Nictiz

API-strategie voor de zorg

Strategie en architectuur

Versie 1.0 | 29 september 2022

Inhoud

Samenvatting

API's (application programming interfaces) zijn technische koppelvlakken binnen en tussen digitale informatiesystemen. Zij zijn van strategisch belang voor het zorginformatiestelsel: voor databeschikbaarheid, voor standaardisatie van informatie, voor samenhang tussen informatiestandaarden en voor technologie-gedreven innovatie. API's spelen een hoofdrol op het systeemniveau van het zorginformatiestelsel. Daarom werkt Nictiz, in samenwerking met velen, aan een API-strategie voor de zorg.

De API-strategie werkt aan meer goede API's, langs een kwaliteitsmodel met drie treden: van open (gedocumenteerde) API's, via technisch gestandaardiseerde API's naar ook inhoudelijk gestandaardiseerde API's. Het onderscheidt daarvoor vier verschijningsvormen: API-afspraken, API-specificaties, API-implementaties en API-deployments, met de respectievelijke verantwoordelijkheden van partijen. Om haar doel te bereiken, raakt de strategie aan belangenbeleid, architectuurbeleid, kwalificatiebeleid en softwaremarktbeleid. Om te beginnen doet zij dat met drie strategische middelen: API-eisen, een API-bibliotheek en een API-technologie-agenda.

De API-eisen worden door een werkgroep van partijen uit het veld opgesteld, en beschrijven waaraan API's op de drie treden moeten voldoen. Op de hoogste trede wordt van API-specificaties implementatie-onafhankelijkheid vereist. De eerste aandacht is er voor API's die over HTTP werken, het protocol waarmee een groot deel van het Web werkt. De API-bibliotheek publiceert API's die aan de eisen voldoen, op de respectievelijke treden, en moet op termijn onderdeel worden van de Nationale Bibliotheek. De API-technologie-agenda moet het gebruik en de uitfasering van specifieke API-technologieën begeleiden, in lijn met internationale technologische ontwikkelingen. Aan deze agenda gaat gewerkt worden wanneer de API-strategie bij haar houder is ondergebracht.

Het document in uw handen beschrijft de algemene aspecten van de API-strategie en, in tweede hoofdstuk, de architectuurkaders waarin zij is ondergebracht, en waarmee zij wordt uitgewerkt. In andere documenten komen onder andere de API-eisen aan de orde, de API-bibliotheek en een eerste vulling daarvan aan de orde.

Documentatie van de API-strategie

De API-strategie staat nu beschreven in vijf documenten:

1. dit document;
2. een document met de genoemde API-eisen;
3. een document waarin de API-bibliotheek en haar ontwikkeling staan beschreven;
4. een document over de eerste verzameling van kandidaat-API's voor de API-bibliotheek;
5. een document over het organiseren van ontwikkeling, beheer en uitvoering van deze API-strategie.

Aan deze reeks documenten wordt de API-technologie-agenda toegevoegd wanneer duidelijk is welke partij houder is van de API-strategie.

Auteurs

Dit document is opgesteld door Paul Oude Luttighuis, met bijdragen en ondersteuning van Gerda Meijboom, Sasja Beers, Marianne Plandsoen, Remko Nienhuis, Marc Sandberg, Alexander Henket (allen in hun Nictiz-rol) en dankzij vruchtbare gesprekken met Gieneke van Veenen en René Meijer (Ministerie van VWS).

1

Een API-strategie voor de zorg

1.1 Strategisch aandacht voor API's

Data beschikbaar voor hergebruik

API's (application programming interfaces) zijn technische koppelvlakken binnen en tussen digitale informatiesystemen. Zij ontsluiten data en transacties: de data die door die systemen worden vastgelegd en de transacties die door die systemen worden uitgevoerd. Voor data en transacties worden aparte typen API's onderscheiden. Data-API's zorgen voor technische *databeschikbaarheid*[1]: zij ontsluiten data buiten transacties om. Zo onthullen zij op een gecontroleerde manier gegevens uit het ene systeem voor hergebruik door het andere systeem. Dat maakt data-API's van strategisch belang voor het zorginformatiestelsel.

Standaardisatie van informatie

In het zorginformatiestelsel worden informatiestandaarden - voor hetzij registratie, hetzij uitwisseling - ontwikkeld en beheerd door verschillende partijen en met verschillende eindverantwoordelijke houders. Elke informatiestandaard is bedoeld voor een zorgsituatie of *use case* en beschrijft:

- in gebruikerstaal wat er aan de orde is (conceptueel);
- de gegevensset die daaraan invulling geeft (logisch);
- de API's waarlangs informatiesystemen die gegevenssets uitwisselen (technisch).

Daarmee vervullen data- en transactie-API's een sleutelrol in de *standaardisatie van informatie* in het zorginformatiestelsel.

Samenhang tussen informatiestandaarden

Het is van groot belang dat informatiestandaarden samenhangen. Dat borgt hun gebruiksvriendelijkheid en implementeerbaarheid. Eenzelfde zorgmedewerker is immers betrokken in vele zorgsituaties; diens informatiesysteem ook, maar weer in andere combinaties. Dat is nu al het geval en dat zal alleen maar toenemen door voorziene ontwikkelingen naar passende zorg en hybride zorg. Passende zorg vraagt om intensievere samenwerking en hybride zorg vraagt meer digitale middelen, die naadloos aansluiten bij de fysieke zorgverlening.

Samenhang tussen informatiestandaarden vereist dat hun gegevenssets zijn samengesteld uit dezelfde *zorginformatiebouwstenen*. Die bouwstenen zijn dan per stuk herkenbaar voor zorgverleners en implementeerbaar in informatiesystemen – los van de specifieke zorgsituatie en het zorginformatiesysteem. API's kunnen dan net zo worden samengesteld uit technische fragmenten, als gegevenssets bestaan uit zorginformatiebouwstenen[2]. Daarmee spelen API's op systeemniveau een strategische rol in de *samenhang in specificatie* van het zorginformatiestelsel.

Scheiding tussen API-specificatie en API-implementatie

Binnen een zorginformatiestelsel is eerst belangrijk wat digitale systemen doen en daarna hoe. In softwarekringen maakt men daarvoor onderscheid tussen een API-specificatie en een API-implementatie. Een API-specificatie is een document waarin precies staat wat een API te bieden heeft. Een API-implementatie is de software die doet wat de API-specificatie zegt. Deze scheiding is van strategisch belang voor de regievoering op het zorginformatiestelsel. Zij maakt het mogelijk om precieze afspraken te maken en besluiten te nemen (regie te voeren) over wat er nodig is los van de vraag hoe de software moet worden gebouwd.

[1] Zie ook paragraaf 2.7.

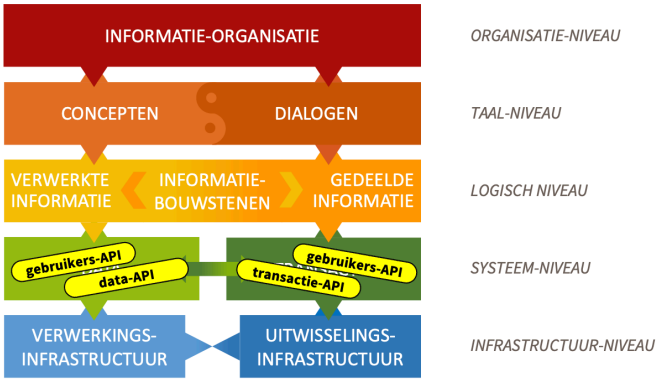
[2] Een actueel voorbeeld hiervan is opgenomen in Bijlage A.

De scheiding tussen specificatie en implementatie heeft op termijn een tweede strategisch effect. Zij maakt het mogelijk om vanuit eenzelfde API-specificatie verschillende softwareleveranciers verschillende API-implementaties te laten bouwen. Dit bevordert een open softwaremarkt. Dit vraagt wel een inhoudelijk rijk genoeg gedocumenteerde API-specificatie, die willekeurige softwareontwikkelaars mogen gebruiken zonder juridische beperkingen. Net zoals bij open standaarden.

Implementatie-onafhankelijkheid van API-specificaties is geen eis vanaf het prille begin. Het is een actief te bevorderen kwaliteit in een voldoende volwassen softwaremarkt. Dit komt omdat API-specificaties doorgaans pas op papier komen nadat er een (eerste) implementatie is gemaakt. Dit geldt zeker in innovatieve domeinen waarin standaardisatie nog minder vergevorderd is.

Technologie-gedreven innovatie

API's zijn dus strategisch belangrijk voor databeschikbaarheid, voor standaardisatie en voor samenhang in specificatie. Toch is de grote rol van API's op deze punten uiteindelijk niet leidend maar dienend. Dat betekent dat een API-strategie (op systeemniveau) voor de verwezenlijking van haar doelen kritiek afhankelijk blijft van andere deelstrategieën van het zorginformatiestelsel, op hogere niveaus (afbeelding 1)[3]. Technologie kan de rol van die hogere niveaus onmogelijk overnemen. Technologie kan wél de noodzaak van die andere deelstrategieën duidelijker maken.



Afbeelding 1: Positie van API's is het zorginformatiestelsel.

Een zorginformatiestelsel dat inzet op passende en hybride zorg moet de kracht van API's inzetten op het juiste niveau.

Digitale technologie is en blijft een middel voor informatiegebruiksdoelen. Toch kunnen API's een krachtige en welkome rol spelen in het uitdagen en ontdekken van nieuwe informatiebehoeften. Via API's kunnen data en transacties technisch beschikbaar komen: nog voordat deze op de hogere niveaus gespecificeerd en gestandaardiseerd zijn en nog voordat deze implementatie-onafhankelijk zijn. Een API-strategie voor de zorg moet ook deze technologie-gedreven innovatiekracht van API's haar werk laten doen. Maar ze moet er wel voor blijven zorgen dat API's uiteindelijk altijd ondergebracht worden in expliciete informatiebehoeften, implementatie-onafhankelijk.

