Inhoud

Voorwoord

De bouw van een populair ontwikkelaarsplatform met een API die door miljoenen ontwikkelaars wordt gebruikt, is een van de opwindendste uitdagingen in uw hele softwarecarrière. In dit boek leert u hoe u dat moet doen.

API's staan centraal in de moderne softwareontwikkeling. Ze tackelen een fundamenteel ontwikkelaarsprobleem: hoe kan ik als software-engineer de code die ik heb geschreven vrijgeven aan andere ontwikkelaars om ermee te innoveren? Software programmeren lijkt in de moderne wereld veel op het bouwen met legostenen. Als ontwikkelaar hebt u toegang tot een uitgebreide set API's die services aanbieden, zoals betalingen, communicatie, autorisatie en authenticatie enzovoort. Bij het programmeren van nieuwe software is het uw taak als software-engineer om met deze API's uw nieuwe product samen te stellen. Om tijd te besparen hergebruikt u daarbij code van anderen. Waarom zou u het wiel opnieuw uitvinden?

Veel software-engineers die als kind graag met Lego speelden, spelen er vandaag de dag nog altijd graag mee. En wie niet? Het is leuk en je kunt nuttige dingen bouwen met van die kleurrijke blokjes die naadloos op elkaar aansluiten. Maar wat als je het zelf zou kunnen bouwen? Zou het niet geweldig zijn als je niet alleen nieuwe legokits kon uitvinden, maar ook de lego-onderdelen zelf, en anderen ermee liet innoveren? Wanneer u uw eigen API bouwt, maakt u in feite uw eigen lego-onderdelen voor andere ontwikkelaars.

API's zijn geen nieuw concept in de informatica – in de jaren zestig begonnen ontwikkelaars standaardbibliotheken voor de eerste programmeertalen aan te leggen en deze met andere ontwikkelaars te delen. Ontwikkelaars gebruikten de standaardfunctionaliteit van bibliotheken zonder daarvan de interne code te kennen.

Met de opkomst van op een netwerk aangesloten computers in de jaren zeventig en tachtig kwamen de eerste netwerk-API's die services ter beschikking stelden waar ontwikkelaars via Remote Procedure Calls (RPC's) gebruik van maakten. Met RPC's konden ontwikkelaars hun functies over het netwerk aanbieden en externe bibliotheken aanroepen alsof ze lokaal waren. Programmeertalen zoals Java brachten verdere abstractie en complexiteit, met messaging middlewareservers die deze externe services inventariseerden en integreerden.

In de jaren negentig, met de opkomst van het internet, wilden veel bedrijven de manier waarop we API's bouwen en vrijgeven standaardiseren. Standaarden zoals de Common Object Request Broker Architecture (CORBA), het Component Object Model (COM) en het Distributed Component Object Model (DCOM) van Microsoft, en vele andere, streefden ernaar dé manier te worden om services over het internet aan te bieden. Het probleem was dat het beheer van de meeste van deze standaarden complex was, vergelijkbare programmeertalen aan beide zijden van het netwerk verplicht stelden en soms de lokale installatie van een deel van de externe service (gewoonlijk een _stub_ genoemd) vereisten om toegang te krijgen. Het was een rommeltje; de droom van interoperabiliteit werd al snel een nachtmerrie van configuraties en beperkingen.

Aan het einde van de jaren negentig en begin 2000 kwamen er meer open en standaard manieren om toegang te krijgen tot externe diensten via internet (web-API's). Eerst met het Simple Object Access Protocol (SOAP) en XML (Extensible Markup Language), en vervolgens met Representative State Transfer (REST) en JavaScript Object Notation (JSON), werd toegang tot services eenvoudiger en gestandaardiseerder zonder afhankelijkheden van code op de client of programmeertaal. We behandelen de populairdere en meer bruikbare methoden in dit boek.

