is hier te vinden:

https://logius-standaarden.github.io/API-Design-Rules/ (werkversie)

https://publicatie.centrumvoorstandaarden.nl/api/adr/ (stabiele versie)

4.1.1 Oorsprong standaard

De API designrules vinden hun oorsprong in de "API strategie" van het digitaal stelsel omgevingswet (DSO). Bij het opstellen van de API Designrules is de DSO API strategie het startpunt geweest en heeft de werkgroep API strategie (onder leiding van Jasper Roes) gekeken hoe deze toepasbaar gemaakt kunnen worden voor de hele Nederlandse publieke sector.

4.1.2 Status bij forum standaardisatie

De API designrules standaard is op 9 juli 2020 goedgekeurd door het OBDO voor opname op de "pas toe of leg uit lijst" van het forum standaardisatie.

4.1.2.1 Werkingsgebied standaard

Bij aanmelding was het beoogd organisatorisch werkingsgebied: Overheden, semi-overheden en instellingen uit de publieke sector

4.1.2.2 Toepassingsgebied standaard

Bij aanmelding was het beoogd functioneel toepassingsgebied: Het aanbieden van RESTful APIs ten behoeve van het ontsluiten van overheids informatie/functionaliteit en enkelvoudige datasets aan eenieder waarvoor deze bedoeld is(burger, bedrijf, andere overheid, etc…). Het ontsluiten van meervoudige en/of statistische datasets via één API valt buiten het functioneel werkingsgebied hiervoor staat de standaard ODATA reeds op de lijst met aanbevolen standaarden.

4.2 API Designrules extensions (Nederlandse API Strategie IIb)

Deze sectie beschrijft de status en uitgangspunten voor de standaard "API Designrules extensions". De (concept) standaard is een op zichzelfstaand document en is hier te vinden:

https://geonovum.github.io/KP-APIs/API-strategie-extensies/ (werkversie)

https://docs.geostandaarden.nl/api/vv-hr-API-Strategie-ext-20190715/ (stabiele versie)

4.2.1 Oorsprong standaard

De API designrules extensions bevatten onderwerpen die bij het opstellen van de eerste versie van de API designrules nog niet als stabiel werden beschouwd. Er wordt nog actief gewerkt om deze extensies tot een stabiele versie te maken. Uiteindelijk zal een deel van de extensies mogelijk direct onderdeel worden van de API design rules. Een ander deel zal een optionele extensie zijn die gebruikt kan worden wanneer specifieke extra functionaliteit nodig is.

4.2.2 Status

De API designrules extensions zijn nog in ontwikkeling en hebben geen enkele status.

4.2.2.1 Werkingsgebied standaard

Een werkingsgebied is (nog) niet van toepassing.

4.2.2.2 Toepassingsgebied standaard

Een toepassingsgebied (nog) niet van toepassing.

4.3 Nederlands profiel OAuth

Deze sectie beschrijft de status en uitgangspunten voor het Nederlands profiel OAuth. het profiel zelf is een op zichzelfstaand document. Het Nederlands profiel OAuth is hier te vinden:

https://publicatie.centrumvoorstandaarden.nl/api/oauth/static.html

In aanvulling hierop is er een document dat de verschillen met iGOV kort samenvat en voorziet van rationales.

https://github.com/Logius-standaarden/OAuth-NL-profiel/blob/master/Additional%20specification%20and%20constraints%20of%20iGov-NL%20to%20the%20iGov%20profile.md

4.3.1 Oorsprong standaard

Het opstellen van deze standaard is voortgekomen uit het Expert advies OAuth [Expert]. Daarin wordt aangeraden eerst een nederlands profiel op stellen alvorens OAuth op de pas toe of leg uit lijst van het forum standaardisatie te plaatsen. Het maken van dit Nederlandse profiel is opgepakt door de werkgroep Authenticatie/Autorisatie van het Kennisplatform API's.

4.3.2 Status bij forum standaardisatie

Het Nederlands OAuth profiel is op 9 juli 2020 goedgekeurd door het OBDO voor opname op de "pas toe of leg uit lijst" van het forum standaardisatie. De internationale standaard OAuth is opgenomen op de lijst met aanbevolen standaarden.

4.3.2.1 Werkingsgebied standaard

Als organisatorisch werkingsgebied wordt in het expert advies geadviseerd: Nederlandse overheden (Rijk, provincies, gemeenten en waterschappen) en instellingen uit de (semi-) publieke sector

4.3.2.2 Toepassingsgebied standaard

Als functioneel toepassingsgebied wordt in het expert advies voorgesteld: Het gebruik van OAuth 2.0 is verplicht voor applicaties waarbij gebruikers (resource owner) toestemming geven (impliciet of expliciet) aan een dienst (van een derde) om namens hem toegang te krijgen tot specifieke gegevens via een RESTful API. Het gaat dan om een RESTful API waar de resource owner recht tot toegang heeft.

4.3.2.3 Bijzonderheden

4.3.2.3.1 OpenID connect buiten scope

de expertgroep van het forum standaardisatie is op 7 juli en op 22 september 2016 bijeengekomen om de standaarden, de aandachtspunten en openstaande vragen uit het voorbereidingsdossier te bespreken. Daarbij is vastgesteld dat OpenID Connect niet voor opneming op de lijst open standaarden in aanmerking komt. Inmiddels is OpenID connect apart voorgedragen voor opname op de "pas toe of leg uit" lijst van het forum standaardisatie. Voor deze standaard komt ook een Nederlands profiel. Dit zal ook gebaseerd zijn op iGOV zoals in de volgende sectie uitgelegd wordt.

4.3.2.3.2 Aansluiting op internationale standaard iGov

Het Nederlands profiel OAuth baseerd zich op het internationale iGOV OAuth 2.0 profiel [iGOV.OAuth2]. Niet alle keuzes van dit internationale profiel worden overgenomen aangezien dit een aantal keuzes bevat die sterk leunen op de amerikaanse situatie. Het kan het best beschouwd worden als een fork waarbij in het profiel aangeven wordt waar afwijkingen zijn. iGov heeft naast het OAuth profiel ook een OpenID connect profiel [iGOV.OpenID]. Wanneer mogelijk ook OpenID connect op de pas toe of leg uit lijst van het Forum standaardisatie komt kan het Nederlandse profiel OAuth uitgebreid worden met een Nederlandse variant van het iGov OpenID Connect profiel. De usecase die wordt beschreven aan het begin van het Nederlands profiel OAuth sorteert daar al op voor.

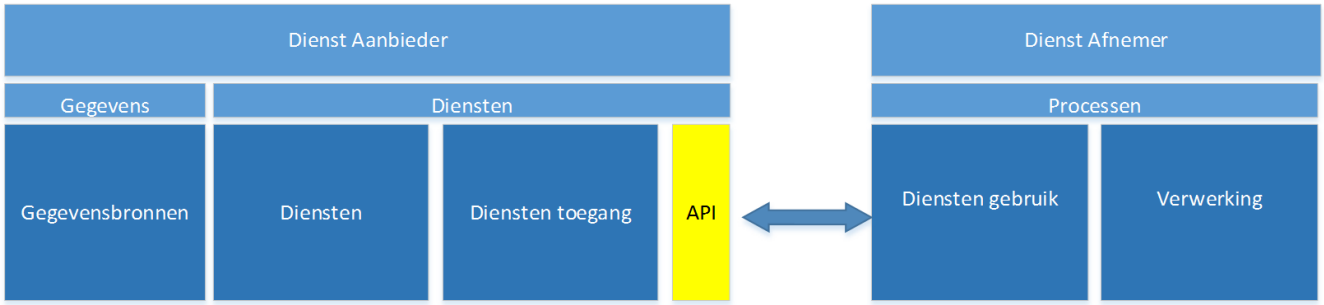
5. Architectuur

Dit onderdeel is niet normatief.

> *Dit hoofdstuk gaat in op de vraag: Hoe kan je je applicatie landschap inrichten zodat je APIs kan aanbieden. Welke componenten zijn hiervoor nodig. Hoe ga je om met opschalen, beschikbaarheid. Wat zijn afwegingen om beveiliging al dan niet toe te passen.*

5.1 Inleiding

Doel van dit hoofdstuk is om een hoog niveau overzicht te geven van relevante onderdelen en concepten en hun samenhang in een op (REST of RESTfull) API gebaseerde architectuur waarbij gebruik gemaakt wordt van REST API's als interface voor de aangeboden (gegevens)diensten. Specifiek voor REST API's is dat deze 'Resource' gericht zijn en een uniforme manier bieden om resources te lezen, wijzigen, toevoegen of verwijderen.



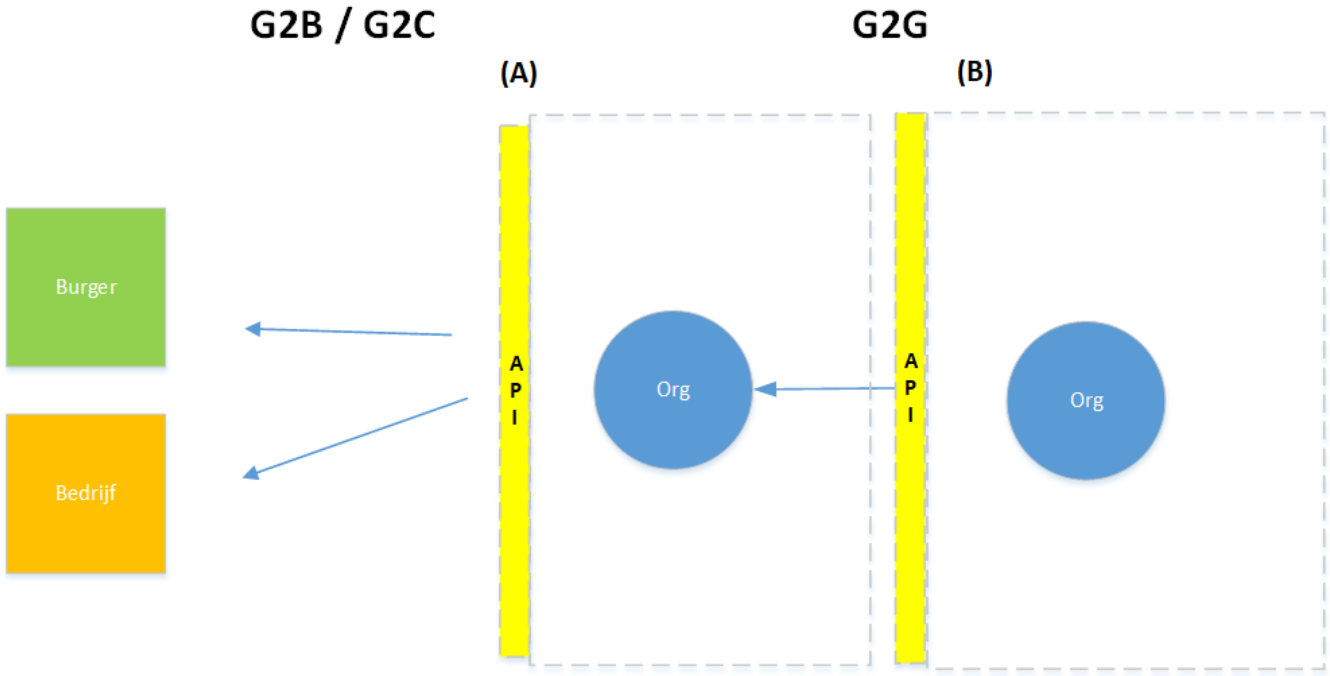
Figuur 1: De plaats van API's bij aanbod en gebruik van (gegevens)diensten;

Figuur 1 toont de plaats van API's in de gegevensuitwisseling en relevante onderwerpen in deze context. Bij de Dienst afnemer speelt het gebruik van API's, bij de Dienst aanbieder speelt het aanbieden van API's.