[3] Het model op de achtergrond is een verfijning van het lagenmodel van Nictiz voor interoperabiliteit. Bijlage B licht de relatie tussen de modellen preciezer toe.

Wat voorafging

De ontwikkeling van een API-strategie voor het zorginformatiestelsel heeft Nictiz ter hand genomen, in samenwerking met vele anderen. Dit is een vervolg op een eerder door Nictiz uitgebracht advies aan het Ministerie van Volksgezondheid, Welzijn en Sport (VWS) [4]. Dat advies werd gevraagd in de context van de beantwoording van Kamervragen over het wetsvoorstel Wegiz (Wet Elektronische gegevensuitwisseling in de zorg).

1.2 Doel en groeipad van een API-strategie

Doel

Databeschikbaarheid vraagt om meer dan alleen data technisch bevrijden uit systemen. Dan komt de data namelijk alleen in ruwe technische vorm beschikbaar, zonder dat duidelijk is of en hoe die data bruikbaar is buiten het systeem. Daarom moet een API-strategie niet alleen streven naar inzet van meer API's, maar ook naar inzet van goede API's: API's die data (en transacties) zo beschikbaar stellen dat zij bruikbaar zijn in expliciete zorginformatiebehoeften. Een API-strategie moet die bruikbaarheid dus definiëren en bevorderen.

Het doel van de API-strategie is:
inzet van meer goede API's.

Groeipad

Goede API's maken data en transacties vrij die bruikbaar zijn in de context van de informatiebehoeften van de zorg. Zoals in paragraaf 1.1 benadrukt, zijn API's voor die gebruikskwaliteit afhankelijk van afspraken op de hogere niveaus: het logische niveau, het taal- en het organisatieniveau. Dat speelt niet alleen bij databeschikbaarheid maar ook bij de standaardisatie, en bij gebruiksvriendelijkheid en implementeerbaarheid van die standaarden.

Toch beperkt een API-strategie zich er niet toe eenvoudigweg te stellen dat API's moeten passen in afspraken op hogere niveaus. Daarmee zouden kansen worden gemist op welkome en passende technologie-gedreven innovatie (paragraaf 1.1). Bovendien zou dat de mogelijkheid ontnemen om actief strategie te voeren op de *groei* van de kwaliteit van API's.

Daarom hanteert de API-strategie een groeipad voor API's langs drie treden. Op de eerste trede stelt de API-strategie eisen aan de documentatie van de API-specificatie, als minimale kwaliteit om op voort te bouwen. Zonder geschikte documentatie mist een API elk aangrijpingspunt voor groei. Op de tweede trede stelt de API-strategie ook technische eisen aan API's. Hier kunnen API's groeien in interoperabiliteit en implementeerbaarheid, zonder een afhankelijkheid van hogere niveaus (het logische niveau, het taal- en het organisatieniveau). Die afhankelijkheid bestaat wel op de derde en hoogste trede (afbeelding 2).

[4] Nictiz, *Op weg naar een API-strategie voor de zorg*, Adviesrapport aan Ministerie van VWS, versie 1.0, 17 december 2021, https://www.nictiz.nl/wp-content/uploads/Advies-API-strategie-v1.0.pdf.

Afbeelding 2: Groeipad binnen de API-strategie.

Het doel van de API-strategie is om veel API's te verwelkomen op de eerste trede van het groeipad en deze te stimuleren in hun groei naar hogere treden. Vanzelfsprekend kunnen API's ook onmiddellijk instappen op hogere treden, wanneer zij aan desbetreffende eisen voldoen.

Langs dezelfde treden groeien API's in hun implementatie-onafhankelijkheid (paragraaf 1.1). Op de eerste twee treden wordt die nog niet vereist maar op de derde wel. Zo staat de API-strategie zo lang mogelijk open voor technologie-gedreven innovatie en ziet zij het implementatie-onafhankelijk worden van API's als volwassen worden. Maar groei is niet verplicht. Het staat een specificerende partij vrij om daarvan af te zien (en een API op een eerste of tweede trede te laten).

Er zijn autoriteiten nodig die de inhoudelijke standaarden vaststellen waarop een API wordt beoordeeld voor de derde trede. Het valt buiten de scope van de API-strategie om deze autoriteiten te benoemen, maar op de derde trede is de API-strategie wel van hun werk afhankelijk.

Een volledig gestandaardiseerde API (op de derde trede) geniet die status alleen voor de inhoudelijke informatiestandaard waarop de API is beoordeeld. Zo kan het voorkomen dan een API vaker voorkomt op trede drie namelijk voor verschillende informatiestandaarden (waarvoor de API geschikt is bevonden).

1.3 API-speelveld en krachten

API specifiers en API developers

Een effectieve API-strategie houdt rekening met diverse rollen van partijen op het API-speelveld. Zo zijn er de *API specifiers* die de API-specificatie schrijven. De *API developers* zijn de partijen die de software leveren waarin die API's zijn ingebouwd.

Als een softwareleverancier specificeert welke API's zijn software biedt, dan vervult deze leverancier beide rollen. Het kan ook voorkomen dat een standaardisatie-organisatie een API standaardiseert (als API specifier), waarna meer softwareleveranciers die API inbouwen in hun software (als API developers).

Bij de dagelijkse inzet van de API's is er een aanbiedend systeem (de server) en een afnemend systeem (de client). Beide zijden moeten eerst geïmplementeerd zijn. Daarom worden de *API server developer* en de *API client developer* als aparte rollen onderscheiden. Bevinden server en client zich binnen eenzelfde (informatie)systeem, dan vervult een API developer beide rollen.

API server deployers en API client deployers

Degene die verantwoordelijk is voor de dagelijkse inzet van een API is de *API deployer*. Een softwareleverancier bijvoorbeeld is als API developer niet verantwoordelijk voor de dagelijkse inzet van de API's uit zijn software. Een gebruikersorganisatie als klant van een softwareleverancier kan deze rol vervullen. Steeds vaker echter draagt een derde partij deze verantwoordelijkheid, als dienstverlener aan de gebruikersorganisatie.

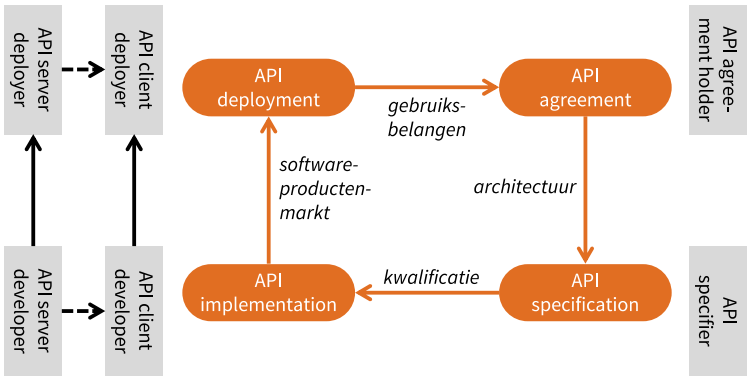
Ook hier is sprake van een aanbiedend (server) en een afnemend systeem (client). Daarom bestaat er een rol die verantwoordelijk is voor de dagelijkse inzet van de API's aan de server-kant en een rol die verantwoordelijk is voor de inzet van de API's aan de client-kant: de *API server deployer* en *API client deployer*. Betreft het API's binnen eenzelfde systeem, zullen de API server deployer en de API client deployer dezelfde partij zijn.

API agreement holder

Tot slot moeten de twee uitwisselende partijen (de API server deployer en de API client deployer) een API-overeenkomst aangaan om elkaar te kunnen vertrouwen. Als de API server deployer en de API client deployer dezelfde partij is, is de 'overeenkomst' een besluit in de software-architectuur van die partij. Deze partij is daarmee de *API agreement holder*. Zijn het twee verschillende partijen dan zijn zij samen de API agreement holder. Een laatste mogelijkheid is dat een grote groep collectief (omschreven) API's van elkaar gebruikt. Dit gebeurt bij een afsprakenstelsel als MedMij. In zo'n situatie is een derde partij de API agreement holder, waarmee alle API server deployers en API client deployers een deelnemersovereenkomst aangaan.

Speelveld

De API-overeenkomst – in welke variant dan ook – verwijst naar de API-specificatie(s). Daarmee is het speelveld rond (afbeelding 3). De vier hoeken in het speelveld zijn onderling afhankelijk. In één API-overeenkomst kunnen meerdere API-specificaties voorkomen. Een API-specificatie kan worden uitgewerkt in verschillende API-implementaties en er kunnen meer partijen verantwoordelijk zijn voor het dagelijks gebruik van software waarin de API's zijn ingebouwd (meer API deployers voor een API-implementatie).



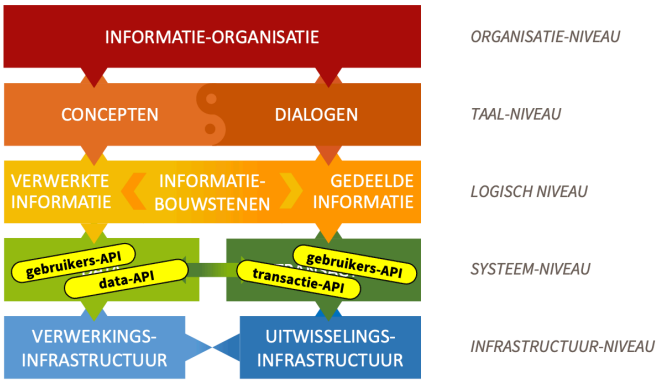
Afbeelding 3: API-speelveld.