Een voor een begonnen technologiebedrijven nuttige services aan te bieden via API's – vanaf de begindagen van de Amazon Affiliate-API (2002), de Flickr-API (2004), de Google Maps-API (2005) en de Yahoo! Pipes-API (2007) zijn er nu duizenden API's die elke denkbare service aanbieden, van beeldmanipulatie tot kunstmatige intelligentie. Ontwikkelaars kunnen deze aanroepen en nieuwe producten maken die samengesteld zijn uit meerdere API's, net als bouwen met legostenen.

API's zijn handelswaar geworden. De toepassing is eenvoudig, maar het bouwen van een API is nog altijd hoge kunst. Neem deze uitdaging niet licht op. Het bouwen van een solide API is niet gemakkelijk. API's moeten briljant eenvoudig en in hoge mate interoperabel zijn, net als Lego; elk onderdeel van elke kit moet goed passen op elk ander stuk in een andere kit. API's moeten ook geleverd worden met ontwikkelaarsprogramma's en resources om ontwikkelaars te helpen bij de implementatie. Het ontwikkelen van een solide API is slechts de eerste stap; u moet ook een bloeiend ecosysteem van ontwikkelaars creëren en ondersteunen. We behandelen deze uitdagingen in het laatste gedeelte van dit boek.

We hebben dit boek geschreven omdat we ons realiseerden dat we gedurende onze loopbaan geconfronteerd werden met vergelijkbare processen, beslissingen, procedures en optimalisaties voor veel API's, maar dat deze richtlijnen nog niet waren samengevoegd in één gezaghebbende bron. We zouden elk kunnen wijzen naar verspreide blogposts en artikelen over afzonderlijke onderwerpen, maar er is niet één centrale plek waar we meer te weten konden komen over ontwerpmethoden die de evolutie en groei van web-API's en hun ecosystemen stimuleren. Met dit boek hopen we u alle gereedschappen in handen te geven die we hebben gemaakt en ontdekt in de loop van onze carrières in API's. Toegang hebben tot deze vaardigheden is uitermate waardevol. Het kan het verschil maken tussen slagen en falen van uw bedrijf of technologie, het kan net dat unieke voordeel bieden dat de motor is van uw carrière.

1 Hoofdstuk

Wat is een API?

Wat is een API? Wanneer een beginnend programmeur deze vraag stelt, krijgt hij meestal als antwoord: "Een interface voor het programmeren van toepassingen". Maar API's zijn zo veel meer dan hun naam doet vermoeden – en om hun kracht te begrijpen én te ontketenen moeten we onze aandacht richten op het sleutelwoord interface.

In dit hoofdstuk:

De definitie van een API en waarom we API's nodig hebben.

De businesscase voor API's.

Wat een API goed maakt.

Definitie van API

Een API is de interface die een softwareprogramma presenteert aan andere programma's, aan mensen en, in het geval van web-API's, via het internet aan de wereld. Een API-ontwerp verhult veel van de programmatuur erachter – bedrijfsmodel, productfuncties, een incidentele bug. Hoewel API's ontworpen worden om met andere programma's samen te werken, zijn ze meestal bedoeld om te worden begrepen en gebruikt door *mensen* die de andere programma's schrijven.

API's zijn de bouwstenen die interoperabiliteit mogelijk maken voor grote zakelijke platforms op internet. API's bieden functionaliteit voor de creatie en handhaving van identiteit tussen cloudsoftwareaccounts, van uw zakelijke e-mailadres en ontwerpsoftware voor samenwerking tot de webapps waarmee u bijvoorbeeld pizza's bestelt. API's bieden de mogelijkheid om meteorologische gegevens van een gerenommeerde bron, bijvoorbeeld het KNMI, te delen met honderden apps die zijn gespecialiseerd in de weergave daarvan. API's verwerken uw creditcardgegevens en stellen bedrijven in staat uw geld probleemloos te innen zonder zich druk hoeven te maken over de details van de financiële technologie en de bijbehorende wet- en regelgeving.