In dit hoofdstuk wordt specifiek ingegaan op de 'aanbod kant': het onderdeel 'Diensten toegang' in het schema.

5.1.1 API's voor Burgers, Bedrijven en Overheden

Overheidsorganisaties bieden diensten aan Burgers, Bedrijven en andere Overheidsorganisaties. Onderstaande figuur geeft de dienstverlening middels API's aan de verschillende partijen grafisch weer.



Figuur 2: API Diensten voor Burgers, Bedrijven en Overheden

Toelichting

- (A) : API Contactoppervlak Overheid naar Burgers en Bedrijven
- (B) : API Contactoppervlak Overheid naar Overheid (bv onderdelen GDI)
- G2C : Government 2 Citizen (Overheid naar Burger)
- G2B : Government 2 Business (Overheid naar Bedrijf)
- G2G : Government 2 Government (Overheid naar Overheid)

5.1.2 Belang van API Standaardisatie

In de bovenstaande figuur wordt ook het belang van standaardisatie van API's zichtbaar:

Burgers, Bedrijven (en ook Overheidsorganisaties zelf) gebruiken (doorgaans) de API's van meerdere Overheidsorganisaties. Wanneer de verschillende 'API contactoppervlakken' uniform zijn (ook over organisaties heen) kunnen dienstafnemers gemakkelijker (en dus sneller en met minder kosten) gebruik maken van 'Overheids API's'

5.2 Typologie van API's

In de volgende paragrafen worden verschillende indelingen voor API's behandeld. Deze indelingen helpen om in de API architectuur van een organisatie verschillende soorten API's te benoemen.

5.2.1 Intern / Extern, Open / Gesloten

Een organisatie heeft verschillende soorten API's:

Extern:

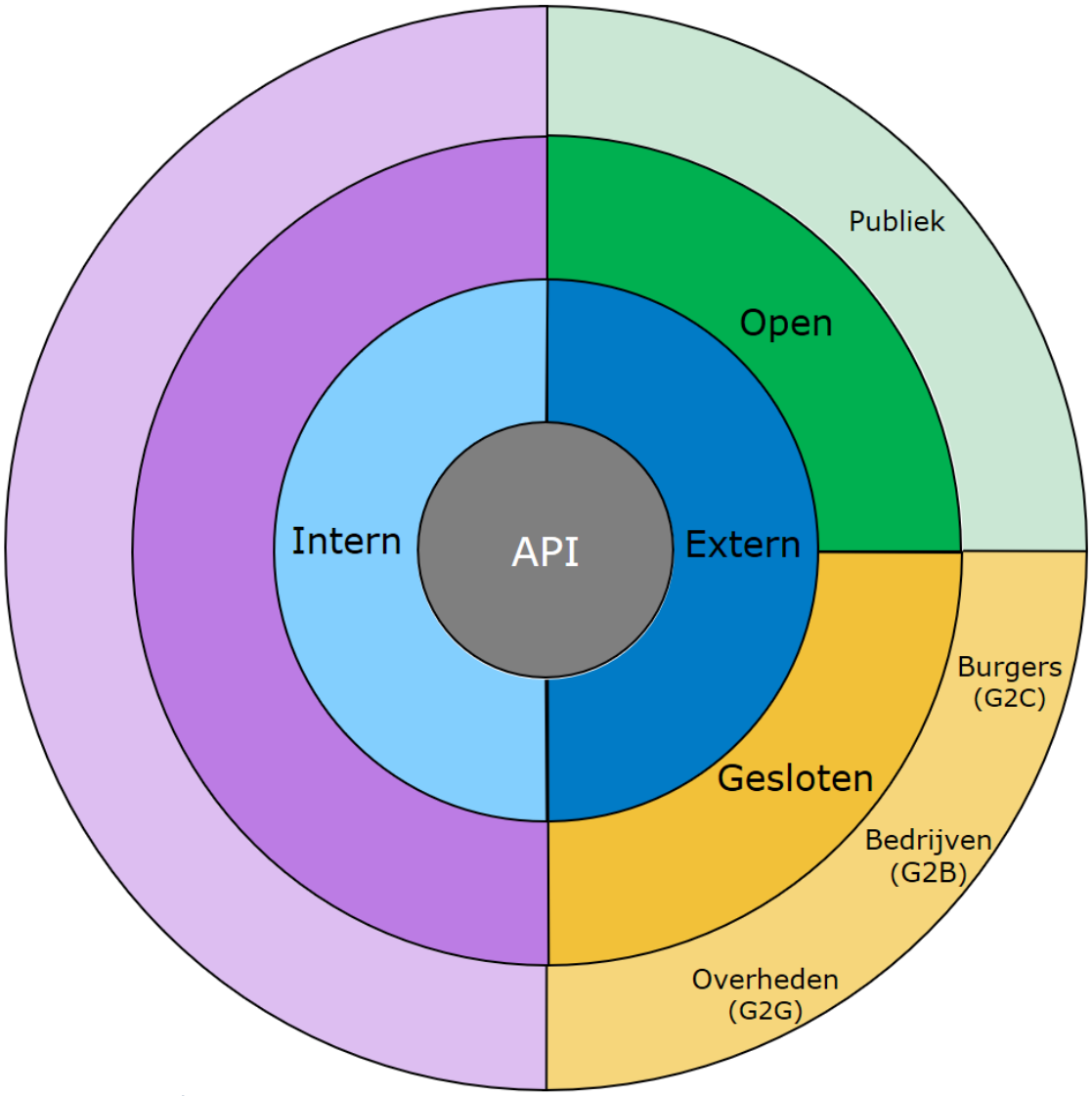
- Open API's: voor ontsluiten van diensten zonder toegangsbeperking bv open data.
- Gesloten API's: voor ontsluiten van diensten met toegangsbeperking bv persoonsgegevens en vertrouwelijke gegevens of diensten voor specifieke partijen.

Intern:

- Interne API's : voor ontsluiten van diensten binnen de organisatie

Een overheidsorganisatie ontsluit zijn diensten naar andere overheidsorganisaties, naar bedrijven/private organisaties en naar burgers.

In onderstaande figuur wordt dit visueel weergegeven.



Figuur 3 : Soorten API's

Afkortingen:

- G2C : Government 2 Citizen (Overheid naar Burger)
- G2B : Government 2 Business (Overheid naar Bedrijf)
- G2G : Government 2 Government (Overheid naar Overheid)

5.2.2 Systeem, Proces, Convenience

Deze indeling maakt onderscheid in Systeem, Proces, Convenience API's:

- System API (werkt op het niveau van de databron);

- Process API (doet aan orchestration door één of meerdere System API's aan te roepen);
- Convenience of Experience API (beantwoord één specifieke gebruikersvraag);

5.2.3 Business / Exposure

Deze indeling maakt onderscheid in Business en Exposure API's:

Business API definitie:

Een API die beschreven is in business termen , gebruik maakt van generieke modellen en welke bedrijfsprocessen ondersteund. (Hierbij wordt vermeden om terminology en payloads te gebruiken die specifiek zijn voor 3rd party software om afhankelijkheid van specifieke infrastructuur te voorkomen en te focussen op bedrijfsdoelstellingen;

(Bijvoorbeeld API’s gebaseerd op de Generieke Functies (capabilities) van de Nederlandse Overheid, zie https://www.noraonline.nl/wiki/Generieke_functies, en op de capabilities van de eigen organisatie).

Exposure API

Een API die toegang geeft tot de (basis)functionaliteit en data van een (specifiek) systeem

5.2.4 Philosophy, Protocol, Encoding

Deze indeling gaat uit van de technische aspecten van de API:

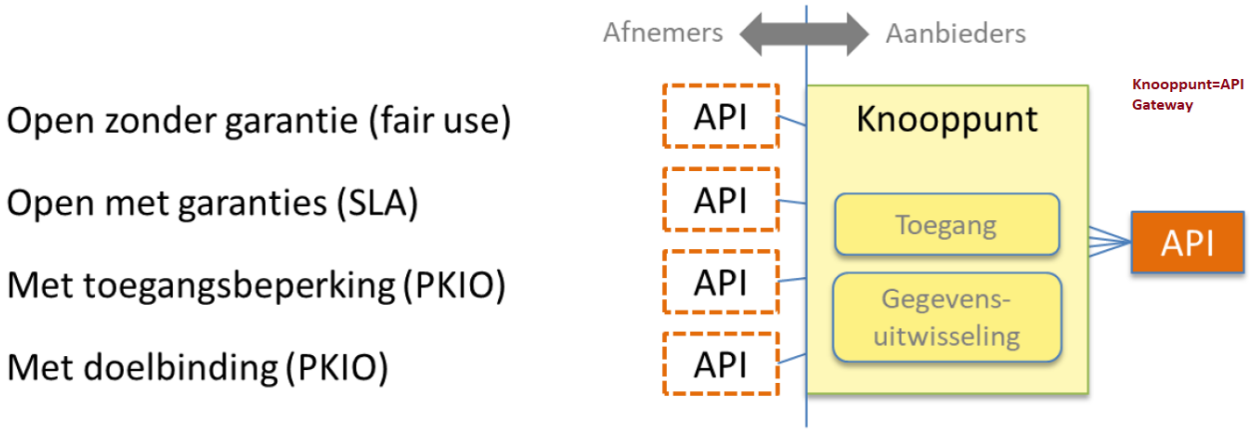
- Design Philosophy (eg RESTful, GraphQL)
- Communications Protocol (eg HTTP, Websockets)
- Encoding (eg JSON, Protobuf (binary))

5.2.5 API Virtualisatie

Een andere mogelijke indeling is op basis van de 'virtuele' varianten van een API: Een en dezelfde API kan in verschillende smaken worden aangeboden. In onderstaand voorbeeld is dit:

- Open zonder garantie

- Open met garanties
- Met toegangsbeperking
- Met doelbinding



5.2.6 API Language styles

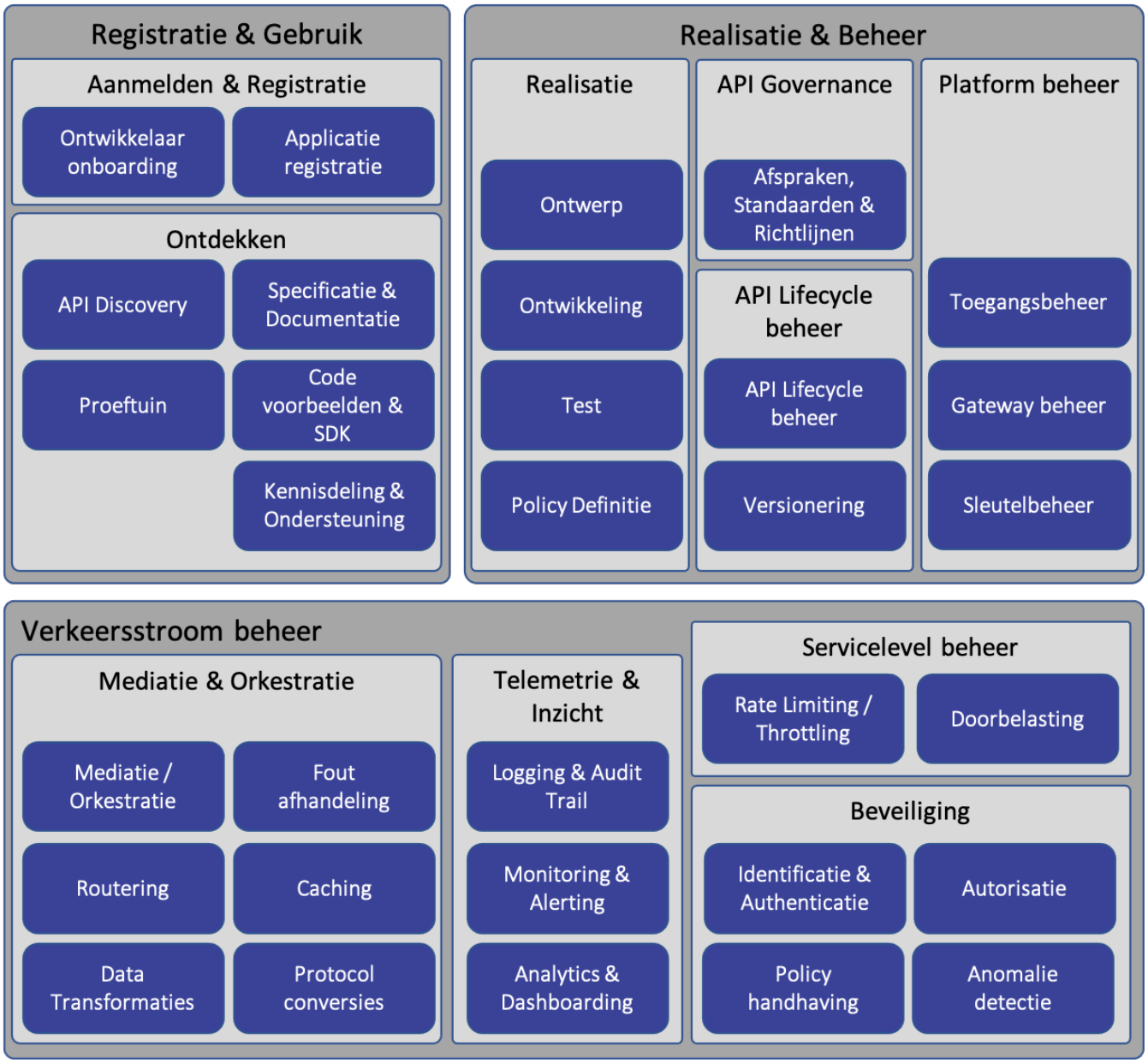
Onderstaande indeling gaat uit van de bij de API toegepaste 'language' style.

- Tunnel Style: XML-RPC, SOAP, gRPC, Avro
- Resource Style: OpenAPI/Swagger, RAML, API Blueprint
- Hypermedia Style: HAL, Siren, Atom, HATEOAS
- Query Style: GraphQL, OData, SPARQL
- Event-based Style: MQ, WebSub, MQTT, XMPP, AMQP, Kafka, AsyncAPI

5.3 API Capability Model

Om als organisatie API's aan te bieden aan andere partijen op een gecontroleerde en beheersbare manier moet je bepaalde functionaliteit bieden, processen inregelen en ondersteuning aanbieden.

Het onderstaande API Capability model geeft weer aan welke onderwerpen men aandacht moet schenken bij het inrichten van API gedreven dienstverlening. De onderwerpen zijn gegroepeerd in de categorieën *Registratie & gebruik*, *Realisatie & beheer* en *Verkeersstroom beheer*.



5.3.1 Registratie & Gebruik

Deze categorie bevat capabilities voor het ondersteunen van ontwikkelaars die gebruik willen maken van de aangeboden API's. De capabilities binnen deze categorie zijn onderverdeeld in de volgende sub-categorieën:

- *Aanmelden & Registratie*: functionaliteiten waarmee nieuwe gebruikers zichzelf en hun applicaties kunnen registreren om gebruik te maken van aangeboden API's;
- *Ontdekken*: functionaliteiten waarmee gebruikers kennis over aangeboden API's op kunnen doen en delen.

Over het algemeen biedt een API leverancier capabilities ten behoeve van API gebruik in een ontwikkelaarsportaal.

5.3.1.1 Ontwikkelaar onboarding

Voordat een ontwikkelaar applicaties kan ontwikkelen die gebruik maken van aangeboden API's, moet deze ontwikkelaar zich registreren bij de API aanbieder. Hierbij is het belangrijk dat een goede balans wordt gezocht tussen het bieden van een zo eenvoudig en snel mogelijk onboarding proces (bij voorkeur self-service) en het borgen van afspraken voor het gebruik van de API's door de ontwikkelaar, bijvoorbeeld door het ondertekenen van gebruikersvoorwaarden en het valideren van de rechtmatigheid van de ontwikkelaar.

Bij ontwikkelaar onboarding moet ook worden gedacht aan het zich kunnen afmelden van ontwikkelaars, of het afsluiten van ontwikkelaars in het geval gebruikersvoorwaarden worden overschreden.

5.3.1.2 Applicatie registratie

Applicaties die als client gebruik willen maken van API's moeten eerst worden aangemeld bij de API aanbieder. Alleen aangemelde applicaties ontvangen credentials om de API's te bevragen.

Applicatie registratie omvat de processen rondom het registreren van applicaties en het kunnen beheren van geregistreerde applicaties, inclusief de nodige controles die hiervoor moeten worden uitgevoerd. Bij het kunnen beheren kan bijvoorbeeld worden gedacht aan het toekennen of afnemen van privileges door de API aanbieder of het kunnen inzien van informatie over de API Client. Voorbeelden van benodigde controles is het controleren dat de juiste toegang en grant types worden aangevraagd voor het doel van de applicatie en of een applicatie de benodigde tests heeft doorstaan voordat deze toegang kan krijgen tot productie data.

Ook de uitgifte van credentials voor applicaties is onderdeel van applicatie registratie.

5.3.1.3 API Discovery

Aangeboden API's moeten vindbaar zijn voor partijen die er gebruik van willen maken. API Discovery richt zich op deze vindbaarheid van API's.

API Discovery kan op verschillende manieren toegepast worden. Allereerst kan API Discovery worden toegepast door een API aan te bieden welke referenties biedt naar de verschillende API's of kan de lijst met aangeboden API's in developer documentatie worden opgenomen. De op de 'Pas toe of leg uit-lijst' van het Forum Standaardisatie opgenomen OpenAPI 3 (OAS 3) specificatie voorziet in deze beide manieren.

Ook bestaan er zoekmachines die aangeboden API's op een centrale locatie toegankelijk maken, waardoor de vindbaarheid van API's nog meer wordt vergroot. Een voorbeeld van een dergelijke zoekmachine die wordt gebruikt binnen de overheid is developer.overheid.nl.

5.3.1.4 Specificatie & Documentatie

Traditioneel worden webservice endpoints beschreven middels statische documentatie welke (bijvoorbeeld als PDF document) beschikbaar wordt gesteld aan aansluitende partijen. Door de dynamische aard van API's en de mogelijkheden die door tools worden geboden, is het bij API's gebruikelijk documentatie dynamisch aan te bieden. Hierbij is het mogelijk om direct vanuit de documentatie API verzoeken op te stellen en uit te voeren.

API specificaties op basis van OpenAPI 3 bieden de mogelijkheid om dynamische documentatie te genereren. Hierbij moet wel de kanttekening worden gemaakt dat OpenAPI 3 specificaties zich primair richten op documentatie van API endpoints en de structuur daarvan. Mogelijke combinaties van inputs en outputs, use cases en overige documentatie die niet direct over de structuur van de API endpoints gaat wordt slechts beperkt ondersteund en dient buiten de OpenAPI specificatie te worden toegevoegd.

5.3.1.5 Proeftuin / Sandbox

Traditioneel worden webservice client implementaties op basis van documentatie en wsdl definities ontwikkeld, waarbij een volledig ontwikkelde client applicatie toegang krijgt tot een preproductie omgeving om de implementatie te toetsen.

Bij API Client ontwikkeling is het gebruikerlijk om, naast het gebruik van dynamische documentatie, een proeftuin aan te bieden in de vorm van sandbox API's. Dit zijn API's welke al vanaf de start van de app ontwikkeling (of zelfs daarvoor, vanuit de dynamische documentatie) natuurgetrouwe API verzoeken kunnen worden uitgevoerd. Hierdoor krijgen app ontwikkelaars direct feedback over hoe hun applicaties aansluiten op de geboden API's.

Bij het aanbieden van een proeftuin is het goed om na te gaan hoe natuurgetrouw deze moet zijn. Uitersten zijn enerzijds een stub met daarin een aantal basis Use Cases uitgewerkt en anderzijds een proefomgeving met een kleine subset aan productie-like geanonimiseerde data welke al dan niet per client gescheiden is en regelmatig wordt ververst indien data vanuit de sandbox aanpasbaar is.

5.3.1.6 Code voorbeelden & SDK

Naast het aanbieden van dynamische documentatie en een proeftuin, kunnen ontwikkelaars erg geholpen zijn wanneer de API aanbieder code voorbeelden, referentie implementaties of SDK's aanbiedt in verschillende programmeertalen. Het gebruik van de OpenAPI 3 specificatie voor API documentatie heeft als voordeel dat er al veel tooling bestaat om SDK's te genereren op basis van de API specificatie.

5.3.1.7 Kennisdeling & Ondersteuning

Naast het kunnen inzien van documentatie en het gebruik maken van sandbox API's, is het belangrijk dat ontwikkelaars van API Clients terecht kunnen met vragen of opmerkingen. Hierbij kan in het meest rudimentaire geval bijvoorbeeld worden gedacht aan een feedback formulier en een veelgestelde vragen pagina, maar ook aan een forum waarbij gebruikers elkaar kunnen helpen en de aanwezigheid van de API aanbieder op publieke fora (bijvoorbeeld Stack Overflow of Super User), social media (bijvoorbeeld Twitter) of invite-only discussie platformen (bijvoorbeeld Discord of Slack).