Belangenbeleid

Tussen de vier hoeken in dit API-speelveld werken krachten die desbetreffende zijde van het vierkant kenmerken. Belangen, verbonden aan informatiegebruik, brengen API deployers ertoe samen een API-overeenkomst of uitwisselovereenkomst aan te gaan. Een API-strategie die de totstandkoming van API-overeenkomsten wil bevorderen, moet invloed uitoefenen op deze belangen. Die kunnen gaan over informatiebehoeften, financiële voorwaarden of een wens van een API deployer om aan te sluiten bij een grote collectieve overeenkomst (strategisch belang).

Architectuurbeleid

Om de API-specificaties binnen één API-overeenkomst samen de belangen van de deployers te laten dienen, en om API-specificaties te standaardiseren voor hergebruik in meerdere API-overeenkomsten, is architectuurbeleid nodig. In hoofdstuk 2 van dit document worden daarvoor de fundamenten gelegd (afbeelding 1).



Afbeelding 4: Positie van API's is het zorginformatiestelsel.

Kwalificatiebeleid

API-specificaties (en daarmee API-overeenkomsten die ernaar verwijzen) worden werkelijkheid als zij worden ingebouwd in API-implementaties. Om te bevorderen dat er gekwalificeerde of goed bevonden API-implementaties beschikbaar komen, moeten een API-strategie kwalificatiebeleid voeren. Bijvoorbeeld door API-specificaties goed beschikbaar te maken voor API developers of door hen te ondersteunen bij het implementeren. Een in software ingebouwde API kan men toetsen op haar voldoen aan de API-specificatie. Zo kwalificeert een API-specificatie software op diens compliance aan de specificatie.

Softwaremarktbeleid

API-implementaties moeten op een gezond functionerende softwaremarkt beschikbaar komen voor gebruik door API deployers. Rolkeuzes van softwarepartijen bepalen voor een belangrijk deel de krachten op deze markt. Zo is menig softwarepartij tegelijk API specifier, API developer als API deployer. Een API-strategie die de beschikbaarheid van API-implementaties wil bevorderen, kan beleid voeren op die rolverdelingen. Andere mogelijkheden zijn de transparantie van de markt vergroten, inkoopsamenwerking organiseren of eisen stellen aan toegang van softwarepartijen tot die markt.

1.4 Middelen van de API-strategie

De API-strategie zet vooralsnog drie strategische middelen in: API-eisen, een API-technologie-agenda en een API-bibliotheek. Zoals in paragraaf 1.3 beschreven bestaan in het speelveld vier soorten beleid: rond belangen, architectuur, kwalificatie en softwaremarkt. Een API-strategie richt zich primair op architectuur- en kwalificatiebeleid. Van een API-strategie kan amper eigenstandig effect worden verwacht op gebruiksbelangen en softwaremarkt. Gebruiksbelangen tussen API deployers beginnen immers op het organisatie- en taalniveau (afbeelding 1) en eindigen op systeemniveau. Het organisatie- en taalniveau liggen buiten bereik van een API-strategie. Dat wil niet zeggen dat een API-strategie hier geen bijdrage kan leveren, maar wel dat die bijdrage alleen effectief kan worden in de context van breder stelselbeleid.

Softwaremarktbeleid op haar beurt is niet alleen een API-aangelegenheid. Software omvat veel meer dan API's, zoals eindgebruikersfunctionaliteit. Ook dat ligt buiten bereik van een API-strategie. Opnieuw geldt dat een API-strategie hier kan en moet bijdragen, maar pas effectief kan worden in de context van algemeen softwaremarktbeleid.

API-eisen

De API-specificatie verbindt architectuur- en kwalificatiebeleid (afbeelding 3) en vormt daarmee de spil van de API-strategie. Niet voor niets is een goede documentatie van die API-specificatie een minimumvereiste in het groeipad voor API's (afbeelding 2). De API-strategie voert architectuurbeleid door *API-eisen* te formuleren op de drie treden van dat groeipad.

De API-eisen zijn het eerste strategisch middel van de API-strategie. Op de eerste trede betreffen deze eisen vooral de documentatie van de API-specificatie; op de hogere treden spelen ook de andere verschijningsvormen van API's: de API-implementatie, de API-inzet en de API-overeenkomst. Steeds worden de eisen verbonden aan een specifieke rol uit afbeelding 3. Zo wordt elke partij in het API-speelveld aangesproken op de API-eisen die zijn verbonden aan precies die rollen die deze partij speelt. Deze precisie maakt dat de API-strategie accuraat kan beïnvloeden en bevordert dus haar effectiviteit.

API-technologie-agenda

De API-strategie moet een 'rollende' *API-technologie-agenda* opstellen en onderhouden. Deze agenda wijst voortdurend aan welke API-paradigma's en -technologiestandaarden van belang zijn op de tweede trede van het groeipad voor API's (afbeelding 2). Dat betekent dat er specifieke API-eisen kunnen bestaan voor elk API-paradigma dat, en elke API-technologie die, op enig moment op de API-technologie-agenda staat.

Op de tweede trede van het groeipad voor API's stelt de API-strategie eisen aan de technische standaardisatie van API's. Op die trede moet de API-strategie rekening houden met een actuele en structurele diversiteit in API-technologieën en hoe die fundamenteel zijn opgezet (de API-paradigma's). Dit maakt het (om diverse redenen) riskant om in het zorginformatiestelsel te kiezen voor slechts een van dergelijke API-paradigma's en -technologieën.

Om te beginnen kennen data-API's van nature een andere opzet dan transactie-API's. Maar ook binnen deze categorieën bestaan wezenlijke verschillen tussen API-paradigma's. De zogenoemde REST-paradigma doet momenteel opgeld voor data-API's. Ook oudere paradigma's zijn nog

steeds zeer gangbaar en aan de horizon melden zich nieuwe paradigma's. En zo bestaan ook voor transactie-API's die diversiteit en evolutie.

De interoperabiliteit en implementeerbaarheid van het zorginformatiestelsel vragen erom voortdurend selectief te zijn in de keuze uit de op enig moment gangbare en voorhanden API-paradigma's en -technologieën. Daarbij moeten terdege rekening worden gehouden met de *installed base*: de op basis van vroegere keuzes in gebruik genomen API-implementaties. Die kan niet zomaar afgeschreven worden, wel gaandeweg aangepast of vervangen.

Mocht het Nederlandse zorginformatiestelsel zich toch willen vastleggen op slechts een technologische standaard, dan komt ooit het moment waarop die keuze moeten worden herzien. Dan staat het stelsel voor een enorme transitie-uitdaging. Een bijkomende keuze voor bijbehorende logische standaarden maakt die transitie bijna ondoenlijk: de transitie vraagt dan ook een herontwerp op de hogere niveaus. Dat is een reden om bij voorkeur een duidelijke scheiding aan te houden tussen de gegevensinhoudelijke en de technische standaardisatie (afbeelding 1).

Standaardisatie van specifieke API-technologieën vindt (ook wereldwijd) vooral plaats per maatschappelijke of economische sector. Als internationale standaardisatieorganisatie voor het zorgdomein kent HL7 zelf standaarden voor verschillende paradigma's, waaronder FHIR voor onder andere het REST-paradigma. De technologische praktijk is en zal waarschijnlijk divers en evoluerend blijven: niet zorg-specifiek, wel internationaal.

Een API-technologie-agenda opstellen en onderhouden is niet alleen een technologische aangelegenheid. Er zijn substantiële belangen, risico's en kosten mee gemoeid op de schaal van het hele zorginformatiestelsel. Zo'n agenda vaststellen is daarmee een wezenlijk aandachtpunt voor de partij die als houder van de API-strategie gaat optreden. Zolang deze partij niet is aangewezen en een API-technologie-agenda voorlopig ontbreekt, kan men op voorhand beginnen met API-eisen voor REST API's en mogelijk daarna met een paradigma voor transactie-API's. Deze twee paradigma's zijn gangbaar en zullen dat waarschijnlijk nog enige tijd blijven.

Zie hiervoor ook paragraaf 2.3.

API-bibliotheek

Om met de API-eisen effect te sorteren in het API-speelveld (afbeelding 3), moet de API-strategie kwalificatiebeleid voeren. Het is nodig de API-eisen en de achterliggende API-technologie-agenda te publiceren en toe te lichten. Daarnaast sorteert de API-strategie gerichter effect door ook te communiceren *wat* voldoet aan de API-eisen.

1. Door te communiceren welke *API-specificaties* voldoen aan de zekere API-eisen, biedt de API-strategie aan API agreement holders de mogelijkheid vertrouwde API-specificaties op te nemen in hun API-overeenkomsten. Dit bevordert de adoptie van gestandaardiseerde API's. Bovendien kan de wetgever zo nodig putten uit deze bibliotheek om zulke API-specificaties te normeren en te verplichten.
2. Door te communiceren welke *API-implementaties* voldoen, biedt dat API deployers inzicht in het te vertrouwen aanbod op de softwaremarkt en stimuleert dat een competitieve softwaremarkt tot aanbod van meer goede-API-implementaties. Daarnaast stimuleert die duidelijkheid kennisdeling tussen API developers en biedt dat API specifiers inzicht in de implementeerbaarheid van hun specificaties.
3. Door te communiceren welke *API (server) deployments* voldoen aan zekere API-eisen, krijgen API client deployers de mogelijkheid met deze API (server) deployers een API-overeenkomst aan te gaan en zo een nieuwe uitwisseling te realiseren.
4. Door helderheid over welke *API-overeenkomsten* voldoen aan de API-eisen, biedt dat API deployers de gelegenheid zich aan te sluiten bij zo'n overeenkomst (voor zover de desbetreffende API agreement holder daarvoor openstaat).

Daarvoor ontwikkelt en onderhoudt de API-strategie een publiek ontsloten *API-bibliotheek*. De API-bibliotheek biedt aan de API-strategie zelf voortdurend overzicht over de huidige situatie en haar eigen effectiviteit.