API's zijn steeds vaker een belangrijke component van schaalbare en succesvolle internetbedrijven zoals Amazon, Stripe, Google en Facebook. Voor bedrijven die een bedrijfsplatform willen maken waarmee de markt voor iedereen steeds toegankelijker wordt, vormen API's een belangrijk puzzelstukje.

Het ontwerpen van uw eerste API is nog maar het begin. Dit boek gaat verder dan de ontwerpprincipes voor uw eerste API en laat u tot in detail zien hoe u tegelijk met de groei van uw bedrijf een API ontwikkelt. Met de juiste keuzes zal uw API de tand des tijds doorstaan.

Waarom hebben we API's nodig?

API's zijn voortgekomen uit de behoefte informatie uit te wisselen met dataproviders. Ze hebben de middelen om specifieke problemen op te lossen zodat mensen bij andere bedrijven geen tijd hoeven te steken in het zelf oplossen van die problemen. U wilt bijvoorbeeld een interactieve kaart op een webpagina insluiten zonder Google Maps opnieuw uit te vinden. U wilt dat een gebruiker zich kan aanmelden zonder de Facebook-login opnieuw uit te vinden. Of u wilt misschien een chatbot maken die af en toe met gebruikers communiceert zonder een realtime messagingsysteem te hoeven bouwen.

In al deze gevallen worden aanvullende functies en producten gecreëerd met behulp van de gegevens of interacties van een gespecialiseerd platform. API's stellen bedrijven in staat om snel unieke producten te ontwikkelen. In plaats van het wiel opnieuw uit te vinden, kunnen startups hun productaanbod differentiëren, profiteren van bestaande technologieën en gebruikmaken van andere ecosystemen.

Wie zijn onze gebruikers?

> *De theorie doet er niet toe als je geen aandacht besteedt aan de bouw van het juiste ding voor de juiste klant.*
> — Bilal Aijazi, CTO bij Polly

Bij het maken van een product is het een goed idee om eerst de klant centraal te stellen. Dit is ook erg belangrijk bij het ontwerpen van een API. In hoofdstuk 8 gaan we het hebben over verschillende soorten ontwikkelaars en praktijktoepassingen, en over strategieën voor de omgang met en waardering van hen. Het is belangrijk dat u begrijpt wie uw ontwikkelaars zijn, wat hun behoeften zijn en

waarom zij uw API gebruiken. Door u te richten op ontwikkelaars zult u geen API's bouwen die niemand nodig heeft of die niet voldoen aan de gebruikseisen van uw ontwikkelaars.

Omdat een API-ontwerp erg moeilijk te wijzigen is, is het belangrijk dat u uw API specificeert en valideert, lang voordat u begint met de implementatie ervan. De overstapkosten van het ene API-ontwerp naar de andere zijn voor de meeste ontwikkelaars extreem hoog.

Hier zijn enkele voorbeelden van hoe ontwikkelaars een API voor het uploaden en opslaan van afbeeldingen willen gebruiken:

- Lisa is webontwikkelaar bij een startup die kunst verkoopt; ze zoekt een eenvoudige manier waarmee kunstenaars foto's van hun werk kunnen uploaden en weergeven.
- Ben is backendontwikkelaar die kwitanties uit een onkostensysteem moet opslaan in zijn audit- en beleidsoplossing.
- Jane is frontendontwikkelaar die een realtime chat voor klantondersteuning op de website van haar bedrijf wil integreren.

Dit zijn slechts enkele voorbeelden, elk met unieke verborgen vereisten. Als u niet voldoet aan de behoeften van uw ontwikkelaars, zal uw API geen succes zijn.

De volgende paragraaf gaat over gebruiksscenario's op een hoger niveau die van invloed zijn op het ontwerp van uw API, maar hoe gedetailleerder u bent in uw scenario's en hoe beter u uw ontwikkelaars begrijpt, hoe beter u ze kunt bedienen.