5.3.2 Realisatie & Beheer

Realisatie & Beheer gaat over het (door)ontwikkelen van API's, het beheren van de API lifecycle van ideatie tot uitfasering en het beheren van het API ecosysteem. De capabilities binnen deze categorie zijn onderverdeeld in de volgende sub-categorieën:

- *Realisatie*: capabilities met betrekking tot het aanbieden van nieuwe API endpoints en de doorontwikkeling van bestaande;
- *API Governance*: standaarden, afspraken en richtlijnen rondom het aanbieden van API's en de processen om hieraan te voldoen;

- *API Lifecycle beheer*: capabilities met betrekking tot de sturing op een uniform API portfolio en lifecycle management van de API's;
- *Platform beheer*: functionaliteiten met betrekking tot het kunnen beheren van het platform of ecosysteem waarin de API's opereren.

5.3.2.1 Ontwerp

Waar traditioneel webservices technisch werden ingestoken en veelal werden gebaseerd op de technische inrichting van het achterliggende IT landschap, is het bieden van een uitmuntende developer experience (DX) één van de belangrijkste doelen bij het aanbieden van API's.

Goed API ontwerp ligt daarbij aan de grondslag. Net als User Experience (UX) ontwerp zich richt op het bieden van de ultieme gebruikerservaring bij websites, worden API's ontworpen met de gebruikers (ontwikkelaars) centraal, in plaats van gebaseerd op de inrichting van back-ends. De API's fungeren daarbij als façade en maskeren de complexiteit van het achterlandschap voor gebruikers van de API's, zonder in te hoeven boeten op functionaliteit.

Aangezien het ontwerp van API's net als bij web-ontwerp een creatief proces is en er veel mogelijkheden zijn om dezelfde uitdagingen het hoofd te bieden, is er een lijst met API Design Rules opgenomen op de ‘Pas toe of leg uit-lijst’ van het Forum Standaardisatie opgenomen, welke API ontwerpers helpt goede ontwerpbeslissingen te maken op basis van best-practices in API ontwerp.

5.3.2.2 Ontwikkeling

API ontwikkeling betreft het ontwikkelen van API endpoints op basis van API ontwerpen. Bij het inrichten van API ontwikkeling moeten een aantal keuzen worden gemaakt, waaronder waar API ontwikkeling plaatsvindt.

Een optie kan zijn dat API ontwikkeling in de API Gateway plaatsvindt; de meeste API Gateway producten bieden connectors om verschillende back-end systemen te ontsluiten, policies om protocollen te vertalen (bv SOAP <-> REST) en de mogelijkheid om API endpoints op basis van OAS 3 specificaties te genereren. Een nadeel van API ontwikkeling in de API Gateway is dat het vaak specifieke productkennis vergt en daardoor de afhankelijkheid van experts van de gekozen API Gateway vergroot.

Alternatieven zijn API ontwikkeling in microservices, waarbij microservices API's aanbieden waarvan een deel via de API Gateway as-is wordt ontsloten, of het gebruik van een middleware oplossing. Hierbij moet er rekening mee worden gehouden dat het gebruik van middleware dezelfde nadelen heeft met betrekking tot specifieke productkennis als het ontwikkelen van API's in de API Gateway, maar kan het zijn dat deze kennis reeds in de teams aanwezig is.

Wanneer bij het API Ontwerp gebruik gemaakt wordt van de OAS 3 standaard, bieden zowel API Gateway oplossingen en programmeertalen vaak goede ondersteuning voor het automatisch genereren van code op basis van de API specificatie, wat de snelheid van ontwikkeling vergroot en de foutgevoeligheid verkleint. Hierbij moet worden opgemerkt dat enkele OAS3 constructies (nog) niet kunnen worden gebruikt, omdat deze fouten geven bij code generatie.

5.3.2.3 Test

Dit betreft het testen van API endpoints tegen de specificatie. Waar mogelijk gebeurt dit geautomatiseerd. Op het gebied van *syntactische* tests biedt veel test tooling ondersteuning voor het importeren van OpenAPI 3 specificaties; voor het uitvoeren van *semantische* tests is het vaak noodzakelijk dat testdata wordt geprepareerd en use cases worden uitgewerkt in test cases.

5.3.2.4 Policy Definitie

Dit betreft het definiëren van generieke API policies, welke worden toegepast op alle API's of door een aantal API's kunnen worden uitgevoerd. Voorbeelden van API policies zijn:

- Validatie van API verzoeken tegen een API specificatie;
- het vertalen van JSON berichten naar een ander berichtformaat op basis van content negotiation;
- het vertalen van (delen van) berichten;
- het filteren van headers in response berichten;
- het 'throttlen' van inkomende API verzoeken om back-ends te ontlasten;
- het afdwingen van gautoriseerde API verzoeken;
- etc.

5.3.2.5 Afspraken, Standaarden en Richtlijnen

Een set van afspraken, standaarden en richtlijnen om ervoor te zorgen dat een uniforme set van API's wordt aangeboden, dat aansluit bij de API strategie van de API aanbiedende organisatie (en breder, in het geval van organisatie-overstijgende API's). Hieronder vallen ook processen om ervoor te zorgen dat aan deze afspraken, standaarden en richtlijnen wordt voldaan en om organisaties de mogelijkheid te bieden gecreëerde afspraken en richtlijnen te verheffen naar landelijk niveau.

5.3.2.6 Governance policy handhaving

Dit betreffen policies die waarborgen dat de juiste stakeholders bij business transacties betrokken worden. Voorbeelden van business transacties zijn:

- Het wijzigen van een policy voor een bestaande API-versie;
- Het updaten van documentatie van een API;
- Het veranderen van de status in de levenscyclus van een API-versie;
- Het abonneren op een API.

5.3.2.7 API Lifecycle beheer

Dit omvat functionaliteiten rondom het beschikbaar maken van nieuwe API endpoints, bijvoorbeeld door ze in een API Gateway te registreren, het maken van non-breaking changes en het uiteindelijk uitfaseren van API endpoints omdat deze niet meer nodig zijn of door een nieuwere API versie worden vervangen.

Het is belangrijk gebruikers van API's duidelijk inzicht te geven in de processen en tijdslijnen rondom lifecycle management, zodat zij zich goed op eventuele wijzigingen voor kunnen bereiden. Voorbeelden van transparante communicatie hierover zijn het al bij beschikbaarstelling van API endpoints definieren en communiceren van deprecation en sunset datums. Deze kunnen bijvoorbeeld in de API documentatie en/of API headers worden gecommuniceerd. In de periode tussen deprecation van een API worden veelal alleen essentiele updates doorgevoerd en worden nieuwe functionaliteiten alleen toegevoegd op nieuwe API versies.

5.3.2.8 Versionering

Versionering van API's zorgt ervoor dat het voor afnemers van API's duidelijk is op welke API wordt aangesloten en welke eigenschappen daarbij horen. Voor aanbieders van API's zorgt het ervoor dat ze wendbaar kunnen zijn door nieuwe API versies te introduceren, zonder dat dit direct impact heeft op API clients.

Bij het toepassen van API versionering hoort ook het duidelijk inzichtelijk maken van hoe lang een specifieke API versie ondersteund wordt, wanneer een nieuwe versie wordt geintroduceerd en hoe de overgangsfase eruit ziet.

De tendens bij API's op het gebied van versionering is om zeer terughoudend te zijn met het introduceren van nieuwe major versies door, welke nodig zijn bij het maken van breaking changes. Om toch wendbaar te kunnen zijn, wordt veelal semantic versioning (semver) toegepast en worden API's aangepast middels niet-breaking minor changes en patches, totdat er een noodzaak is om breaking changes, en daarmee een major versie upgrade, uit te voeren.

5.3.2.9 Toegangsbeheer

Het registreren en beheren van API endpoints gebeurt zo veel mogelijk op basis van self-service door teams die API's ontwikkelen. Daarnaast worden centrale policies en API Gateway configuratie vaak centraal uitgevoerd.

Hiervoor is het belangrijk dat het API & Gateway beheer op basis van gebruikers en een autorisatie matrix verloopt.

5.3.2.10 Gateway beheer

Hieronder vallen alle werkzaamheden met betrekking tot het beheren van de API Gateway zelf. Hierbij kan worden gedacht aan de technische inrichting van het platform, het uitvoeren van upgrades, het beheren van gebruikers, rollen en autorisaties en het oplossen van problemen die volgen uit monitoring en alerting.

5.3.2.11 Sleutelbeheer

Voor een veilige uitwisseling van gegevens tussen API Gateway en client (en eventuele externe Authorisatie Servers) spelen ondertekening en versleuteling van berichten een belangrijke rol.

Hiervoor is het belangrijk dat de API aanbieder het sleutelmateriaal dat hiervoor wordt gebruikt op een veilige manier kan opslaan en waar nodig kan distribueren naar aansluitende API clients.

5.3.3 Verkeersstroom beheer

Verkeersstroom beheer gaat over de functionaliteiten die te maken hebben met verwerken van daadwerkelijk verkeer tussen de API client en aanbieder en het verwerken daarvan binnen het API landschap.

De capabilities binnen deze categorie zijn onderverdeeld in de volgende sub-categorieën:

- *Mediatie & Orkestratie*: het verwerken, valideren, routeren en bewerken van API verzoeken en antwoorden om een gebruiksvriendelijke en uniforme beleving aan API clients te kunnen bieden;
- *Telemetrie & Inzicht*: het meten en inzichtelijk maken van API verzoeken en antwoorden om inzicht te krijgen in het gebruik van API's en de gezondheid van het platform;
- *Servicelevel Beheer*: het definieren en afdwingen van afspraken rondom beschikbaarheid en de toegestane hoeveelheid API verzoeken dat een API client mag doen, eventueel gebaseerd op staffels voor doorbelasting;
- *Beveiliging*: functionaliteiten rondom de beveiliging van API endpoints en de data en processen die hiermee ontsloten worden.

5.3.3.1 Mediatie & Orkestratie

In het geval back-end systemen niet de mogelijkheden hebben te communiceren volgens de gespecificeerde API contracten of als een API betrekking heeft op meerdere back-end systemen, kan het noodzakelijk zijn data transformaties toe te passen om toch de gebruiksvriendelijke API's aan afnemers te kunnen bieden. In dit geval wordt API Mediatie, ook wel data transformatie of orkestratie, toegepast met als doel de API's die aan API Clients geboden worden zo gebruiksvriendelijk mogelijk te laten zijn, zonder afhankelijk te zijn van implementatiedetails en technologische beperkingen van back-end systemen. API Mediatie kan plaatsvinden in microservices, een integratielaag in de back-end of in de API Gateway.

5.3.3.2 Routering

In het geval de back-end functionaliteit wordt ingevuld door meerdere back-end systemen of microservices, dienen inkomende API verzoeken te worden gerouteerd naar de juiste systemen. In het eenvoudigste geval kan dit betekenen dat verzoeken voor verschillende API endpoints naar verschillende back-end systemen worden gerouteerd, maar in ingewikkeldere situaties kan het ook noodzakelijk zijn te routeren op basis van informatie uit het verzoek, zoals bijvoorbeeld afzender of inhoud.