Om de inhoud van de API-bibliotheek te beheren is het nodig de verschillende verschijningsvormen van API's (zoals API-specificaties en API-implementaties) te toetsen op hun voldoen aan API-eisen. Hoe dat moet plaatsvinden valt nog te bepalen (onafhankelijke toetsing; zelfverklaring; anderszins).

2

API-architectuur

2.1 Definitie en verschijningsvormen van API's

Definitie

Met een *application programming interface* (API) ontsluit een softwareapplicatie functionaliteit voor gebruik door andere softwareapplicaties, zonder die andere applicaties te belasten met hoe die functionaliteit wordt uitgevoerd.

Aanbieder en afnemer

API's hebben aanbieders en afnemers. Een aanbieder ontsluit een functionaliteit (volgens de API-specificatie). Een afnemer gebruikt deze functionaliteit door een API aan te roepen. Afnemers kunnen meer API's aanroepen en bij verschillende aanbieders.

Verschijningsvormen: overeenkomst, specificatie, deployment en implementatie

Een afnemer gaat met de aanbieder een API-overeenkomst aan. Daarin staan hun afspraken over de functionele en niet-functionele kenmerken van de API en gebruiks- en leveringsvoorwaarden. De term *overeenkomst* mag breed worden opgevat. Als een partij aanbieder en afnemer tegelijk is, betreft de overeenkomst een intern architectuurbesluit.

De functionaliteit wordt beschreven om andere applicaties te laten weten wat hun wordt aangeboden en hoe zij hierop kunnen aansluiten. Die technische specificatie is de *API-specificatie*. Zowel aanbieder als afnemer moet software hebben draaien die het eigen aandeel in het API-verkeer afhandelt: de *API-deployments*. Die gebruiken softwareproducten waarin de API-specificatie zijn ingebouwd: de *API-implementaties*.

Bij elke API-specificatie horen één of meer API-implementaties en bij elke API-implementatie horen één of meer API-deployments. Idealiter hoort bij elke API-deployment een API-overeenkomst[5]. Een API-overeenkomst betreft één of meer API-specificaties. Waar het onderscheid tussen specificatie, implementatie, deployment en overeenkomst niet van belang is, spreekt de API-strategie kortweg van een *API*. De verantwoordelijkheden die horen bij deze verschijningsvormen staan geordend in een rollenmodel (zie paragraaf 2.5).

De API-strategie betreft alle aspecten van API's die van belang zijn voor de interoperabiliteit tussen aanbiedere en afnemer. Daaronder vallen:

— de gegevens en acties;
— alle technische eigenschappen van de API die een potentiële API-afnemer moet kennen;
— de processen rondom API-specificatie en -documentatie.

Op onderdelen kunnen API-specificaties verwijzen naar breder toepasbare specificatie-onderdelen (en uiteindelijk naar internationale technische basisstandaarden).

2.2 Typologieën van API's

De internationale gemeenschap classificeert API's op verschillende manieren. Het algemene deel van de Nederlandse API-strategie[6] biedt daarvan goede voorbeelden:

— interne – en externe API-overeenkomsten;

[5] Als eenzelfde partij aanbieder en afnemer is, staat de term API-overeenkomst voor interne architectuurbeslissing.

[6] https://docs.geostandaarden.nl/api/API-Strategie/#typologie-van-api-s

— unrestricted-use en restricted-use API deployments;
— interactiepatronen;
— system –, process – en convenience API's.

Interne en externe API-overeenkomsten

De Nederlandse API-strategie onderscheidt interne en externe API's. Interne API's vinden toepassing binnen één organisatie; externe API's worden gebruikt voor verkeer tussen organisaties. Als de inhoudelijk eindverantwoordelijke aanbieder dezelfde partij is als de inhoudelijk eindverantwoordelijke afnemer, werkt die partij met interne API's. Zijn aanbieder en afnemer verschillende partijen dan spelen tussen die partijen externe API's.

Het onderscheid tussen intern en extern is een kwestie voor de API-overeenkomst. Dezelfde API-specificatie, API-implementatie of API-deployment kan zowel extern als intern worden ingezet. Die vrijheid wil de API-strategie ook behouden. Hooguit waar eisen aan API-overeenkomsten worden gesteld, kan het onderscheid aan de orde komen.

Unrestricted – en restricted-use API-deployments

De Nederlandse API-strategie verdeelt externe API's in *open*[7] en *gesloten* API's. Bij gesloten API's gelden voorwaarden om de gespecificeerde functionaliteit te mogen gebruiken. Voor de verificatie van het recht op gebruik zijn ook autorisatie-API's en authenticatie-API's nodig. Bij open API's (gebruikt voor bijvoorbeeld open data) ontbreken toegangsvoorwaarden en ontbreekt de behoefte aan autorisatie en authenticatie. Internationaal echter betekent *open API* iets anders. Omdat de API-strategie voor de zorg de internationale termen wil overnemen, gebruikt zij voor het onderhavige onderscheid de termen *unrestricted-use API* en *restricted-use API*.

Het onderscheid tussen unrestricted-use en restricted-use API's speelt uiteindelijk bij een API-deployment. Daarin krijgt toegangsbeleid zijn beslag, dat in voorafgaande API-overeenkomsten, API-specificaties en API-implementatie vorm heeft gekregen.

De Nederlandse API-strategie verdeelt de restricted-use API's naar de betrokken partijen: burger, bedrijf, overheid. Het is de vraag of deze driedeling zinvol is voor de doelen van de API-strategie voor de zorg. In de zorg springen natuurlijk allereerst de zorgaanbieder en de patiënt (of anders genoemd) in het oog. Maar ook andere organisatiesoorten spelen een belangrijke rol in het zorginformatieverkeer: overheidsuitvoerders, verzekeraars, kwaliteitsregistraties, onderzoeksorganisatie en andere. De API-strategie voor de zorg maakt op voorhand geen keuze voor typen partijen in het zorginformatiestelsel, zorgsectoren of dergelijke. De kans bestaat zelfs dat API's die bedoeld zijn voor patiënt-zorgaanbiederverkeer ook bruikbaar zijn voor ander verkeer.

Interactiepatronen

API's verschillen ook in hoe client en server afstemmen *wanneer* en *welke* gegevens uitgewisseld moeten worden. De client weet welke gegevens hij nodig heeft, de server weet wanneer die beschikbaar zijn. In het meest eenvoudige patroon (*pull*-patroon of *polling*) vraagt de client en antwoordt de server. Nadeel daarvan is dat, wanneer er helemaal geen nieuwe gegevens blijken te zijn, de client het later nog eens moet proberen. Dat kan leiden tot onnodig vraag–antwoordverkeer[8].

[7] niet te verwarren met de technologie OpenAPI uit paragraaf 2.4

[8] HTTP kent technische mogelijkheden voor het beperken van dit verkeer aan de server-zijde, namelijk met het in de cache identificeren van versies van resources, met een zogenoemde ETag.

Een ander patroon is het *push*-patroon waarin de server het initiatief neemt op het moment dat nieuwe gegevens beschikbaar zijn. De server stuurt naar de clients die daarvoor staan geregistreerd, het bericht dat er wijzigingen zijn en welke. Ook hieraan kleven in sommige situaties nadelen. Omdat de client weet welke gegevens hij wanneer nodig heeft, kan het tweede deel van de boodschap (wat de wijzigingen zijn) voorbarig worden.

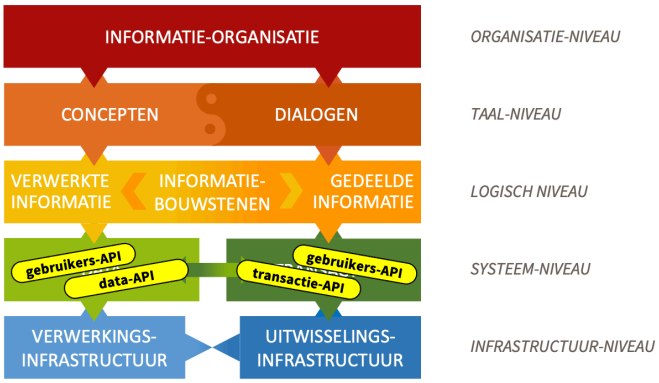
Een subtieler patroon is de *pull-notificatie* of *notified-pull*. De server vertelt de client eerst dát er nieuwe gegevens beschikbaar zijn (de *notificatie*). In reactie daarop vraagt de client op door hem gekozen momenten deze of andere servers de gegevens. Door de scheiding tussen dát en wát is dit patroon het meest geraffineerd in de afstemming tussen client en server en in de mogelijkheden voor dataminimalisatie. Bovendien is dit patroon veiliger omdat de client gevraagd kan worden zich opnieuw te authentiseren voordat de gegevens zelf worden gestuurd.

De API-strategie voor de zorg kent vooraf geen voorkeur voor een van deze drie patronen. Het derde patroon spreekt de client en server het meest precies aan: de cliënt voor het wát, de server voor het dát. Het pull-patroon legt beide bij de client, het push-patroon juist bij de server. Dat maakt deze twee eenvoudiger dan *notified-pull*. Al naar gelang de situatie geniet één van de drie de voorkeur.

System API's, process API's en convenience API's

Een andere indeling is die naar kolom (tier) in de software-architectuur (van betrokken applicatie) waarop de API aangrijpt. Een *system* API grijpt aan op de database, een *process* API op de business logica en de *convenience* of *experience* API op de user interface.

Dit onderscheid is in de API-strategie voor de zorg van groot belang. Het onderscheid tussen *system* API en *process* API weerspiegelt het onderscheid tussen verwerking van informatie (links in afbeelding 1) en uitwisseling van informatie (rechts in afbeelding 1) in het zorginformatiestelsel.[9]



Afbeelding 5: Positie van API's is het zorginformatiestelsel.