De businesscase voor API's

Het is geen geheim dat het internet de aanjager is van een groot deel van de productinnovatie en technologiesector. Als gevolg hiervan zijn API's belangrijker dan ooit voor het succes van een bedrijf. Er zijn veel modellen om ze in een product op te nemen. In sommige gevallen leidt een API tot het genereren van directe winst en inkomsten (via winstdelingsmodellen, abonnementsgelden of gebruikskosten). Maar er zijn ook vele andere redenen om een API te maken. API's kunnen de algehele productstrategie van uw bedrijf ondersteunen. Ze kunnen een cruciaal puzzelstukje zijn voor het integreren van services van derden in het product van uw bedrijf. API's kunnen onderdeel zijn van een strategie om anderen te stimuleren aanvullende producten te creëren, waarbij de ontwikkelaar van het hoofdproduct niet bereid of in staat is om erin te investeren. API's kunnen ook een manier zijn om leads te genereren, nieuwe distributiekanalen voor een product te openen of producten te verkopen. Zie de presentatie van John Musser over API-bedrijfsmodellen voor meer informatie over deze strategieën: **www.slideshare.net/jmusser/j-musser-apibizmodels2013**.

Een API moet zijn afgestemd op de kernactiviteit, zoals het geval is bij veel bedrijven die software-as-a-service (SaaS) leveren. Bekende voorbeelden zijn GitHub, Salesforce en Stripe. Soms worden de producten die op deze API's zijn gebouwd 'service-integraties' genoemd. Consumer-API's werken goed als er een kritieke massa van door gebruikers gegenereerde inhoud is, zoals met de mogelijkheden voor delen van foto's van Facebook en Flickr. Hoewel er veel redenen zijn om een API te maken en een ontwikkelaarsplatform te starten, is er ook een duidelijke reden om geen ontwikkelaarsplatform te maken, namelijk wanneer de API-strategie niet in lijn ligt met de kernactiviteit. Als bijvoorbeeld advertenties de belangrijkste inkomstenbron van het

product zijn, leiden API's die alternatieve 'clients' voor het product mogelijk maken het verkeer weg van de toepassing waar de advertenties worden gehost. Dat kost inkomsten en dat gebeurde bijvoorbeeld met de Twitter API.

Directe winst en commerciële prikkels daargelaten, laten we eerst eens gedetailleerder kijken naar manieren waarop bedrijven hun API-ontwikkeling hebben gestructureerd:

- API's voor interne ontwikkelaars eerst, externe ontwikkelaars als tweede.
- API's voor externe ontwikkelaars eerst, interne ontwikkelaars als tweede.
- API's als product.

API's voor interne ontwikkelaars eerst, externe ontwikkelaars als tweede

Sommige bedrijven bouwen eerst hun API's voor interne ontwikkelaars en geven deze vervolgens vrij aan externe ontwikkelaars. Daar kunnen een aantal redenen voor zijn. Een voorbeeld is dat het bedrijf potentiële waarde ziet in het toevoegen van een externe API. Dit kan een ecosysteem van ontwikkelaars opleveren, nieuwe vraag naar het product van het bedrijf stimuleren of andere bedrijven in staat stellen producten te maken die het bedrijf zelf niet wil bouwen.

Laten we een specifiek geval nemen en eens kijken naar hoe de API van Slack is gestart – als API voor de berichteninterface van Slack op het web, de desktop en mobiele clients. Hoewel Slack oorspronkelijk zijn API's ontwikkelde voor zijn interne ontwikkelaars, gebeurde er iets anders naarmate het bedrijf groeide: een handvol 'integraties' met belangrijke bedrijfssoftware werden uitermate waardevol voor de groei en ontwikkeling van Slack als communicatiesoftware. In plaats van op maat gemaakte apps te bouwen om zijn aanbod met andere bedrijven te integreren, lan-

ceerde Slack zijn ontwikkelaarsplatform en een reeks producten voor externe ontwikkelaars om hun eigen apps te bouwen, zowel bij gevestigde bedrijven als bij nieuwe startups.