5.3.3.3 Data Transformaties

Dit betreft functionaliteiten rondom het vertalen van API endpoints naar het achterliggende datamodel. Dit kan plaatsvinden in de API Gateway, integratietoepassingen in de back-end of in back-end (micro)services.

Data transformaties worden, wanneer benodigd, bij voorkeur lokaal in back-end (micro)services toegepast. Niet-domein-specifieke transformaties (bijvoorbeeld voor het vertalen tussen interne en externe representaties of het digitaal ondertekenen van berichten) kunnen ook eventueel centraal in de API Gateway plaatsvinden.

5.3.3.4 Foutafhandeling

In het geval dat er fouten optreden in de API's, bijvoorbeeld door niet-functionerende back-end systemen, moeten API Clients worden voorzien van duidelijke foutmeldingen en moeten betrokkenen bij de API aanbieder voorzien worden van voldoende informatie om de oorzaak van de fouten op te kunnen lossen en te verhelpen, indien nodig.

5.3.3.5 Caching

Een belangrijk onderdeel van developer experience bij API's is de performance ervan. Caching zorgt ervoor dat niet voor elke aanroep het achterlandschap geraadpleegd hoeft te worden. RESTful API's lenen zich zeer goed voor het toepassen van caching door toepassing van het Proxy architectuur patroon, echter dient zorg te worden gedragen dat gewijzigde broninformatie leidt tot invalidatie van de informatie in de cache, zodat API Clients geen achterhaalde informatie ophalen.

5.3.3.6 Protocol conversie

Om operabiliteit te vergroten zodat meer API Clients aan kunnen sluiten, kan het nuttig zijn API's aan te bieden in andere protocollen en architectuurstijlen dan REST / JSON. Zonder specifieke code in back-end systemen kan bijvoorbeeld de API Gateway API's op basis van REST en JSON omvormen naar API's op basis van SOAP en XML. Hierdoor kunnen API Clients die geen JSON ondersteunen worden bediend, zonder dat dit leidt tot extra ontwikkel effort aan de back-end / microservices kant (het is overigens meestal wel benodigd de API Gateway van instructies te voorzien voor de conversie, bijvoorbeeld middels XSLT specificaties).

Ook kan protocol conversie nodig zijn om back-end systemen die gebruik maken van verouderde protocollen toch via API's aan te bieden.

5.3.3.7 Logging & Audit trail

Geeft inzicht in de verkeersstromen tussen API Clients, API Gateway en het achterliggende applicatie landschap en zorgt ervoor dat alle handelingen herleidbaar zijn door inzicht te geven in historische verkeersstromen, ten behoeve van audit doeleinden (i.h.k.v. privacy, beveiliging en transparantie).

Onder andere de volgende informatie kan inzichtelijk worden gemaakt:

- Wie heeft data geraadpleegd?
- Welke data is geraadpleegd?
- Waarom de data is geraadpleegd? Doelbinding wordt vastgelegd.
- Wanneer de data is geraadpleegd
- Hoe de data is geraadpleegd?

5.3.3.8 Monitoring & Alerting

API Monitoring & Alerting geeft inzicht in de technische staat van de API's en informeert (via RCS, SMS en/of e-mail) betrokken groepen of personen indien bepaalde gebeurtenissen optreden. Voorbeelden van gebeurtenissen zijn onbeschikbaarheid van bepaalde gegevensverbindingen of certificaten die op korte termijn zullen verlopen.

5.3.3.9 Analytics & Dashboarding

Naast onder andere productvisie en input van afnemers is API Analytics een belangrijke aandrijver bij het bepalen van de richting van API doorontwikkeling. Op basis van actueel API gebruik kan het resultaat van Key Performance Indicators (KPI's) worden vastgesteld. Dit kan leiden tot inzicht in welke API functionaliteiten succesvol zijn en welke minder succesvol, en zodoende kan leiden tot aanscherping of aanpassing van de productvisie.

5.3.3.10 Rate limiting / throttling

Rate limiting (throttling) biedt bescherming tegen een bovenmatig aantal verzoeken die worden afgevuurd op de omgeving van de API aanbieder, waardoor er storingen op de backend systemen kunnen optreden. Dit kan bij reguliere bedrijfsuren het 'spitsuur' zijn waardoor deze beveiliging moet worden ingeschakeld. Rate limiting is ook een manier om een SLA af te dwingen op basis van een contract.

Rate limiting kan worden ingesteld op verschillende tijdseenheden, bijvoorbeeld het aantal toegestane verzoeken per uur of per maand, en er kunnen verschillende profielen of plannen worden aangemaakt waarin deze quota worden geconfigureerd. Of het toekennen van voorrang op de afhandeling van bepaalde verzoeken (requests) is mogelijk, bijvoorbeeld door bulk-processen voor een korte periode voorrang te verlenen boven andere verzoeken. Hierdoor is gecontroleerd traffic management mogelijk, waardoor eventuele verstoringen op een later moment door piekbelasting juist kunnen worden voorkomen. Een profiel of plan is te koppelen aan een specifieke API.

5.3.3.11 Doorbelasting

In sommige gevallen kan er voor gekozen worden om API gebruik door te belasten. Dit is voor overheden interessant om zo eventuele kosten door te belasten op de afzonderlijke organisaties, maar ook richting ketenpartners.

Er bestaan verschillende doorbelastingsmodellen, welke vallen onder de volgende hoofdcategorieën:

- *Vrij:* de API is vrij te gebruiken zonder kosten.
- *Betaald:* gebruikers van de API moeten betalen voor het gebruik ervan. Onder deze categorie vallen verschillende subcategorieën, zoals Freemium, Tiered, Pay-as-you-go en doorbelasting op basis van een vaste prijs per tijdseenheid.

- *Indirect:* gebruikers van de API betalen niet direct voor het gebruik ervan, maar dragen indirect bij aan het business model van de API aanbieder, waardoor API gebruik geld oplevert voor de API aanbieder.
- *Affiliate:* vergelijkbaar met indirect, behalve dat de API aanbieder gebruikers van API's betaalt om het gebruik van de API's zoveel mogelijk te stimuleren.

5.3.3.12 Identificatie & Authenticatie

Hieronder vallen de identificatie en authenticatie van API Clients, eindgebruikers en organisaties. Identificatie en authenticatie van eindgebruikers is overigens niet altijd nodig, bijvoorbeeld bij back-office en batchprocessen die niet in de context van een eindgebruiker werken.

5.3.3.13 Autorisatie

Dit betreft de autorisatie van API toegang op basis van de geauthentiseerde API Client end eindgebruiker.

Bij autorisaties op basis van API's wordt vaak onderscheid gemaakt tussen grofmazige, role based, en fijnmazig, domein-specifieke autorisatie. Hierbij worden grofmazige, role based, autorisaties vaak centraal ingeregeld, bijvoorbeeld middels de API Gateway. Dit geeft mede invulling aan de eisen die worden gesteld vanuit wet- en regelgeving, namelijk dat autorisatie is ingericht conform doelbinding.

Fijnmazige, domein specifieke, autorisatie wordt idealiter afgehandeld door de domein applicatie. Echter ondersteunen verschillende gebruikers niet de correcte manier van API's aanroepen en doelbindingsregisters ontbreken veelal nog (en soms bieden de bronhouders nog niet de benodigde API's). Ook is IAM in veel overheidsorganisaties niet in die mate op orde dat hierin volledig sturing aan gegeven kan worden. Fijnmazige autorisatie kan hierdoor eventueel nog niet door de backend worden afgehandeld. Een alternatief hiervoor is dat er per doelbinding een andere API wordt aangeboden vanuit de API Gateway of dat de integratielaag functionaliteit biedt om aan de fijnmazige autorisatie invulling te geven.

5.3.3.14 Policy handhaving

API Policy handhaving gaat over het toepassen van de ontwikkelde API policies op API endpoints. Hierbij kan onderscheid worden gemaakt tussen policies die altijd van toepassing zijn en policies die afhankelijk van de aangeroepen API of de afgesproken service levels worden toegepast. Voorbeelden van policies zijn syntactische en semantische validaties van inkomende en uitgaande API aanroepen, maar ook autorisatie en throttling worden veelal geïmplementeerd als verkeersstroom policies.

5.3.3.15 Anomalie detectie

Het detecteren van verdacht gebruik van API endpoints, bijvoorbeeld ten behoeve van het opsporen van fraude of malware.

5.4 API Management Functionaliteit

5.4.1 Inleiding

In dit hoofdstuk wordt aandacht besteed aan de positionering van API-Management binnen de informatie architectuur van een overheidsorganisatie. Daarbij wordt stilgestaan bij de functies die verschillende tools moeten invullen om API-Management goed op te kunnen zetten. De kaders en richtlijnen die in dit hoofdstuk worden benoemd zijn in lijn met landelijke geldende architectuurrichtlijnen (zoals NORA, GEMMA en Common Ground), hiermee kan dit document worden gezien als een referentiearchitectuur op het gebied van API-Management.

5.4.2 Informatie architectuur

Alle overheden hebben een uitdaging op het gebied van data integratie[1]. Hoe faciliteer je gegevensuitwisseling tussen bronnen en afnemers op een efficiënte en beheersbare manier die voldoet aan de eisen van wet- en regelgeving?

Een overheidsinstantie geeft hier het beste invulling aan door organisatiebreed een zogenaamde 'integratielaag' binnen de infrastructuur te positioneren. Dit kan je beschouwen als de gereedschapskist waarbinnen verschillende tools beschikbaar zijn voor data-integratie. Gezien de wereldwijde ontwikkelingen in het gebruik van API's, kan API-Management tooling hierbinnen niet ontbreken. Het is voor overheden essentiële functionaliteit om in lijn met de NORA te kunnen opereren (of specifieker in lijn met Common Ground).

Referentiecomponenten

API-Management tooling omvat de referentiecomponenten API-Gateway, API-Manager en een API-Portaal. Hieronder is beknopt beschreven wat overheidsorganisaties hieronder kunnen verstaan:

- API-Gateway: hiermee worden gegevensverbindingen technisch gefaciliteerd, beveiligd en gemonitord. Alleen de applicaties (afnemers en aanbieders van API's) gebruiken de gateway.

Een API-Gateway wordt ingezet als poort tot het achterliggende datalandschap.