De huidige behoefte aan informatiebeschikbaarheid maakt dit onderscheid actueel en legt de nadruk op data-API's, hoewel de API-strategie ook transactie-API's betreft.

De nationale API-strategie onderscheidt ook *business* en *exposure* API's. De functionaliteit van business API's omvat specifieke (proces)aspecten. Bij exposure API's ontbreken die en is de

[9] Afbeelding 1 noemt system API's en process API's data-API's en transactie-API's. Engelstalige teksten gebruiken de in de software-community gangbare termen.

functionaliteit algemeen en gegevensgericht. Zo'n onderscheid mist scherpte, lijkt op het vorige onderscheid en is daarom vooralsnog niet van waarde in de API-strategie voor de zorg.

Open API's, technisch gestandaardiseerde API's en volledig gestandaardiseerde API's

De doelstelling van de API-strategie geeft aanleiding om drie kwaliteitstreden te onderscheiden: open API's, technische gestandaardiseerde API's en volledig gestandaardiseerde API's (afbeelding 2). Voor een beschrijving zie paragraaf 1.2.

Tot slot

Van de hier benoemde typologieën spelen er dus twee een funderende rol in de API-strategie voor de zorg:

— het architectuur-onderscheid tussen data-API's en transactie-API's (afbeelding 1);
— het kwaliteits-onderscheid tussen open API's, technisch gestandaardiseerde API's en volledig gestandaardiseerde API's (afbeelding 2).

Beide typologieën betreffen allereerst API-specificaties, maar gelden ook voor de API-implementaties en API-deployments die gebaseerd zijn op zo'n API-specificatie.

2.3 API-paradigma's en technologiestandaarden

Met een *API* ontsluit een softwareapplicatie functionaliteit aan andere softwareapplicaties, zonder die andere applicaties te belasten met hoe dat precies gebeurt. Belangrijk daarbij is hoe beide zijden aankijken tegen functionaliteit of welk *paradigma* zij gebruiken voor functionaliteit. Zo zijn er bijvoorbeeld (zonder te pretenderen een definitieve en complete lijst te geven):

— het berichten-paradigma: via een API stuurt de ene applicatie een bericht naar de andere. Bekende technologiestandaarden voor dit paradigma zijn HL7v2, FHIR Messaging en EDIFACT[10];
— het document-paradigma: via een API stuurt de ene applicatie een document naar de andere. Bekende technologiestandaarden voor dit paradigma voor de zorg zijn HL7-CDA[11], XDS[12] en FHIR Documents;
— het procedure-paradigma: via een API zet de ene applicatie een procedure bij de andere in gang. Een bekende technologiestandaard voor dit paradigma is SOAP[13];
— het resource-paradigma: via een API spreekt de ene applicatie een gegevensbron aan bij de andere. De API wordt dan RESTful genoemd. Bekende technologieën voor dit paradigma zijn OpenAPI[14] en – voor de zorg – HL7-FHIR[15];
— het query-paradigma: via een API bevraagt de ene applicatie een samenhangend geheel van gegevensbronnen bij andere. Bekende API-technologieën in dit paradigma zijn SPARQL[16] en GraphQL[17].

Een keuze voor een API-paradigma is voor de software-architectuur van groot belang en verdient expliciete aandacht. De paradigma's verschillen vooral in de verdeling van basisverantwoordelijkheden tussen server en client, bijvoorbeeld voor het bijhouden van status of het samenstellen

[10] https://www.nictiz.nl/standaarden/edifact/
[11] https://www.nictiz.nl/standaarden/hl7-cda-r2/
[12] https://www.nictiz.nl/standaarden/xds/
[13] https://nl.wikipedia.org/wiki/SOAP_(protocol)
[14] ook wel Swagger genoemd en niet te verwarren met de term open API die deze API-strategie hanteert voor de eerste trede van het API-groeipad (afbeelding 2)
[15] https://www.nictiz.nl/standaarden/hl7-fhir/
[16] https://www.w3.org/TR/rdf-sparql-query/
[17] https://graphql.org

van data uit onderdelen. In het ene paradigma ontslaat de server de client meer van dergelijke verantwoordelijkheden dan in het andere paradigma, maar trekt daarmee ook meer controle naar zich toe. Bij elk paradigma hoort een eigen verzameling technische standaarden die de verschillende paradigma-kenmerken standaardiseren. Voorbeelden zijn communicatieprotocollen en coderingsstandaarden. Maar vaak gebruiken verschillende paradigma's dezelfde technische standaarden.

De geschiedenis laat een evolutie zien in de populariteit van de paradigma's: van het bericht-paradigma, via het document-paradigma naar het momenteel populaire resource-paradigma. Ondertussen lijkt het query-paradigma aan populariteit te winnen. Door deze wereldwijde en zorgoverstijgende dynamiek in API-paradigma's kan en moet een API-strategie voor de zorg met een rollende API-technologie-agenda (paragraaf 1.4) de wereldwijde API-ontwikkelingen volgen en vertalen naar de actuele behoefte van het Nederlandse zorginformatiestelsel. De API-eisen zullen waar nodig aangeven voor welk paradigma of welke technologie zij zijn bedoeld.

2.4 Infrastructuur

API's zijn ook in te delen naar de infrastructuren waarover zij werken. De API-strategie voor de zorg richt zich allereerst op API's over infrastructuren die zich laten gebruiken volgens het HTTP-protocol[18]. Het meeste internetverkeer is HTTP-verkeer. Het internet wordt echter ook gebruikt via andere application-layer[19] protocollen zoals SMTP[20] of FTP. Veel API's (zoals die gebaseerd op SOAP) werken over zowel HTTP als SMTP en FTP[21]. RESTful API's werken alleen over HTTP, omdat zij aan HTTP methods functionele betekenis hechten.

De voorlopige nadruk van de API-strategie op HTTP-gebaseerde API's is een pragmatische keuze. Die is ingegeven door behoefte aan (technische) scopebeperking en door de wetenschap dat de meeste API's binnen en buiten de zorg over HTTP worden aangeboden. Het streven is om de API-eisen zo op te zetten, dat uitbreiding eenvoudig mogelijk is, bijvoorbeeld naar op SMTP-gebaseerde eisen.

2.5 Rollen

Om API-eisen te formuleren is een rollenmodel nodig. Een *rol* is een set verantwoordelijkheden, bedoeld om individuele partijen aan te binden of zich te laten binden. Een partij kan elke combinatie van rollen spelen, behalve waar dat nadrukkelijk wordt uitgesloten.

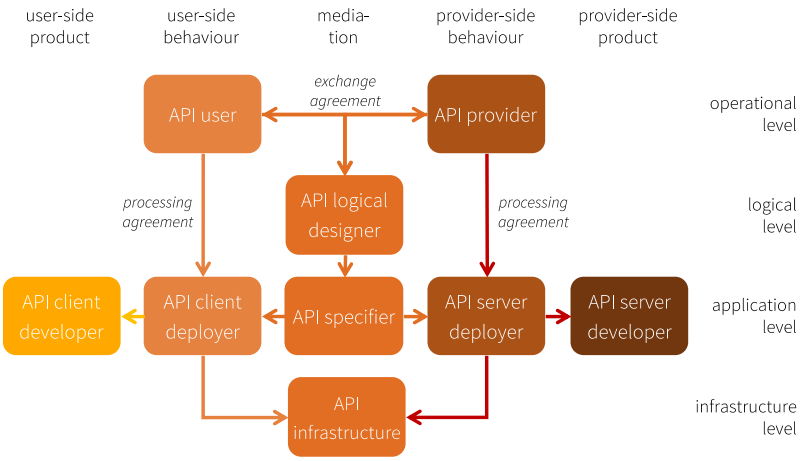
Afbeelding 4 toont een rollenmodel dat een detaillering is van de rollen die al in afbeelding 3 stonden genoemd. De API user en API-provider zijn samen de API agreement holder. De pijlen staan voor rol-relaties. Een eenzijdige pijl van rol A naar rol B betekent dat de verantwoordelijkheden van rol B die van rol A respecteren (rol-hiërarchie, verticaal). Een tweezijdige relatie staat voor wederzijds respect voor elkaars verantwoordelijkheden (contract-relatie, horizontaal).

[18] https://nl.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[19] Deze term komt uit het OSI-model en moet niet worden verward met de applicatielaag van bijvoorbeeld het Nictiz-vijflagenmodel. OSI gaat alleen over infrastructuur: de application-layer protocollen zijn de protocollen die binnen de infrastructuur het dichtst tegen de applicaties aanliggen en hen in zekere zin 'direct' van dienst zijn.

[20] https://nl.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

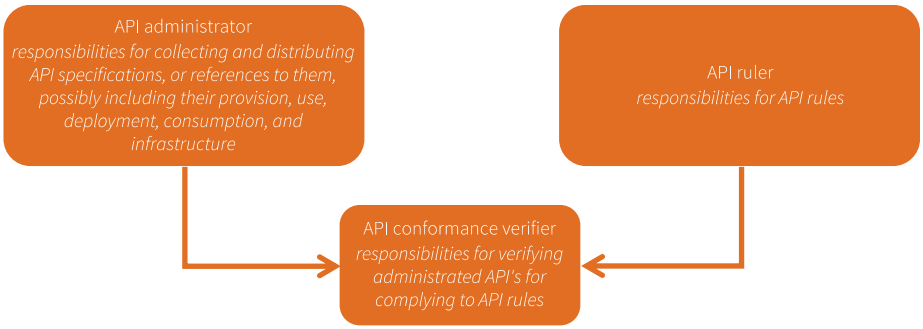
[21] https://nl.wikipedia.org/wiki/File_transfer_protocol



Afbeelding 6: Rollen in de API-strategie.

