Engineer

API

⇵ det beskrivna sättet för ett program att anropa ett annat

Vad det är

API, förkortning för application programming interface, är den publicerade uppsättning operationer som ett program erbjuder andra program: vad varje operation heter, vad den tar emot, vad den returnerar och hur den rapporterar problem. I bestämd form heter det API:et och i plural API:er. Den som anropar lär sig den ytan och ingenting bakom den, så programmet på andra sidan kan skrivas om, flyttas till en annan maskin eller delas upp i tre, och anroparen fortsätter att fungera så länge ytan håller.

Ordet täcker flera skikt. Ett biblioteks API är de funktioner och typer som det exporterar, och de anropas inuti en och samma process. Ett operativsystems API är dess systemanrop, och POSIX standardiserar en gemensam kärna av dem, så att ett program som skrivs mot `open` och `read` går att bygga på mer än en Unix-liknande systemkärna. Ett webb-API erbjuds över ett nätverk, nästan alltid över HTTP, och det är vad de flesta menar med ordet i dag: en server som besvarar förfrågningar på adresser som `/v1/orders`, oftast med JSON i kroppen.

Webb-API:er finns i en handfull stilar. De resursorienterade, ofta kallade REST-liknande, ger varje sak en adress och använder HTTP:s metoder för att läsa och ändra den. De RPC-liknande, som gRPC, publicerar namngivna procedurer vars argument och resultat deklareras i förväg. GraphQL erbjuder en enda endpoint och ett schema, och klienten skriver en fråga som namnger exakt de fält den vill ha tillbaka. Webhooks vänder på riktningen: när något händer anropar leverantören en adress som kunden har registrerat.

Tre närliggande ord blandas ofta ihop med API. Ett nätverksprotokoll, som HTTP, bestämmer hur meddelanden färdas, medan ett API är det som en viss tjänst säger med hjälp av protokollet. Ett SDK är ett bibliotek i ett visst programspråk

som kapslar in någons API, så att en utvecklare anropar en funktion i stället för att sätta ihop förfrågningar själv. Och implementationen är koden bakom API:et, som den som anropar aldrig behöver läsa.

Använd begreppet i ett beslut

Använd ”Vad är API?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Den del användarna aldrig ser: där data lagras, vilka reglerna är och hur andra system får kommunicera med den.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Utformade och drev 6 ETL/ELT-pipelines — 6 ETL-pipelines över tre databasmotorer

Utformade och drev 6 ETL/ELT-pipelines Hör av er →

Varför det spelar roll

Ett API är den del av ett system som andra bygger vidare på, och det gör det till ett löfte som lever längre än koden bakom det. Att skriva om en handler tar en eftermiddag. Att ändra vad samma handler returnerar kan få en app som redan är installerad i kundernas telefoner att sluta fungera, liksom en partners nattliga import eller ett skript som ingen längre minns att någon skrev.

Det första beslutet handlar därför om publiken. Ett privat API, som bara teamets egen frontend använder, kan ändras i samma pull request som sin enda klient. Ett publikt API ändras genom utfasning och förvarning, och varje fält som det levererar är ett fält att stödja så länge någon läser det.

Det andra beslutet gäller hur strikt kontraktet är. Ett strikt API deklarerar en typ och en gräns för varje indatavärde, avvisar det som det inte känner igen och besvarar varje misslyckande med en exakt statuskod. Det fångar misstag vid dörren och är lättare att dokumentera, testa och generera klienter för. Ett tillåtande API tar emot vad som än kommer och gör sitt bästa, vilket känns vänligt den första månaden och sedan gör varje tillfällighet i implementationen till ett beteende som någon är beroende av. Stränghet kostar designtid i början. Det tillåtande kostar mer senare, vid en tidpunkt som ingen själv väljer.

Det tredje beslutet gäller andras API:er. En betaltjänst, en karttjänst eller en språkmodell är ett API som produkten anropar men inte styr över: priser ändras, versioner pensioneras och kvoter stramas åt. Den som anropar en leverantör

direkt från femtio ställen i koden gör ett byte av leverantör till en omskrivning. Den som anropar den från en enda modul som teamet äger gör bytet till en ändring i den modulen. Omslaget kostar lite indirektion och köper en väg ut.

Semantic Versioning ger löftet ett ordförråd: huvudversionen räknas upp när det publika API:et ändras på ett sätt som inte är bakåtkompatibelt, så att en klient kan avgöra enbart utifrån versionsnumret om en uppgradering är säker. Över HTTP finns flera ställen där ett sådant brott kan signaleras, och RFC 9205, IETF:s vägledning för protokoll som byggs ovanpå HTTP, räknar bl.a. upp en ny länkrelation, en ny medietyp och ett nytt header-fält.

Så ser det ut i praktiken

API:erna bakom flera projekt på den här webbplatsen formades av en och samma fråga: vad hör hemma i API:et, och vad hör hemma någon annanstans?