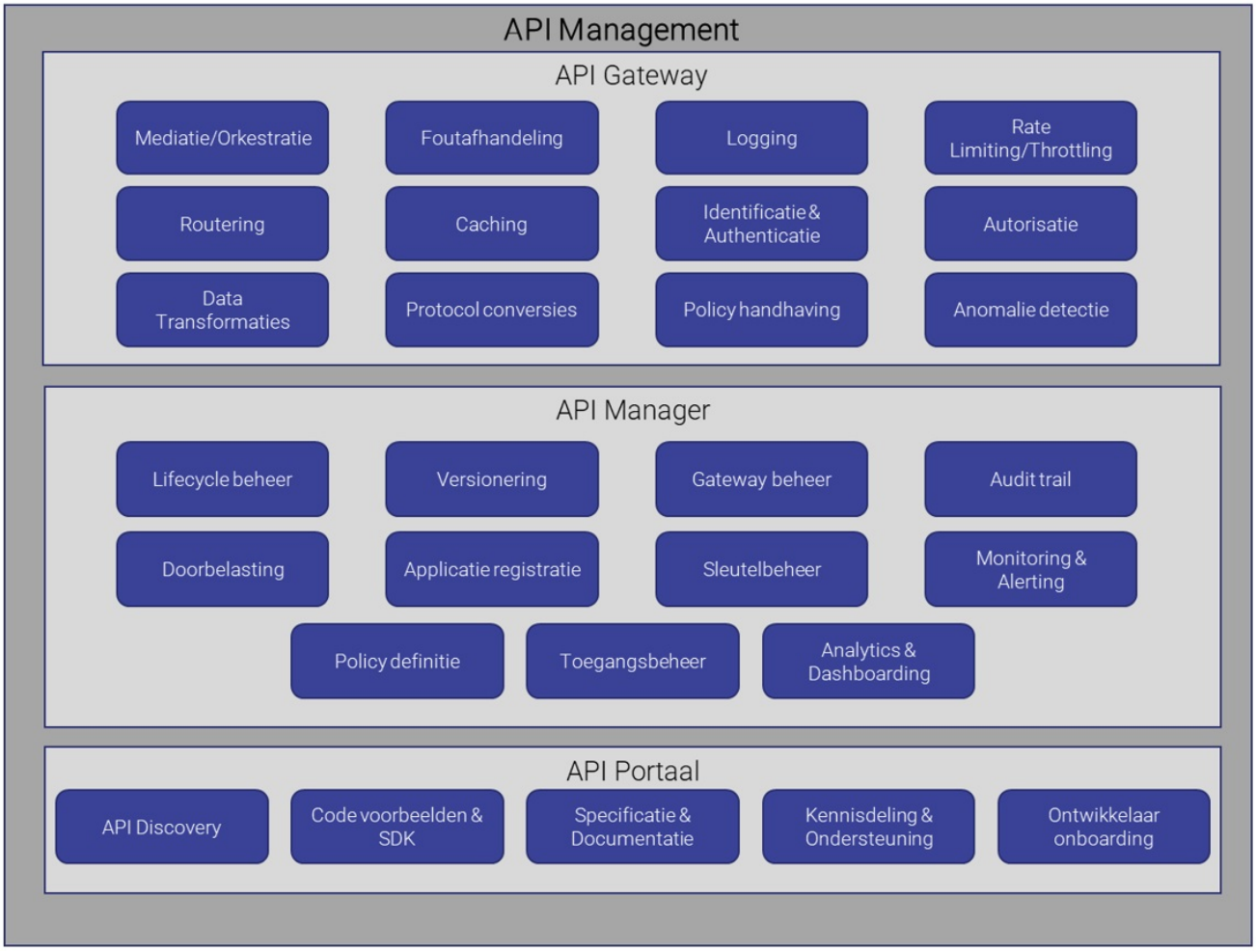
In cloud-native implementaties zie je steeds vaker een meer gedistribueerd model met behulp van micro gateways (als ingang voor iedere Cloud-omgeving) in plaats van een corporate API-Gateway die alles regelt, eventueel in combinatie met een service mesh[2]. Een hybride opstelling is ook mogelijk.

- API-Manager: zorgt voor de configuratie van de gateway en het beheer van de API's, op basis van patronen en zogenaamde policies.

- API-Portaal: een portaal waarin de aangeboden API's aan het brede publiek worden gepresenteerd. Via de specificaties kunnen ze een proeftuin creëren. Gebruikers zijn medewerkers geïnteresseerd in de ontwikkeling en het gebruik van API's. Dit kunnen zowel medewerkers van de overheid zijn, als ook externen (bijvoorbeeld van software leveranciers of ketenpartners).

Note: Landelijke ontwikkelingen op dit vlak i.r.t. developer.overheid.nl kunnen er voor zorgen dat deze functionaliteit op termijn centraal landelijk beschikbaar is.

In het API-Capability model zijn de functionaliteiten in detail beschreven. API-Capability model



Figuur 5: Functionaliteit binnen API-Management

Rol van servicebus

Binnen de integratielaag opereert veelal ook een organisatiebrede servicebus, vaak is er veel energie gestoken in het faciliteren van gegevensstromen via de servicebus (met name op basis van StUF[3]). De API-Management tooling komt naast de servicebus te staan en kan zo aanvullende functionaliteit bieden binnen de integratielaag. Het is voor overheden geen doel om bestaande verbindingen via de servicebus te elimineren of de servicebus uit te faseren.

Wel lijkt gezien de wereldwijde ontwikkelingen de aandacht van integratievraagstukken te gaan verschuiven van de inzet van een organisatiebrede servicebus naar een landschap waarin lightweight API-Management tooling een belangrijke rol speelt. Nieuwe verbindingen (met name voor het ophalen van data) zullen vaker gelegd gaan worden via enkel een API Gateway en de inzet van de servicebus wordt teruggedrongen. Enkel op het gebied waar de huidige servicebus specifieke toegevoegde waarde levert, wordt deze voor overheden nog ingezet voor nieuwe verbindingen (eventueel in combinatie met een API Gateway. Dit zal naar verwachting voor overheden de meest voor de hand liggende oplossing zijn, gezien het huidige applicatielandschap.

Toegevoegde waarde servicebus i.r.t. API-Gateway:

- StUF-koppelvlakken
- Complexe transformaties
- Orkestratie/logica
- Gegevensautorisatie op doelbinding (bij gemeentelijke servicebussen vaak geïntegreerd)

[Beschrijf welke product(hoofd)groepen worden veranderd, toegevoegd of verwijderd. Wat is de impact van het project op de geleverde producten en diensten?]

✧ ✧ ✧

[1] Verder aangeduid als 'integratie', van ook bijvoorbeeld informatie.

[2] Service mesh - Wikipedia

[3] Zie StUF Berichtenstandaard - GEMMA Online

✧ ✧ ✧

5.5 Informatiemodel & API

In dit hoofdstuk wordt ingegaan op de relatie tussen informatiemodel en API

5.5.1 Informatiemodellen

Een informatiemodel beschrijft een werkelijkheid. We onderscheiden vier niveaus variërend van een zo getrouw mogelijke beschrijving van die werkelijkheid tot een specificatie van de wijze van vastlegging van die werkelijkheid in een database of uitwisselformaat MIM NEN3610 :

Niveau 1 - Model van begrippen: een model van begrippen waarin de werkelijkheid wordt beschreven door middel van de daarin gehanteerde begrippen en de relaties tot elkaar.

Niveau 2 - Conceptueel informatiemodel: een conceptueel informatiemodel waarin de werkelijkheid wordt beschreven door middel van de informatie die voor dit domein relevant is,

onafhankelijk van het ontwerp en de implementatie in systemen. Het geeft een zo getrouw mogelijke beschrijving van die werkelijkheid en is in natuurlijke taal geformuleerd.

Niveau 3: - Logisch informatie- of gegevensmodel: een logisch informatie- of gegevensmodel waarin de werkelijkheid wordt beschreven door middel van de representatie van de informatie in de systemen en de uitwisseling tussen systemen en gebruikers. Een logisch informatiemodel is implementatie onafhankelijk en kan in meerdere technische modellen worden geïmplementeerd.

Niveau 4: - Fysiek of technisch gegevens- of datamodel: een fysiek of technisch gegevens- of datamodel waarin de werkelijkheid wordt beschreven door middel van de structuur en eigenschappen van de technologie die wordt gebruikt bij de opslag of uitwisseling. Een fysiek of technisch datamodel is afhankelijk van de gekozen techniek of tooling die wordt gebruikt en kan daarvoor geoptimaliseerd zijn.

(Zie ook NORA Informatielaag)

5.5.2 Definities

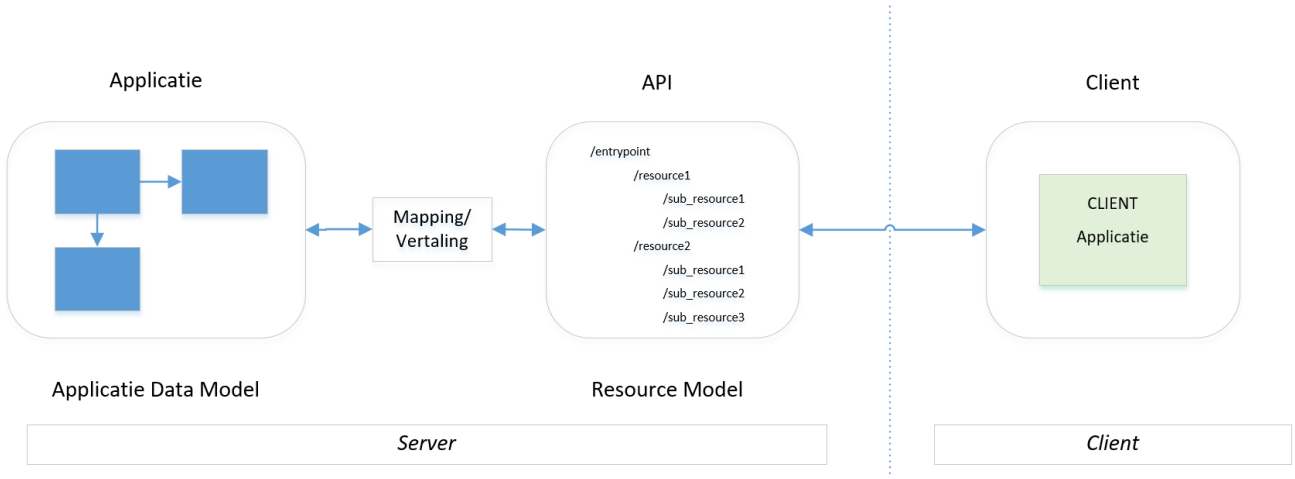
De volgende begrippen worden gehanteerd in dit hoofdstuk:

- *Informatiemodel* : Niveau 3 - Logisch gegevensmodel / datamodel
- *Resource model* : Niveau 4 - Fysiek / technisch datamodel (directe datamodel van een API)
- *Applicatie data model* : Niveau 4 - Fysiek / technisch datamodel (datamodel van een achterliggend systeem)

5.5.3 API, Informatiemodel en Resource model.

Via een API ontsluit een applicatie data en functionaliteit. Hierbij helpt het om onderscheid te maken tussen het 'interne' Applicatie Data Model en het Resource Model dat via de API wordt aangeboden.

Het Resource Model is als het ware een logische view op het achterliggende Data Model.



Figuur 6: Resource Model

Wanneer het Resource Model 1 op 1 gelijk is aan het achterliggende Data Model is de vertaling/mapping eenvoudig en kan dit ook geautomatiseerd worden : het Resource Model en de API kunnen bijvoorbeeld worden gegenereerd vanuit het Applicatie Data Model.