Verticale relaties weerspiegelen de gelaagdheid van het zorginformatiestelsel: conceptueel-logisch-applicatie-infrastructuur. Horizontaal onderscheidt het model de gebruikende kant (links) van de aanbiedende kant (rechts), met per kant een gedragskolom met erachter een productkolom. In het midden een bemiddelende kolom. De verantwoordelijkheden van deze rollen staan beschreven in Bijlage C.

De API-strategie kent enkele meta-rollen met verantwoordelijkheden voor de API-strategie zelf: verantwoordelijkheid voor administratie van API's in een bibliotheek, verantwoordelijkheid voor het opstellen van de API-eisen en verantwoordelijkheid voor het verifiëren van API's.



Afbeelding 7: Meta-rollen in de API-strategie.

Een API-eis wordt opgelegd aan één rol. Hierop zijn twee uitzonderingen:

— Wanneer een eis de interactie betreft tussen server- en client-zijde kunnen de betreffende varianten van developer- en deployer-rollen samen worden aangesproken.
— Wanneer een eis de afhankelijkheid betreft tussen verschillende verschijningsvormen van API's (bijvoorbeeld tussen API-implementatie en API deployment) kunnen bijbehorende rollen samen worden aangesproken.

2.6 Principes en uitgangspunten

Voor de uitwerking van de API-strategie gelden principes en uitgangspunten (zie tabel 1).

	Aspect of kern-merk	Principe	Zie
1	rolvastheid in het stelsel	Een API-specificatie laat kenmerken die niet bij die API horen (op grond van afbeelding 1) over aan toepasselijke andere elementen van het zorginformatiestelsel. Dit gebeurt ook als de API afhankelijk is van deze elementen en ook als die elementen nog ontbreken.	paragraaf 1.1
2	scheiding van logica	De technische API-specificaties zijn gescheiden van de logische (gegevens- en transactie)modellen die zij implementeren (afbeelding 1).	paragraaf 1.1
3	kwaliteitsniveaus	De API-strategie hanteert drie niveaus (treden) voor de mate waarin een API voldoet aan een set API-eisen (afbeelding 2).	paragraaf 1.2
4	implementatie-onafhankelijkheid	Een API-specificatie is op het hoogste kwaliteitsniveau onafhankelijk van specifieke API-implementaties (software).	paragraaf 1.1
5	scheiding van infrastructuur[22]	Een API is gescheiden van de infrastructuur waarover de API wordt afgehandeld (afbeelding 1).	paragraaf 2.4
6	eisen en rollen	Elke API-eis hoort bij voorkeur bij één rol of bij zo min mogelijk rollen uit het rollenmodel (afbeelding 4).	paragraaf 2.5
7	autorisatiebeleid	Elke API deployment functioneert onder een autorisatiebeleid (restricted use API's).	paragraaf 2.2
8	paradigma-afhankelijkheid	Een API-eis kan passen bij één of meer API-paradigma's.	paragraaf 2.3

Tabel 1: Principes voor de API-strategie voor de zorg.

[22] Deze term is in het veld onderhavig aan aanzienlijke vervaging. Hier wordt met deze term niet geappelleerd aan pretenties van brede toepassing of monopolisering van functionaliteit, maar aan het resultaat van het wegfilteren van applicatie-afhankelijkheid uit functionaliteit.

2.7 API's voor databeschikbaarheid

API's zijn van strategisch belang voor databeschikbaarheid in het licht van het zorginformatiestelsel (paragraaf 1.1). Men kan overwegen om als principe op te nemen dat applicatiesystemen hun data, en eventueel sommige transacties, moeten aanbieden via een API (tabel 1), een API die bovendien moet voldoen aan de API-eisen. Zo wordt bestreden dat applicatiesystemen hun data gevangen houden binnen hun systeemgrenzen (*information blocking*).

De API-strategie kan echter het voor het zorginformatiestelsel strategische doel databeschikbaarheid niet alleen dragen, ook al verzorgt de API-strategie wezenlijke technische voorwaarden voor databeschikbaarheid. Hiervoor zijn twee redenen.

1. Ten eerste is databeschikbaarheid een informatie-inhoudelijke doelstelling die om afspraken vraagt op de hogere niveaus van afbeelding 1 (paragrafen 1.1 en 1.2). Anticiperend op zulke afspraken kan een API-strategie voorwaarden scheppen op systeemniveau. Bovendien kan zij de noodzaak van zulke hoger-niveau afspraken urgenter maken, door technologisch gedreven innovatie te stimuleren. Maar via een louter technologische weg kan de API-strategie geen databeschikbaarheid bewerkstelligen.
2. Ten tweede vraagt databeschikbaarheid uiteindelijk om een meer genuanceerde strategie dan een verplichting tot het ontsluiten van alle data van alle applicaties. Een passende strategie voor databeschikbaarheid moet zeggen welke verantwoordelijken vanuit welke soorten applicatie welke informatie in welke situaties voor welke doelen ter beschikking moeten stellen aan welke anderen onder welke voorwaarden. Hoezeer men ook vanuit secundaire informatiebehoeften ongespecificeerd aanspraak doet op databeschikbaarheid. Zo'n passende strategie voor databeschikbaarheid moet men nastreven met een brede stelselstrategie, een stelselregie die stoelt op een uitgewerkte visie op informatie(her)gebruik.

Hoe kan deze API-strategie toch − binnen haar eigen grenzen − maximaal bijdragen aan databeschikbaarheid? Daarvoor helpt het de afhankelijkheid van te specifieke informatiebehoeften te beperken (punt twee hierboven). Dat kan door je te concentreren op generieke informatiebouwstenen en databeschikbaarheid te bevorderen per bouwsteen met een API voor die informatiebouwsteen. Dit vraagt om een strategische samenwerking met de huidige parallel lopende ontwikkeling van een zib-strategie. Dat vereist dan om, op basis van een generiek conceptueel informatiemodel, informatiebouwstenen-nieuwe-stijl aan te wijzen en te definiëren. Daarna kan per bouwsteen een API gespecificeerd worden. De zib-strategie vertegenwoordigt zo de conceptuele en de logische niveaus van afbeelding 1 en vult daarmee het eerste punt van hierboven in: de afhankelijkheid van hogere niveaus.

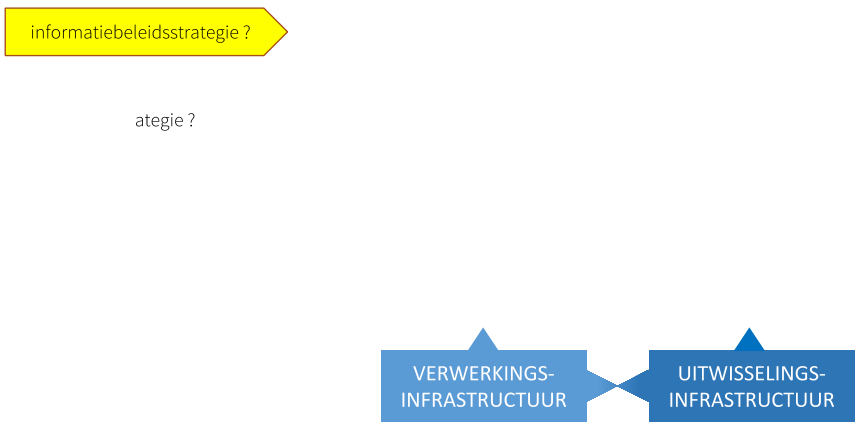
Binnen de API-strategie kunnen zich − na identificatie van informatie-bouwstenen-nieuwe-stijl − typen API's aandienen voor opname in de API-bibliotheek. Deze opname start op de eerste trede. Daarbij vormen Wegiz-verplichtingen een belangrijke stimulans. Voor doorgroei naar de derde trede (volledig gestandaardiseerde API's) is nodig dat die informatiebouwstenen-nieuwe-stijl niet alleen *geïdentificeerd* zijn, maar ook *gespecificeerd* en *afgebeeld* op een API-specificatie.

Zie ook Bijlage A.

2.8 Naar samenhangende deelstrategieën

De API-strategie is gepositioneerd op het systeemniveau (afbeelding 1). Dat biedt houvast bij de vormgeving van de API-strategie. Dat begrenst het werkingsgebied van de API-strategie, maar ook wat ervan verwacht mag worden. Die begrenzing is een natuurlijke voorwaarde voor de effectiviteit van de API-strategie, in relatie tot omliggende strategieën.

Hier kan een landschap worden geschetst van inhoudelijke strategieën waarvan de API-strategie er één is. Dit landschap volgt de niveaus van het specificatiecanvas uit de stelselarchitectuur-visie[23]. Elk niveau beschouwt een aspect van alle informatie die omgaat in het zorginformatiestelsel (afbeelding 6).



Afbeelding 8: Vijf deelstrategieën.

Hierbij horen enkele toelichtende opmerkingen. De zib-strategie is pril. Hoewel het denken zich beperkt heeft tot het logische niveau, ontstaat het bewustzijn dat een gezamenlijk conceptueel model nodig is. Het conceptuele gehalte van dit model zal bepalen of de zib-strategie zich ook op dat niveau gaat begeven, of dat een aparte deelstrategie voor het conceptuele niveau nodig is.

Een informatiebeleid-strategie en infrastructuurstrategie worden nog niet als zodanig onderkend. Elementen hiervan kleuren evengoed al jaren het debat over het zorginformatiestelsel. Deze twee niveaus zijn het minst afhankelijk van de specifieke complexiteit van informatie en daarom wellicht het snelst en eenvoudigst opgeworpen. Daaronder zou de verwachting kunnen schuilgaan dat met deze 'deksel' en 'bodem' van het specificatiecanvas ook de uitdagingen van de middelste drie niveaus kunnen worden beslecht. Dat zou een onderschatting zijn. Het is eerder andersom: organisatie en infrastructuur zijn letterlijk de sluitstukken van het stelsel. De informatie in het midden vormt de kern.

