Engineer

API

🎼 описаний спосіб, яким одна програма звертається до іншої

Що це таке

API (від англ. application programming interface, прикладний програмний інтерфейс) — це опублікований набір операцій, які програмне забезпечення відкриває для іншого програмного забезпечення: як називається кожна операція, що вона приймає, що повертає і як повідомляє про проблеми. Той, хто викликає, знає лише цю поверхню й нічого з того, що за нею.
Тому програму з іншого боку можна переписати, перенести на іншу машину або розділити на три частини, а клієнт працюватиме далі, доки поверхня тримається.

Слово охоплює кілька рівнів. API бібліотеки — це функції та типи, які вона експортує і які викликають у межах одного процесу. API операційної системи — це її системні виклики, і POSIX стандартизує спільний їх набір, тож програму, написану під `open` і `read`, можна зібрати для кількох Unix-подібних систем. Веб-API доступний через мережу, майже завжди через HTTP, і саме його сьогодні найчастіше мають на увазі: сервер, що відповідає на запити за адресами на кшталт `/v1/orders`, зазвичай із JSON у тілі.

Веб-API бувають кількох стилів. Ресурсно орієнтовані, які часто називають REST-подібними, дають кожній сутності власну адресу й використовують методи HTTP, щоб її читати та змінювати. RPC-подібні, як-от gRPC, відкривають іменовані процедури з аргументами й результатами, оголошеними заздалегідь. GraphQL має одну точку входу та одну систему типів, а клієнт пише запит, у якому перелічує саме ті поля, які хоче отримати. Вебхуки обертають напрямок: постачальник сам викликає адресу, яку зареєстрував клієнт, коли щось стається.

Три сусідні поняття з ним часто плутають. Протокол, як-от HTTP, визначає, як передаються повідомлення; API — це те, що конкретний сервіс каже цим протоколом. SDK — це бібліотека однією мовою програмування, яка обгортає чужий API, щоб розробник викликав функцію, а не складав запити вручну. А реалізація — це код за API, який тому, хто викликає, читати зовсім не потрібно.

Застосуйте поняття до рішення

Застосуйте «Що таке API?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Та частина, якої користувач не бачить: де зберігаються дані, які діють правила і як інші системи можуть із нею взаємодіяти.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Спроєктувала й підтримувала 6 пайплайнів ETL/ELT — 6 пайплайнів ETL на трьох рушіях баз даних

Спроєктувала й підтримувала 6 пайплайнів ETL/ELT Зв'язатися з нами →

Чому це важливо

API — це частина системи, на якій будують інші, тому він є обіцянкою, що живе довше за код, який за ним стоїть. Переписати обробник можна за пообіддя. Змінити те, що він повертає, означає ризикнути застосунком, уже встановленим на телефонах клієнтів, нічним імпортом партнера або скриптом, автора якого вже ніхто не пам’ятає.

Отже, перше рішення стосується аудиторії. Приватний API, яким користується лише фронтенд власної команди, можна змінити в тому самому pull request, що й його єдиного клієнта. Публічний API змінюють через оголошення застарілим і попередження, і кожне поле, яке він віддає, доведеться підтримувати, доки хтось його читає.

Друге рішення — наскільки суворим буде контракт. Суворий API оголошує тип і межу для кожного вхідного значення, відхиляє те, чого не знає, і на кожну помилку відповідає точним кодом статусу. Він ловить помилки на вході, і його легше документувати, тестувати й генерувати для нього клієнтів. Поблажливий API приймає все, що надійшло, і робить, що може. Перший місяць це здається дружнім, а потім кожна випадкова риса реалізації стає поведінкою, на яку хтось покладається. Суворість коштує часу на проєктування одразу; поблажливість коштує більше згодом, у момент, якого ніхто не обирає.

Третє рішення стосується чужих API. Платіжний провайдер, картографічний сервіс чи мовна модель — це API, який продукт викликає, але не контролює: ціни змінюються, версії виводять з обігу, квоти зменшуються. Якщо викликати постачальника напряму з п'ятдесяти місць, перехід до іншого стане переписуванням. Якщо викликати його з одного модуля, яким володіє команда, перехід зведеться до зміни цього модуля. Обгортка коштує трохи опосередкованості й дає вихід.

Semantic Versioning дає цій обіцянці словник: старший номер версії зростає, коли публічний API змінюється несумісно, тож клієнт за самою версією бачить, чи безпечне оновлення. У HTTP є кілька способів позначити несумісну зміну, і RFC 9205, настанова IETF для протоколів на основі HTTP, серед них називає нове відношення посилання, новий тип медіа й нове поле заголовка.

Як це виглядає на практиці

Кілька проєктів на цьому сайті будували свій API навколо одного питання: що належить до API, а що — до іншого місця.