Ontkoppeling

Door Applicatie Data Model en Resource Model te ontkoppelen kunnen deze apart van elkaar evolueren. Dit heeft een aantal voordelen:

Breaking changes

Liefst wil je voorkomen dat een aanpassing aan het Applicatie Data Model welke verder geen waarde heeft voor de afnemers / clients van de data leidt tot verplichte aanpassingen/updates. Door te ontkoppelen kunnen breaking changes worden voorkomen voor bestaande clients;

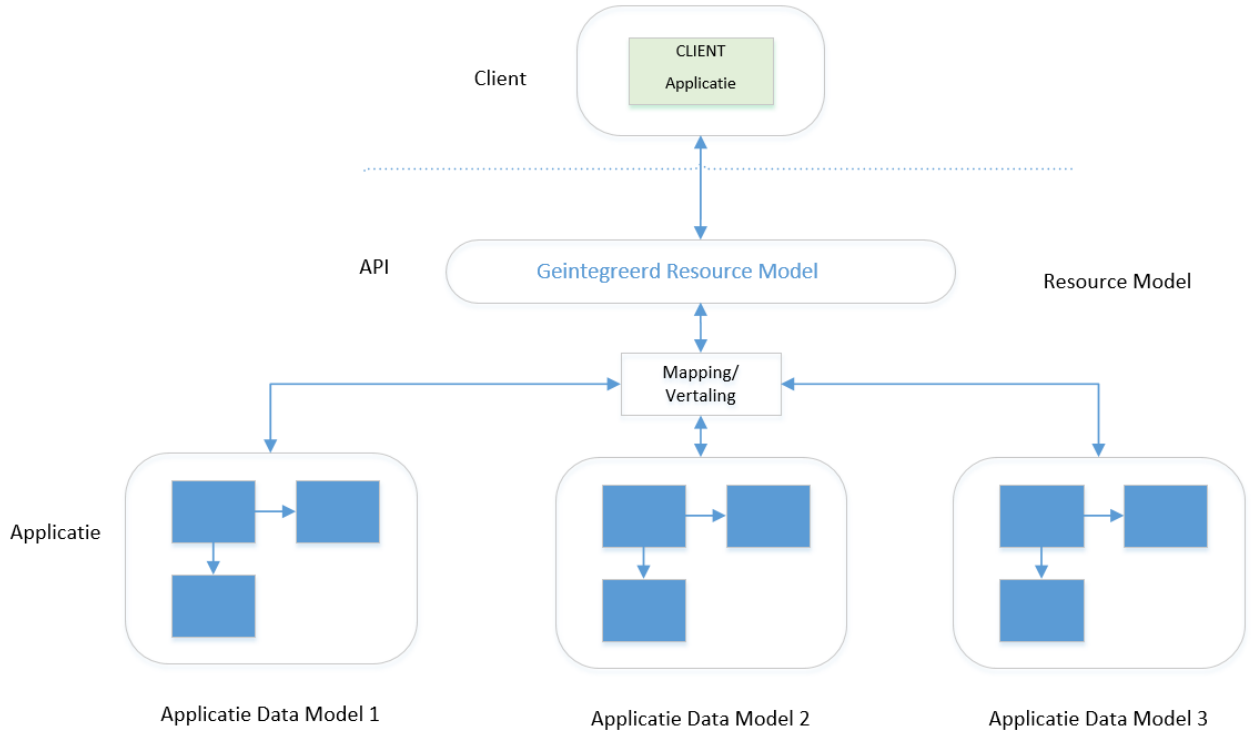
Verbergen complexiteit

Voor complexe Domeinen of gecombineerde Diensten waarbij meerdere bronnen worden gecombineerd is het waardevol om het Resource Model zo eenvoudig en duidelijk mogelijk te houden. In het Resource Model kan de achterliggende complexiteit verborgen blijven en kan de interface gebruikersvriendelijk worden gemaakt. Het Resource model is hier dan een abstraherende laag die alleen die zaken aanbiedt die de gebruiker nodig heeft en die aansluiten op de belevingswereld van de gebruiker.

Integreren & Innoveren

Het Resource Model als logische view op achterliggende datamodellen heeft ook als voordeel dat op de laag van het Resource Model al integratie van data modellen kan plaatsvinden nog voordat de achterliggende modellen zijn aangepast of volledig geïntegreerd. Een (nieuw) geïntegreerd

Resource model over meerdere achterliggende datamodellen heen kan zo databronnen integreren en aanbieden. Met behulp van een geïntegreerd resourcemodel op het niveau van API's kan sneller gestandaardiseerd worden en kan men ook sneller innoveren.



Figuur 7: Geïntegreerd Resource Model

5.5.4 Aanbevelingen

Verbindt het Resource Model met een Informatiemodel

Bij het aanbieden van data via een API is het van groot belang om de verbinding met een informatiemodel te hebben en deze verbinding ook te beschrijven en te publiceren bij de API documentatie.

- Dit bevordert begrip bij afnemers, zodat zij de ruwe data goed kunnen interpreteren.
- Dit houdt de API beheersbaar en zorgt dat de API gemakkelijker kan mee-evolueren met het informatielandschap (immers wijzigingen in het informatiemodel kunnen dan gerelateerd worden aan wijzigingen in de API).
- Wanneer een API informatie uit een stelsel van gegevensbronnen ontsluit bevordert dit interoperabiliteit in het stelsel.

Het Resource Model van een API is een (of mogelijk meerdere, afhankelijk van uitwisselformaat) Niveau 4 - Fysiek / technisch datamodel(len). Het is namelijk een model van de uitwisseling van gegevens in een concreet uitwisselformaat. Bijvoorbeeld gespecificeerd in een OAS document. OAS. Het is echter belangrijk om de verbinding met een Niveau 3: - Logisch informatie- of gegevensmodel uit te drukken, omdat dit een implementatie-onafhankelijk model is, wat begrip, maar ook interoperabiliteit, bevordert. Het bevordert interoperabiliteit met andere Niveau 3 informatiemodellen in een stelsel, maar biedt ook één overkoepelend informatiemodel wanneer er sprake is van gegevensuitwisseling conform verschillende Niveau 4 informatiemodellen gebaseerd op hetzelfde Niveau 3 informatiemodel.

5.6 API Security Architectuur

ICT beveiliging is over het algemeen gebaseerd op de aspecten *beschikbaarheid*, *integriteit* en *vertrouwelijkheid*. Dit hoofdstuk gaat allereerst in op deze drie aspecten en hun relaties met API beveiliging, waarna een aantal aan API beveiliging gerelateerde architectuurprincipes en architectuurpatronen zullen worden beschreven.

5.6.1 Beschikbaarheid

Beschikbaarheid gaat erover om te allen tijde bij informatie en informatiebronnen te kunnen en dat de beschikbaarheid van diensten voldoen aan gemaakte continuïteitsafspraken. Beschikbaarheid van gegevens en systeemfuncties wordt over het algemeen gegarandeerd door vermeerdering van systeemfuncties, door herstelbaarheid en beheersing van verwerkingen, door voorspelling van discontinuïteit en handhaving van de functionaliteit. Binnen de NORA is beschikbaarheid opgenomen als afgeleid principe AP41.

In de context van API's gaat beschikbaarheid erover dat consumenten van aangeboden API's juist worden geinformeerd over de afspraken omtrent (on)beschikbaarheid van de API's, dat de beschikbare capaciteit wordt verdeeld over de aangesloten API Clients en dat onvoorziene onbeschikbaarheid voor zowel aanbieders als consumenten van API's inzichtelijk wordt gemaakt, zodat daar juist op ingespeeld kan worden.

Aan beschikbaarheid gerelateerde API capabilities zijn *Caching*, *Rate limiting* / *Throttling*, *SLA Management*, *API Monitoring* / *Alerting* en *Foutafhandeling*. De onderstaande sectie *Componenten* beschrijft deze in meer detail en geeft aan waar deze worden toegepast in een API architectuur.

5.6.2 Integriteit

Integriteit gaat over het waarborgen van de integriteit van gegevens en systeemfuncties. Dit wordt over het algemeen bereikt door validatie en beheersing van gegevensverwerking en geautoriseerde toegang tot gegevens en systeemfuncties, door scheiding van systeemfuncties, door controle op communicatiegedrag en gegevensuitwisseling en door beperking van functionaliteit. Binnen de NORA is integriteit opgenomen als afgeleid principe AP42.

In de context van API's gaat integriteit over het versleutelen en ondertekenen van berichten en gegevens, het "tamper-proof" maken van API's (API Hardening) middels validatie van API verzoeken en de vastlegging van de gegevensuitwisseling tussen API aanbieders en consumenten.

Aan integriteit gerelateerde API capabilities zijn *Logging / Audit Trail*, *Policy Enforcement*, *Identificatie / Authenticatie / Autorisatie* en *Sleutelbeheer*. De onderstaande sectie *Componenten* beschrijft deze in meer detail en geeft aan waar deze worden toegepast in een API architectuur.