De term *infrastructuur* is aanzienlijk vertroebeld in het debat over het zorginformatiestelsel. In het kader van de API-strategie verstaan we daaronder alle systemen waarover API's worden afgewikkeld, ongeacht om welke API het precies gaat. De term infrastructuur zegt zo dus nog niets over hoeveel partijen haar (zouden moeten) gebruiken.

[23] Deze zal binnenkort verschijnen.

2.9 Eerder adviesrapport

Het adviesrapport[Fout! Bladwijzer niet gedefinieerd.] dat de aanleiding is voor deze API-strategie, doet zes aanbevelingen. Aanvullend staat hier hoe deze API-strategie zich daartoe verhoudt.

1. Stimuleer de ontwikkeling van een *API-strategie voor de zorg* en zorg daarbij voor een brede vertegenwoordiging van het veld. Hieraan geeft deze API-strategie opvolging.
2. Stimuleer vroegtijdige opname van API's in de API-bibliotheek om de meerwaarde van een API-bibliotheek en mogelijkheden van hergebruik aan te tonen. Dit krijgt opvolging in paragraaf 1.4 en in een apart document.
3. Stuur op een groeimodel van niet-gestandaardiseerde API's naar gestandaardiseerde API's zodat zowel de innovatiekracht van leveranciers als de interoperabiliteit in het stelsel is geborgd. Dit krijgt opvolging in paragrafen 1.2 en 1.4 en in een apart document met API-eisen.
4. Onderken de verschillen tussen zibs, API's, applicaties en infrastructuur en zorg voor borging in het duurzame informatiestelsel. Dit krijgt in paragraaf 2.8 opvolging.
5. Onderzoek welke onderdelen van de API-strategie normatief kunnen worden verklaard. In de stelselarchitectuur-visie wordt wet- en regelgeving als sluitstuk gezien van beleidsvorming. Daar waar noodzakelijke kwaliteiten meer vragen dan regulier beleid, is *normering* aan de orde.
6. Zorg ervoor dat normen voor generieke functies leiden tot implementaties die breed bruikbaar zijn in de zorg, zodat ook API's daarvan gebruik kunnen maken. Dit moet onderdeel zijn van een strategie voor generieke functies, al dan niet als onderdeel van een brede strategie voor het zorginformatiestelsel.

Hetzelfde adviesrapport somt drie groepen adviezen op die gelden als vertrekpunt voor de uitwerking van de API-strategie en haar middelen. De eerste groep adviezen vloeit voort uit de wensen van zorgprocessen en zorgaanbieders (tabel 2) en vraagt daarom allereerst om afspraken op de hogere niveaus (afbeelding 1). Afspraken die doorwerken op API's (zie paragraaf 2.7).

Een tweede groep adviezen vloeit voort uit de wensen van softwareleveranciers (tabel 3). Aan deze groep wensen komen de in hoofdstuk 1 genoemde strategische middelen van de API-strategie tegemoet: de API-eisen, de API-bibliotheek en de API-technologie-agenda.

Voor een laatste groep wensen over het zorginformatiestelsel (tabel 4) geldt hetzelfde als voor de eerste groep. Veel adviezen in deze tabellen worden dus geraakt met de vierde hoofdaanbeveling: onderken de verschillen tussen zibs, API's, applicaties en infrastructuur en zorg voor borging in het duurzame informatiestelsel.

	Advies
ontwerp	Ontwerp API's zo dat ze een betekenisvolle set van gegevens ontsluiten.
	Zorg voor flexibiliteit in de gegevens die API's ontsluiten. API's bevatten voldoende filtermogelijkheden om de juiste informatiebehoefte te dekken en voldoende selectiemogelijkheden op de gegevenselementen die beschikbaar gesteld worden. Houd hierbij rekening met autorisatie, capaciteit, doseerbaarheid, et cetera.
	Ontsluit in een API gegevens die voldoende rijk zijn. Daarmee kan men eenvoudig doorvragen naar aanvullende informatie en makkelijk vervolgvragen stellen en beantwoorden.
	Bied API's aan die nieuwe of gewijzigde gegevens actief uitsturen (push) om afhankelijke toepassingen en processen te voeden (bijvoorbeeld Acute zorg, AI-modellen of datawarehouses).
	Geef in API-specificaties informatie over de betrouwbaarheid van de gegevens en transparantie over het toegangsbeleid. Afnemers om commerciële redenen uitsluiten is niet toegestaan.
toepassing	Maak API's herbruikbaar voor verschillende toepassingen zoals primaire zorg, managementinformatie, kwaliteitsregistraties en wetenschappelijk onderzoek.
	Geef de API-eisen dusdanig vorm dat ze van toepassing zijn op API's van informatiesystemen, (regionale) platformen en portalen.
	Bied (convenience) API's aan voor bijvoorbeeld wetenschappelijk onderzoek of kwaliteit die een mogelijkheid bieden voor anonieme of gepseudonimiseerde ontsluiting van gegevens.
beleid	Werk uit hoe API's gebruik maken van generieke functies en voorzieningen voor bijvoorbeeld identificatie, authenticatie, autorisatie, toestemming en logging. Ook leveranciers hebben deze behoefte geuit. De functies zijn nodig om te voldoen aan wet- en regelgeving bij gegevensuitwisseling.
	Geef richting aan de keuzes op de applicatie-laag van het meer-lagenmodel. Daarbij is de keuze voor internationale standaarden zoals Open API en HL7 FHIR preferent en kunnen we leren van ervaringen uit andere landen.
	Bied kaders en handreikingen aan de technische specificaties voor het ontsluiten van zowel losse concepten (bijvoorbeeld zibs) als composities (bijvoorbeeld medicatieoverzicht of zwangerschapskaart).
	Wees leveranciers-onafhankelijk en voorkom information blocking. API-eisen en API-specificaties in de bibliotheek trekken geen leveranciers voor en zijn niet beperkt tot gebruik door één leverancier. Leveranciers onderscheiden zich op bijvoorbeeld functionaliteit, user interface en dienstverlening. Zorgaanbieders houden de keuzevrijheid om van verschillende aanbieders gebruik te maken.

Tabel 2: Geadviseerde eisen van zorgprocessen en aanbieders.

	Advies
ontwerp	Maak gebruik van internationale standaarden en sluit aan bij ontwikkelingen die in het buitenland spelen. HL7 FHIR mag niet ontbreken in de strategie; voor bestaande API's die gebaseerd zijn op andere standaarden moeten keuzes worden gemaakt en een migratie-strategie worden geformuleerd.
	Maak eenduidige afspraken over releasemanagement en versionering van API's.
	Verplicht het beschikbaar stellen van een testomgeving voor API's.
	Onderzoek de mogelijkheden voor een gedeelde ontwikkelomgeving of tooling.
beleid	Stimuleer het hergebruik van API's voor verschillende informatiebehoeften.
	Bied ruimte voor innovatie en standaardisatie door ondersteuning van zowel gestandaardiseerde als niet-gestandaardiseerde API's.
	Stimuleer hergebruik van bestaande API's en ondersteun daarin ook het doorgroeien van niet-gestandaardiseerde API's naar gestandaardiseerde API's.
	Stel API-specificaties kosteloos ter beschikking. Onderzoek voor het gebruik van API's of publieke of private afspraken nodig zijn.

Tabel 3: Geadviseerde eisen van leveranciers.

	Advies
ontwerp	Baseer de beschrijving van de gegevens die API's aanbieden op bouwstenen uit de informatie-laag (zibs).
	Baseer de maatvoering van API's op een of meerdere zibs in combinatie met de maatvoering in de gebruikte communicatiestandaard.
toepassing	Gebruik bij de ontwikkeling van informatiestandaarden zoveel mogelijk bestaande API's.
	Gebruik filters en selecties op bestaande API's voor een toepassing-specifieke invulling in plaats van het ontwikkelen van nieuwe API's.
	Introduceer hergebruik om kwalificatie te vereenvoudigen: splitskwalificatie van de techniek door generieke API's en de toepassings-specifieke invulling aanvullend op elkaar te kwalificeren.
beleid	Beschrijf in de API-strategie hoe API's zich verhouden tot andere onderdelen in het duurzame informatiestelsel.
	Breid de huidige bouwstenen op de informatie-laag uit zodat ontbrekende gegevens (bij-voorbeeldprocesgegevens, contextgegevens, identificaties en relaties naar andere zibs) beschikbaar zijn voor toepassing in API-specificaties.

Tabel 4: Geadviseerde eisen in relatie tot het zorginformatiestelsel.

Bijlage A

API's en informatie-bouwstenen: voorbeeld

Een vijfenvijftigjarige vrouw gebruikt haar Persoonlijke Gezondheidsomgeving (PGO) om een verzameling zorggegevens over zichzelf te downloaden uit het dossier van haar huisarts. Dit geeft haar in één oogopslag een beeld van haar gezondheid en wat daarover in het dossier bekend is. Bovendien kan zij deze gegevens delen met andere zorgaanbieders bij wie zij onder behandeling is of komt. Wanneer de PGO, de huisarts en die andere zorgaanbieders daarvoor de MedMij-afspraken gebruiken, kan dit interoperabel en vertrouwd.

In het kader van MedMij zijn hiervoor zogenoemde *gegevensdiensten* gestandaardiseerd en opgenomen in een catalogus[24]. Haar huisarts heeft besloten de gegevensdienst *Verzamelen Basisgegevens zorg 3.0* aan te bieden en de PGO heeft besloten die voor haar gebruikers ook af te nemen, niet alleen van deze huisarts, maar van alle zorgaanbieders die in MedMij-verband deze gegevensdienst aanbieden.