Door deze beslissing groeide het ecosysteem voor apps die het berichtenplatform van Slack integreren. Het betekende ook dat gebruikers van Slack die ook andere bedrijfssoftware gebruikten, naadloos konden samenwerken omdat de communicatie al was geregeld in de messaging client van Slack.

Het voordeel voor de API's van Slack ten tijde van de lancering van het ontwikkelaarsplatform was dat de API's al waren getest en werden gebruikt door interne ontwikkelaars. De nadelen van deze aanpak kwamen in de loop van de tijd aan het licht, toen de behoeften van externe ontwikkelaars begonnen af te wijken van de behoeften van interne ontwikkelaars. Interne ontwikkelaars hadden behoefte aan flexibiliteit om nieuwe ervaringen te creëren voor eindgebruikers van de messaging client, van nieuwe soorten gedeelde kanalen, bestanden en berichten tot andere steeds complexere communicatie-ervaringen. Ondertussen ontwikkelden externe ontwikkelaars niet langer vervangende gebruikersinterfaces (UI's) voor Slack – ze begonnen krachtige bedrijfsapplicaties en hulpmiddelen te bouwen die meer waren ontworpen voor workflows dan voor berichtenweergave. Externe ontwikkelaars hadden ook stabiliteit nodig en de spanning tussen API-compatibiliteit met eerdere versies en de noodzaak om de API te wijzigen voor nieuwe productfuncties vertraagde de projectsnelheid binnen Slack.

API's voor externe ontwikkelaars eerst, interne ontwikkelaars als tweede

Sommige bedrijven maken eerst API's voor externe belanghebbenden en geven deze daarna vrij aan interne belanghebbenden. Zo heeft GitHub sinds het begin gewerkt. Laten we eens bekijken hoe en waarom GitHub zijn API heeft ontwikkeld en hoe zijn ontwikkelaarspubliek de evolutie van de API heeft beïnvloed.

In het begin bestond de API-doelgroep van GitHub voornamelijk uit externe ontwikkelaars die programmatische toegang tot hun eigen gegevens wilden verkrijgen. Kort na de eerste release van hun API begonnen zich kleine bedrijven te vormen rond de API van GitHub. Deze bedrijven ontwikkelden tools voor ontwikkelaars en verkochten deze aan gebruikers van GitHub.

Sindsdien heeft GitHub zijn API-aanbod aanzienlijk uitgebreid. Het heeft een API gebouwd die zowel individuen bedient die hun eigen persoonlijke projecten of workflows willen maken, als teams die willen samenwerken om botscripts of workflowtools te maken die integreren met GitHub. Deze teams, *integrators* genoemd, bouwen ontwikkeltools, verbinden gebruikers met het GitHub-platform en verkopen deze tools aan gemeenschappelijke klanten.

Toen GitHub uiteindelijk zijn GraphQL-API bouwde, waren ontwikkelaars van derde partijen de eerste gebruikers. GraphQL is een zoekinterface voor web-API's. Hoewel het niet de eerste van dergelijke interfaces is, was er voordat dit boek verscheen nogal wat over te doen vanwege de implementatie door Facebook, een bekende API-provider, en de adoptie ervan door GitHub, een andere bekende API-provider. Toen externe ontwikkelaars GitHubs nieuwe GraphQL-API begonnen te gebruiken, kozen interne GitHub-ontwikkelaars het ook als motor van de GitHub-webinterface en clienttoepassingen.