5.6.3 Vertrouwelijkheid

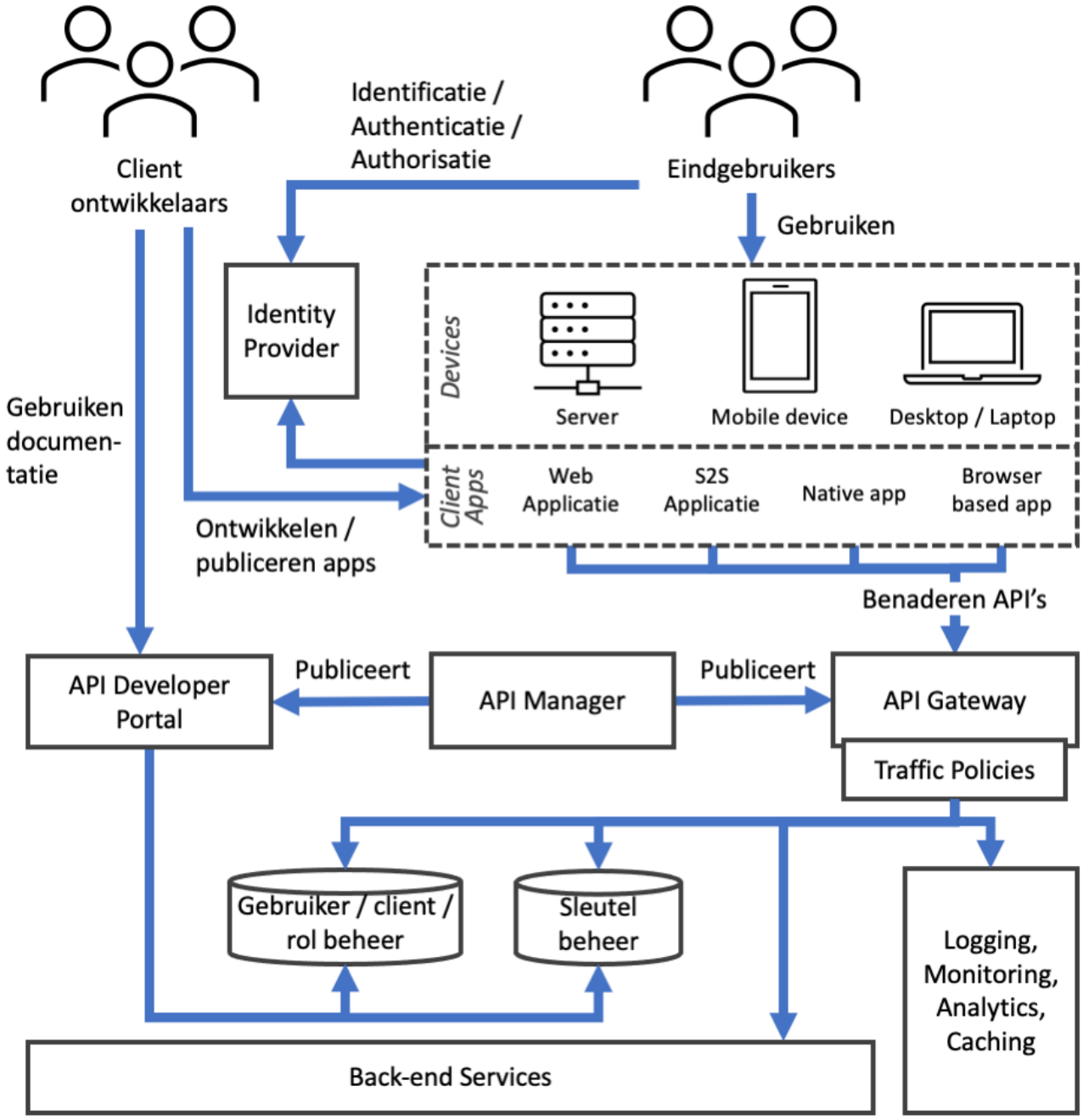
Vertrouwelijkheid gaat over het geheimhouden van gegevens en gegevensbronnen. Dit wordt gegarandeerd door scheiding van systeemfuncties, door controle op communicatiegedrag en gegevensuitwisseling, door validatie op toegang tot gegevens en systeemfuncties en door versleuteling van gegevens. Binnen de NORA is vertrouwelijkheid opgenomen als afgeleid principe AP43.

In de context van API's gaat vertrouwelijkheid over het ervoor te zorgen dat tussen API aanbieder en consument uitgewisselde gegevens niet door onbevoegden kunnen worden ingezien en misbruikt.

Aan vertrouwelijkheid gerelateerde API capabilities zijn *Caching*, *Analytics*, *Logging / Audit Trail*, *Identificatie / Authenticatie / Autorisatie*, *Sleutelbeheer* en *Gebruiker / Rol beheer*. De onderstaande sectie *Componenten* beschrijft deze in meer detail en geeft aan waar deze worden toegepast in een API architectuur.

5.6.4 Componenten

Onderstaande afbeelding geeft een overzicht van standaard componenten in een API architectuur. Deze sectie beschrijft de aan API beveiliging gerelateerde componenten in dit diagram.



Figuur 8: API Security

5.6.4.1 Actors en Clients

Onderstaand overzicht beschrijft de actoren en API Clients welke een rol spelen bij API beveiliging uit bovenstaand diagram.

- **Eindgebruikers:** Eindgebruikers welke door middel van devices en client apps gebruik maken van API's.

- **Client ontwikkelaars:** Ontwikkelaars van client API gebruikende client apps voor verschillende devices.
- **Web Applicatie:** een API client die op een server draait, welke door eindgebruikers wordt gebruikt middels een web interface.
- **System-to-System (S2S) Applicatie:** een API client die op een server draait, welke namens zichzelf (in plaats van namens een eindgebruiker) API's benadert middels een system-to-system koppeling, in plaats van namens een eindgebruiker.
- **Native App:** een API client die als native applicatie op het device van de eindgebruiker draait en zelfstandig API's benadert.
- **Browser-based App:** een API client die volledig in de browser draait en zelfstandig API's benadert, bijvoorbeeld op basis van JavaScript.

5.6.4.2 Componenten

Onderstaand overzicht beschrijft de componenten welke een rol spelen bij API beveiliging uit bovenstaand diagram.

- **Identity Provider:** Biedt de mogelijkheid aan eindgebruikers en Client applicaties om zichzelf te identificeren en Authentiseert deze. Geeft een authenticatie token af aan de Client applicatie waarmee deze API endpoints kan benaderen.
- **API Gateway:** De toegangspoort tot het achterliggende applicatie landschap. Alle API interactie verloopt via de API Gateway, waardoor de API Gateway een centrale rol heeft in de API beveiliging. Over het algemeen bevat de API Gateway een aantal traffic policies (zie volgend punt) welke bij elk API verzoek worden gevalideerd en voert de API Gateway initiële autorisatie van API verzoeken uit. Deze autorisatie omvat minimaal een controle of de ontvangen verzoeken van correcte Access Tokens zijn voorzien en kan eventueel role-based autorisaties toepassen. De API Gateway kan een verbinding met de Identity Provider hebben voor het geval door de Identity Provider afgegeven tokens onvoldoende informatie bevatten om autorisaties toe te passen.
- **Traffic Policies:** Een verzameling policies welke worden toegepast op al het API verkeer, of een specifiek deel daarvan. Voorbeelden zijn *Rate Limiting / Throttling*, *SLA Management* en *Input validatie*. Over het algemeen bieden API Gateways de mogelijkheden om traffic policies direct in te bouwen.
- **Gebruiker / client / rol beheer:** Het beheren van gebruikers van de API tooling, zoals beheerders, API developers en client developers, geregistreerde Client applicaties en een

mapping van eindgebruikers identiteiten en rollen naar specifieke autorisaties. Dit is *niet* de identity store welke credentials van eindgebruikers bevat; dat gedeelte is de verantwoordelijkheid van de Identity Provider.

- **Sleutel beheer:** Beheert het sleutelmateriaal dat wordt gebruikt voor versleuteling en ondertekening van berichten welke worden uitgewisseld tussen API Gateway, Identity Provider en Client applicaties.
- **Logging, Monitoring, Analytics, Caching:** Cross-cutting functionaliteiten welke op verschillende en/of meerdere plekken in het landschap geïmplementeerd kunnen worden. Voor al deze functionaliteiten is het van belang dat deze in lijn met AVG richtlijnen worden opgezet. Bij caching is het daarnaast belangrijk cache invalidaties te implementeren om te voorkomen dat verouderde cache data kan worden teruggehaald, zeker als daar privacy gevoelige informatie in kan staan.
- **Back-end Services:** Applicaties die de daadwerkelijke resources implementeren.

5.6.5 Principes

De volgende basisprincipes voor API beveiliging moeten worden toegepast bij het aanbieden van API dienstverlening:

- Beschouw elke API alsof deze potentieel als externe API aangeboden wordt, zelfs als daar momenteel nog geen plannen voor zijn.
- Zero-trust networking: elk applicatie-component gedraagt zich alsof deze aan het publieke netwerk zit.
- Gebruik generieke methoden voor authenticatie en authorisatie over alle API's, bij voorkeur op basis van bestaande componenten. Voorkom specifieke oplossingen voor individuele API's.
- Versleutel alle data in opslag (at rest) en in uitwisseling (in transit).
- Least privilege: API clients krijgen alleen de toegang die zij minimaal nodig hebben om hun functie uit te oefenen.

Verder moeten de aanbevelingen in de volgende externe documenten worden overwogen bij API ontwikkeling:

- OWASP Top Ten Cheat Sheet
- OWASP REST Security Cheat Sheet

- OWASP API Security Project

5.6.6 Architectuurpatronen

5.6.6.1 Gedelegeerde Identificatie en Authenticatie

Authenticatie beschrijft het met een bepaalde zekerheid vaststellen of een persoon of systeem echt degene is die deze zegt te zijn.

In de context van API's is authenticatie van toepassing op de API Client, oftewel de applicatie die de API resources benadert, en de eindgebruiker, oftewel de persoon namens wie de API Client de API resources benadert.

Waar authenticatie van de API Client bij de API resources plaats kan vinden, bijvoorbeeld door middel van asymmetrische versleuteling of gedeelde geheimen, wordt authenticatie van eindgebruikers over het algemeen gedelegeerd aan een externe Identity Provider, bijvoorbeeld door middel van OAuth2 of OpenID Connect.

Bij gedelegeerde Authenticatie identificeren de API Client en eindgebruiker zichzelf en ontvangt de API Client, na succesvolle authenticatie, een token waarmee de API resources kunnen worden bevraagd.

5.6.6.2 Token-based Autorisatie

Autorisatie beschrijft het vaststellen wat een geauthenticeerd persoon of systeem mag, of juist niet mag.

Door de toepassing van gedelegeerde identificatie en authenticatie in combinatie met tokens, wordt het ingewikkelder om applicatie-specifieke autorisatie toe te passen. De API aanbieder moet bijvoorbeeld autorisatie beslissingen nemen op basis van de beperkte set aan data in het ontvangen token.

Voor gevallen dat het token onvoldoende informatie biedt om een autorisatie beslissing op te baseren, kan het benodigd zijn om token introspection toe te passen. In dit geval vraagt de API aanbieder op basis van het ontvangen token meer informatie over de geauthenticeerde gebruiker bij de Identity Provider.

Een belangrijk component bij autorisaties in de context van API's is de API Gateway. Op basis van een door de Identity Provider afgegeven access token kan de API Gateway beslissen of API aanroepen zijn toegestaan of niet. Hierbij kan de API Gateway *role-based access control (RBAC)* toepassen; *domein-specifieke autorisaties* worden over het algemeen toegepast door de back-ends om implementatie van business logica in de API Gateway te voorkomen.

Voor de implementatie van *domein-specifieke autorisaties* in de back-ends kunnen tevens microgateways of een service mesh oplossing worden gekozen. Bijkomend voordeel van het gebruik van een service mesh is dat deze standaard werken op basis van zero-trust networking.

5.6.7 Referenties

- OWASP Top Ten Cheat Sheet
- OWASP REST Security Cheat Sheet
- OWASP - API Security Project
- NORA - Beschikbaarheid principe
- NORA - Integriteit principe
- NORA - Vertrouwelijkheid principe

A. Referenties

A.1 Informatieve referenties

[Expert]

Expertadvies OAuth 2.0. P. Dam. Forum Standaardisatie. 24 februari 2017. URL: https://www.forumstandaardisatie.nl/sites/bfs/files/Expertadvies%20OAuth%202.0.pdf

[iGOV.OAuth2]

International Government Assurance Profile (iGov) for OAuth 2.0. J. Richer, M. Varley, P. Grassi. OpenID foundation. October 5 2018. URL: https://openid.net/specs/openid-igov-oauth2-1_0.html

[iGOV.OpenID]

International Government Assurance Profile (iGov) for OpenID Connect 1.0 - draft 3. M. Varley, P. Grassi. OpenID foundation. October 5 2018. URL: https://openid.net/specs/openid-igov-openid-connect-1_0.html

↑