Om dit mogelijk te maken hebben de PGO-leverancier en de applicatiedienstverlener van de huisarts een standaard moeten laten inbouwen. De informatiestandaard bij deze gegevensdienst heet de *Basisgegevensset zorg*[25] en kent maar liefst 28 verschillende stukjes informatie, die de vrouw dus ineens uit het huisartsdossier kan downloaden. Voor elk van die 28 stukjes, de zogenoemde informatiebouwstenen, is een vertaling gemaakt naar een API. Die API's zijn geïmplementeerd op de PGO en op het dossiersysteem van de huisarts, elk voor zijn respectievelijke helft: de afnemers- of de aanbiederszijde.

Doordat die specificaties gestandaardiseerd zijn, kunnen alle PGO's die dat willen de betreffende gegevens voor hun gebruikers ophalen bij alle zorgaanbieders die die gegevens aanbieden. Dat gebeurt dan steeds in het vertrouwde MedMij-kader. Maar de standaardisatie- en implementatie-inspanning voor deze API's kan ook nuttig blijken voor andere contexten dan MedMij alleen. Zo is het denkbaar dat dezelfde gegevensset ook rechtstreeks tussen zorgaanbiedersdossiers moet kunnen worden uitgewisseld, nog steeds onder passende afspraken, maar in een andere context dan de huidige MedMij-afspraken. Die kans wordt des te groter als daarbij niet steeds alle 28 API's tegelijk aan de orde zijn, maar een selectie ervan, passend bij een specifieke uitwisselbehoefte.

Het zou van de betreffende zorgaanbieders en hun applicatiedienstverleners dan veel vergen om voor elke nieuwe context en selectie opnieuw een standaard te moeten implementeren, en zich daarop te moeten kwalificeren. Voor de standaardisatie-organisatie Nictiz vergt het bovendien ook veel om voor elke nieuwe context en selectie een nieuwe standaard te moeten ontwikkelen en beheren. De tijd en kosten die dit vergt kunnen een stevige wissel trekken op het tempo en zelfs de realiseerbaarheid überhaupt van al die behoeften aan gegevensuitwisseling.

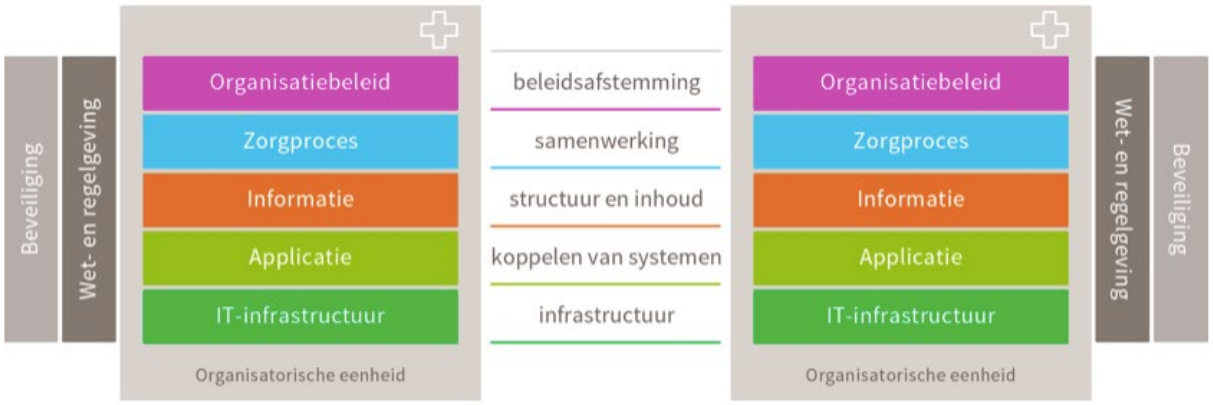
Dat roept de vraag op of de standaardisatie-, implementatie- en kwalificatie-inspanning zich niet zou moeten richten op elke API apart, zodat daarna API's naar believen kunnen worden toegepast en gecombineerd in allerlei contexten zonder die inspanningen te hoeven herhalen.

[24] https://afsprakenstelsel.medmij.nl/display/MMCatalogus/Actuele+gegevensdiensten
[25] https://www.nictiz.nl/standaarden/basisgegevensset-zorg/

Bijlage B

Toelichting op de verfijning van het vijfla-genmodel

Met zijn vijf niveaus roept het model in afbeelding 1 de vraag op hoe het zich verhoudt tot het bekende lagenmodel van Nictiz[26]. Met dat model bestaan vergaande overeenkomsten en geen tegenstrijdigheden. Afbeelding 1 is op meer manieren een wezenlijke verfijning van het Nictiz lagenmodel.



Afbeelding 9: Lagenmodel van Nictiz.

Ten eerste wordt het niet omringd gedacht door organisatie- of systeem-grenzen. Het is niet een gelaagd model van organisaties of systemen, maar een model dat op alle schalen van toepassing is, en gebruikt wil worden als achtergrond waartegen *elke* specificatie in het zorginformatiestelsel zijn plek krijgt, of het nu op Europese schaal is, of op de schaal van vele federatieve niveaus kleiner. Zo geeft het alle specificaties hun plaats en hun verband, door verschillende graden van algemeenheid en bijzonderheid heen.

Ten tweede brengt het ook een horizontaal onderscheid aan tussen verwerking en uitwisseling. Een onderscheid dat van wezenlijk belang is voor de API-strategie en bijvoorbeeld haar bijdrage aan informatiebeschikbaarheid.

Ten derde kent afbeelding 1 geen zorgorganisaties en zorgprocessen (althans niet als toplagen) zoals in het Nictiz lagenmodel. Toch staan zij wel degelijk in het model, namelijk als de betekenis van de taal: op taalniveau. Afbeelding 1 gaat louter over informatie en alleen *via de betekenis van informatie* (taalniveau) over zorg. Het zorgstelsel is dus de betekeniswereld van die informatie, maar geen specificatiedoel voor het zorg*informatie*stelsel zelf; natuurlijk wel voor het zorgstelsel. De zogenoemde zorgkwaliteitsstandaarden[27] bijvoorbeeld moeten daarom niet tegen de achtergrond van afbeelding 1 worden gespecificeerd. Andersom moeten de informatiespecificaties, waaronder de API-specificaties, dus ook niet rechtstreeks in de zorgkwaliteitsstandaarden worden opgenomen.

De relatie tussen zorgstelsel en zorginformatiestelsel is niet een geheel-deel-relatie, maar (saillant genoeg) zelf een zorgrelatie: het zorginformatiestelsel *zorgt* voor informatie voor zorg. Dat betekent ook dat informatie niet zozeer een materiaal is dat door zorgprocessen 'stroomt', maar dat informatie 'zorgt voor betekenis' in zorgprocessen[28]. Anders is het waardeloos.

[26] https://www.nictiz.nl/standaardisatie/interoperabiliteit/

[27] https://www.zorginstituutnederland.nl/over-ons/commissies/advies-en-expertgroep-kwaliteits-standaarden-aqua

[28] Dit zorgt er onder andere voor dat informatiebouwstenen als strikt logische eenheden moeten gaan worden gezien en niet, zoals nog gebruikelijk, ook als concepten.

Ten slotte onderkent afbeelding 1 drie niveaus binnen wat in het Nictiz lagenmodel één laag is: de informatie-laag in het midden. Die laag wordt in het canvas opgedeeld (van wat vooralsnog vooral als logische laag is behandeld) tot ook een conceptueel niveau (taal) en een organisatieniveau[i]. Zo zijn er toch weer vijf niveaus, maar niet dezelfde als de eerdere lagen. De onderste twee lagen komen wel overeen, zij het dat het Nictiz lagenmodel het horizontale onderscheid tussen verwerking en uitwisseling niet kent.

En zo verschijnt er in het specificatiecanvas ook een organisatieniveau maar dat gaat specifiek over *informatie*-organisatie. Informatie is een middel dat – hoe dienstbaar ook aan hogere zorgdoelen – onherroepelijk zijn eigen aspectbeleid vraagt onder de hoede van CIO-achtige verantwoordelijkheden. Zolang informatie die aandacht moet missen, zal een duurzaam effectief zorginformatiestelsel buiten bereik blijven.

Bijlage C

Beschrijving van de API-rollen

API role	responsibilities
API client deployer	technical responsibilities for employing an API, as deployed by the API server deployer, and specified by the API specifier, thus implementing the end responsibilities of the API user
API client developer	technical responsibilities for supplying software for the API client deployer
API infrastructure	technical responsibilities for conveying specified API's between API clients and API servers
API logical designer	responsibilities for logically specifying both data and operations to be implemented in the API
API provider	end-responsibilities for providing the value and meaning of an API, as agreed with API users
API server deployer	technical responsibilities for deploying an API, as specified by an API specifier, thus implementing the end responsibilities of the API provider
API server developer	technical responsibilities for supplying software for the API server deployer
API specifier	responsibilities for technically specifying the API so that an API server deployer knows what to deploy and an API client knows what to employ
API user	end-responsibilities for using the value and meaning of an API, as agreed with the API provider

Tabel 5: Beschrijving van API-rollen[29].

[29] Omdat de rollen eenheden zijn waarin verantwoordelijkheden worden geordend in het Engelstalige document met de API-eisen, is ook deze tabel in het Engels.

Nictiz is de Nederlandse kennisorganisatie voor digitale informatievoorziening in de zorg. Nictiz ontwikkelt een visie op het zorginformatiestelsel en de architectuur die dat stelsel ondersteunt. We ontwikkelen en beheren standaarden die digitale informatievoorziening mogelijk maken en zorgen ervoor dat zorginformatie eenduidig kan worden vastgelegd en uitgewisseld. Daarnaast adviseren we en delen we kennis over digitale informatievoorziening in de zorg. Daarbij kijken we niet alleen naar Nederland, maar ook naar wat er internationaal gebeurt.

Nictiz | Postbus 19121 | 2500 CC Den Haag | Oude Middenweg 55 | 2491 AC Den Haag
070 - 317 34 50 | www.nictiz.nl

Nictiz
Betere gezondheid
door betere informatie