På NextMariner, en maritim plattform för professionellt nätverkande, företagsrecensioner och sponsrad utbildning, ingick det i grundarens arkitekturbidrag att hålla Go-API:et medvetet tunt. Det är byggt med Huma ovanpå Fiber, och det routar, validerar förfrågan, tillämpar säkerhetsfiltreringen och orkestrerar, men innehåller ingen affärslogik alls. Reglerna ligger i PostgreSQL-funktioner, så när en regel ändras sker det i ett lager, i SQL. Gränserna skrevs ned och upprätthölls i granskningen. Det gör API:et till en tråkig adapter, i ordets goda bemärkelse.

Ett närliggande projekt flyttade även formningen av svaren ut ur API:et. PostgreSQL-funktioner sätter ihop varje svar som JSON, och Go-handlern tar emot det som `json.RawMessage` och vidarebefordrar det orört, genom en liten hjälpfunktion som heter `QueryJSON`. DTO:erna, mappningsfunktionerna och svars-structarna som ett lagerindelat API brukar samla på sig föll bort, handlerkoden blev dramatiskt kortare och frontenden slutade vara kopplad till Go. Avvägningen är värd att nämna: formen på API:ets svar definieras nu i SQL, och det är där en granskare läser kontraktet.

Ett API som vem som helst kan nå måste också kunna försvara sig. En publik katalog över företag och yrkespersoner skyddas av 22 anropsbegränsare, var och en byggd för den väg den skyddar, med sitt tillstånd i Redis så att en gräns delas mellan instanser. När Redis inte går att nå stänger begränsarna vid fel och nekar trafiken. Cloudflare Turnstile, ett ursprungslås, en specialbyggd utmaning på själva katalogen och 351 rader middleware för idempotens, som hindrar att en

skrivning som görs om landar två gånger, fullbordar försvaret. Sidan anger kostnaden rakt ut: ett Redis-problem blir ett problem som användarna ser, och det valdes hellre än att ett Redis-problem blir ett faktureringsproblem.

Var det går fel

Ett API går oftast sönder sent, när klienterna och datan väl har vuxit.

Ett omdöpt fält slår ut en klient som ingen kände till. När ett API väl är publicerat är någon beroende av varje detalj som går att observera: fältnamn, ordningen i en lista, formuleringen i ett felmeddelande. Döp om `customer_name` till `name`, så slutar en partners integration att fungera vid nästa förfrågan. Ändra genom tillägg i stället: lägg det nya fältet bredvid det gamla, markera det gamla för utfasning, följ loggarna tills ingen läser det och ta bort det först i en ny huvudversion.

En list-endpoint blir långsammare för varje månad. Den returnerar varje matchande rad, vilket gick bra med femtio poster vid lanseringen och ger timeout vid tvåhundratusen. Paginera från första versionen, med en standardstorlek på sidan och ett hårt maximum som klienten inte kan gå över. För stora eller livligt använda samlingar är en markör (cursor) bättre än en förskjutning (offset): med en förskjutning puttar en rad som infogas mellan två förfrågningar en annan rad till nästa sida, där klienten läser den två gånger.

```
GET /v1/orders?limit=50&after=ord_8f2c HTTP/1.1
Host: api.example.com
```

En kund läser en annan kunds order. API:et kontrollerar att anroparen är inloggad och hämtar sedan `/v1/orders/42` enbart utifrån identifieraren, så den som ändrar numret i adressen kan bläddra genom andra människors data. Autentisering besvarade frågan om vem som anropar, men ingenting frågade om just den anroparen äger order 42. Avgränsa varje uppslag efter ägare i själva databasfrågan, och behåll ett test som loggar in som ett konto och begär ett annat kontos poster.

Svaret bär kolumner som ingen tänkte publicera. En endpoint serialiserar en rad ur databasen eller ett ORM-objekt precis som det är, så att hashen av ett lösenord, en intern flagga eller en leverantörs inköpspris går ut till varje klient, och en kolumn som läggs till nästa år publiceras samma dag som den skapas.

Samma misstag på indatasidan låter en förfrågan sätta `role` eller `is_admin` bara genom att ta med fältet i kroppen. Ge varje endpoint uttryckliga typer för in- och utdata som räknar upp de tillåtna fälten, så att publiceringen av ett fält blir ett beslut som någon fattar.

Läs mer

- RFC 9110: HTTP Semantics — metoderna, statuskoderna och header-fälten som varje webb-API byggs av, så som IETF definierar dem.
- RFC 9205: Building Protocols with HTTP — aktuell bästa praxis för att utforma en applikation ovanpå HTTP, bl.a. hur den kan utvecklas över tid.
- RFC 9457: Problem Details for HTTP APIs — en standardiserad JSON-form för felsvar, så att klienter tolkar varje misslyckande på samma sätt.
- RFC 6585: Additional HTTP Status Codes — där 429 Too Many Requests definieras, tillsammans med `Retry-After` som talar om för en begränsad klient när den kan försöka igen.
- Semantic Versioning 2.0.0 — reglerna för att signalera en API-ändring som inte är bakåtkompatibel genom ett versionsnummer.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är API?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/api/