На NextMariner, морській платформі для професійних контактів, відгуків про компанії та спонсорованого навчання, засновниця компанії в межах свого архітектурного внеску, зокрема, навмисно залишила Go API тонким. Побудований на Huma поверх Fiber, він маршрутизує, перевіряє запит, застосовує фільтрацію з міркувань безпеки й оркеструє, але не містить жодної бізнес-логіки. Правила живуть у функціях PostgreSQL, тож зміна правила відбувається в одному шарі, у SQL. Межі записали й забезпечили їх дотримання під час рев'ю. Це робить API нудним адаптером у доброму сенсі.

Споріднений проєкт виніс з API ще й формування відповідей. Функції PostgreSQL збирають кожну відповідь як JSON, а обробник на Go отримує її як `json.RawMessage` і пересилає без змін через невеликий помічник `QueryJSON`. DTO, мапери й структури відповідей, які зазвичай накопичує багатошаровий API, відпали, код обробників став значно коротшим, а фронтенд перестав залежати від Go. Варто назвати й ціну: форма відповідей API тепер визначена в SQL, тож саме там рецензент читає контракт.

API, до якого може дістатися будь-хто, мусить і захищатися. Публічний каталог компаній і фахівців стоїть за 22 обмежувачами частоти запитів, кожен для свого шляху, а їхній стан зберігається в Redis, тож ліміт діє для всіх

екземплярів. Коли Redis зникає, обмежувачі закриваються й відхиляють трафік. Навколо них — Cloudflare Turnstile, 351 рядок проміжного ПЗ для ідемпотентності, щоб повторений запис не перетворився на два, блокування джерела та окрема перевірка-виклик на самому каталозі. Ціну сторінка називає прямо: проблема з Redis стає проблемою, яку бачать користувачі, і цей варіант обрали замість проблеми з Redis, що закінчується рахунком.

Де виникають проблеми

Збої API зазвичай з'являються пізно, коли клієнтів і даних уже стало більше.

Перейменоване поле ламає клієнта, про якого ніхто не знав. Щойно API опубліковано, хтось покладається на кожну видиму деталь: назви полів, порядок у списку, формулювання повідомлення про помилку. Перейменуйте `customer_name` на `name` — і партнерська інтеграція зламається на першому ж запиті. Змінюйте натомість додаванням: покладіть нове поле поруч зі старим, позначте старе застарілим, стежте за журналами, доки його перестануть читати, і приберіть його лише в новій старшій версії.

Ендпоінт зі списком щомісяця сповільнюється. Він повертає всі рядки, що підпадають під запит; із п'ятдесятьма записами на старті це працювало, а на двохстах тисячах закінчується тайм-аутом. Додайте пагінацію з першого релізу: типовий розмір сторінки й жорстку межу, яку клієнт не може перевищити. Для великих або активних колекцій курсор кращий за зсув (offset): зі зсувом рядок, вставлений між двома запитами, виштовхує інший рядок на наступну сторінку, і клієнт читає його двічі.

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

Один покупець бачить замовлення іншого. API перевіряє, що той, хто викликає, увійшов у систему, а потім завантажує `/v1/orders/42` лише за ідентифікатором, тож, змінюючи число в адресі, можна гортати чужі дані. Автентифікація відповіла на питання, хто викликає; ніхто не спитав, чи належить цьому користувачеві замовлення 42. Обмежуйте кожен пошук власником у самому запиті до бази й тримайте тест, який входить як один обліковий запис і запитує записи іншого.

Відповідь містить стовпці, які ніхто не збирався публікувати. Ендпоінт серіалізує рядок бази даних або об'єкт ORM як є, тож хеш пароля, внутрішній прапорець чи закупівельна ціна постачальника йдуть до кожного клієнта, а стовпець, доданий наступного року, стає публічним у день, коли його створили. Та сама помилка на вході дає запиту змогу встановити `role` або `is_admin`, просто додавши поле в тіло. Дайте кожному ендпоінту явні типи входу й виходу з переліком дозволених полів, щоб публікація поля стала свідомим рішенням.

Що почитати далі

- RFC 9110: HTTP Semantics — методи, коди статусу й заголовки, з яких складається кожен веб-API, у визначенні IETF.
- RFC 9205: Building Protocols with HTTP — найкраща поточна практика проєктування застосунку поверх HTTP, зокрема того, як його розвивати.
- RFC 9457: Problem Details for HTTP APIs — стандартна JSON-форма для відповідей із помилками, щоб клієнти розбирали кожен збій однаково.
- RFC 6585: Additional HTTP Status Codes — де визначено 429 Too Many Requests і заголовок `Retry-After` для клієнтів, яких обмежують.
- Semantic Versioning 2.0.0 — правила, за якими несумісну зміну API позначають номером версії.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке API?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/api/