In het geval van GitHub had de API duidelijk tot doel om eerst externe belanghebbenden te bedienen; uiteindelijk breidde dit zich uit tot interne ontwikkelaars. Een voordeel van deze aanpak is dat de API kan worden aangepast om externe ontwikkelaars te bedienen, in plaats van in een spagaat te belanden tussen twee doelgroepen. Naarmate de API van GitHub zich ontwikkelde, kon deze zijn JSON-reacties uitbreiden met meer en meer gegevens die ontwikkelaars nodig hadden. Uiteindelijk waren de payloads

zo groot dat GitHub GraphQL implementeerde, zodat ontwikkelaars konden specificeren op welke velden ze wilden zoeken. Een nadeel van deze aanpak is in het geval van GraphQL dat vanwege de flexibiliteit die GraphQL ontwikkelaars biedt, knelpunten in de prestaties zich in allerlei toegangspatronen kunnen voordoen. Dit maakt het oplossen van problemen lastiger dan wanneer er met slechts één eindput tegelijk gewerkt wordt, bijvoorbeeld in het geval van REST. (Zie hoofdstuk 2 voor meer informatie over GraphQL.)

API's als product

Voor sommige bedrijven is de API het product. Dat is het geval met Stripe en Twilio. Stripe biedt API's voor betalingsverwerking op het internet. Twilio biedt API's voor communicatie via sms, spraak en berichten. Bij deze twee bedrijven is het bouwen van een API volledig afgestemd op een publiek voor één product. De API is het product en het hele bedrijf richt zich op het bouwen van een naadloze interface voor klanten. Wat betreft het beheer van API's en het voldoen aan gebruikersbehoeften is de API als product het eenvoudigste bedrijfsmodel.

Wat maakt een API goed?

We stelden experts uit de branche deze vraag. De antwoorden kwamen erop neer dat de API moet doen wat hij zou moeten doen. In ons onderzoek naar de aspecten die een gunstige invloed hebben op de bruikbaarheid van een API gaan we niet alleen in op het ontwerpen en schalen van API's, maar ook op de ondersteuning en ecosystemen die ontwikkelaars in staat stellen API's te gebruiken.

Advies van een expert

Een goede API komt neer op het oplossen van uw probleem en hoe waardevol het oplossen ervan is. U bent misschien bereid om een verwarrende, inconsistente, slecht gedocumenteerde API te gebruiken als dit betekent dat u toegang krijgt tot een unieke gegevensset of complexe functionaliteit. Goede API's bieden daarentegen doorgaans duidelijkheid (van doel, ontwerp en context), flexibiliteit (mogelijkheid om te worden aangepast aan verschillende toepassingen), kracht (volledigheid van de aangeboden oplossing), aanpasbaarheid (mogelijkheid om een en ander snel op te pakken door iteratie en experimenteren) en documentatie.
— Chris Messina, Developer Experience Lead bij Uber

Bruikbaarheid, *schaalbaarheid* en *prestaties* zijn enkele aspecten van een goede API. We behandelen veel van deze onderwerpen in de hoofdstukken 2 tot en met 4 van dit boek. *Documentatie* en *ontwikkelaarsbronnen* zijn ook belangrijk voor succesvolle implementatie. We behandelen die in de hoofdstukken 7 tot en met 9. Omdat een API onmogelijk voor alle omstandigheden te optimaliseren is, moet het implementatieteam moeilijke beslissingen nemen over wat het belangrijkst is voor de eindgebruiker. We leren u in hoofdstuk 7 hoe u hier een strategie voor ontwikkelt.

Nog een aandachtspunt is hoe een goede API de tand des tijds doorstaat. Verandering is moeilijk en onvermijdelijk. API's zijn flexibele platforms die bedrijven met elkaar verbinden, waarbij de verandersnelheid variabel is. In grote bedrijven is de verandersnelheid langzamer dan in kleine startups waarvan de producten nog niet naadloos aansluiten op de marktvraag. Maar soms leveren deze kleine startups onschatbare diensten via

API's die bedrijven moeten gebruiken. In hoofdstuk 5 leert u ook hoe u API's kunt ontwerpen om de tand des tijds en veranderingen te doorstaan.

Ter afsluiting

Samengevat zijn API's een belangrijke component van moderne technologische producten en er zijn veel manieren om de bedrijvigheid ermee vorm te geven. In hoofdstuk 2 geven we u een overzicht van API-ontwerpmodellen.