Engineer

OpenAPI

машиночитний опис HTTP-API

Що це таке

OpenAPI — це стандартний формат, у якому в одному файлі JSON або YAML записують усе, що треба знати тому, хто викликає HTTP API: адреси, операції, вхідні й вихідні дані, помилки та безпеку, — і цей файл однаково читають люди й програми. Сам стандарт має назву OpenAPI Specification, скорочено OAS, а файл, написаний за ним, називають описом OpenAPI. Формат починався як Swagger Specification, і стара назва досі живе в інструментах, побудованих навколо нього, як-от Swagger UI.

Специфікація називає себе незалежним від мови програмування описом інтерфейсу для HTTP API. Мета, яку вона ставить, — щоб людина чи програма могла зрозуміти, що робить сервіс, не маючи ні його вихідного коду, ні додаткової документації, ні запису мережевого трафіку. Головний документ рекомендовано називати `openapi.json` або `openapi.yaml`, а великі описи часто ділять на кілька файлів, пов'язаних посиланнями.

Історія формату проходить через одну передачу. Swagger 2.0 вийшов у вересні 2014 року, а 31 грудня 2015 року його передали OpenAPI Initiative. Версія 3.0.0 з'явилася в липні 2017 року вже під новою назвою, 3.1.0 — у лютому 2021-го, а 3.2.0 — у вересні 2025-го.

Опис має кілька частин верхнього рівня. `info` називає API та його версію; `servers` перелічує, де він працює; `paths` містить кожну адресу, а під нею — по одній операції на кожен метод HTTP з її параметрами, тілом запиту й можливими відповідями. `components` збирає те, що використовують повторно: схеми даних, параметри, відповіді та схеми безпеки, які описують, як автентифікується той, хто викликає. Від версії 3.1 розділ `webhooks` може описувати ще й запити, які API сам надсилає своїм споживачам. Мінімальний фрагмент:

```
openapi: 3.1.0
info:
    title: Orders
    version: 1.4.0
paths:
    /orders/{id}:
        get:
            operationId: getOrder
            parameters:
                - name: id
                  in: path
                  required: true
                  schema:
                      type: string
            responses:
                "200":
                   description: One order
```

Структури даних задають окремою мовою. Від 3.1 Schema Object є надмножиною JSON Schema Draft 2020-12; 3.0 використовувала розширену підмножину старішої чернетки з власними доповненнями. JSON Schema описує один фрагмент даних, а OpenAPI — цілий HTTP API і для окремих фрагментів позичає JSON Schema. Сусіди описують інші види інтерфейсів: AsyncAPI охоплює системи, побудовані на обміні повідомленнями, сервіси gRPC оголошують у файлах `.proto` формату Protocol Buffers, а сервер GraphQL публікує опис власних типів.

Застосуйте поняття до рішення

Застосуйте «Що таке OpenAPI?» 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

Опис у стандартному форматі — це один файл, який читає багато інструментів, і кожен із них заощаджує роботу, яку команда інакше робила б уручну. З нього рендерять довідкову документацію. З нього генерують клієнтські бібліотеки кількома мовами, тож фронтенд викликає типізовану функцію, а не складає URL сам. Мок-сервери відповідають на запити на його

основі, тож фронтенд можна будувати ще до появи бекенду. Шлюзи імпортують із нього маршрути, тести звіряють із ним трафік, а ШІ-агенти, які викликають API, читають його, щоб дізнатися, що їм пропонують.

Це охоплення водночас і ризик. Кожен із цих інструментів довіряє файлу. Хибний опис гірший за відсутній, бо згенерований клієнт компілюється, документація має авторитетний вигляд, а помилка виявляється вже під час роботи, у чужому продукті.

Тож головне рішення — звідки береться опис. У підході design-first люди пишуть опис, рецензують його як контракт і лише тоді будують сервер відповідно до нього. Форму API обговорюють ще до появи коду, що пасує публічним API і командам, які працюють паралельно. Ціна в тому, що опис і сервер — дві окремі речі, які можуть розійтися. У підході code-first опис генерують із власних типів і анотацій сервера. Він не може описати маршрут, якого сервер не має, але точний лише настільки, наскільки точні ці типи, а обговорення дизайну відбувається в code review, а не в документі. Хоч би який шлях обрали, опис лишається правдивим, тільки якщо його перевіряє збірка.

Версії важать більше, ніж здається. Перехід із 3.0 на 3.1 змінив спосіб опису структур даних, бо 3.1 перейняла JSON Schema 2020-12. Генератори, валідатори й шлюзи підтримують кожен свій діапазон версій, тож версію, якою пише команда, обмежує найстаріший інструмент у ланцюжку.

Як це виглядає на практиці

API однієї платформи починався на Fiber із валідацією, написаною вручну для кожного ендпоінта, і без єдиного документа, що описував би його форму. HTTP-рівень перейшов на Huma v2 поверх Fiber, тож кожен ендпоінт оголошують через вхідні й вихідні структури, а опис OpenAPI генерується з цих типів; документація встигає за кодом, а не старіє у вікі. Два ендпоінти WebSocket лишилися на чистому Fiber, поза моделлю запиту й відповіді, яку пропонує Huma. Сторінка прямо пояснює, чому важить масштаб: коли операцій 760, а шляхів 649, опис стає по суті єдиним способом прочитати API, тож упевненість у його повноті важить більше, ніж тоді, коли їх було п’ятдесят. Перевірка в CI тримає відповідність між маршрутами й описом на рівні 100%.

На іншій платформі опис — середня ланка ланцюга, що тягнеться від бази даних назовні. Схема та її функції задають типи Go, Huma видає опис, а з нього генерують усе, чим користується фронтенд: типізований клієнт TypeScript на 44 076 рядків, 61 мок-обробник для тестів фронтенду та обмеження, які застосовує кожна форма. Так долають розходження між фронтендом і бекендом, яке зазвичай дається взнаки як 422 у браузері, коли розбіжність уже потрапила в реліз. Перевірка контракту в CI зупиняє збірку в разі розбіжності й окремо прискіпливо звіряє необов'язкові поля тіла запиту, де плутають «відсутнє» і null, та enum-и параметрів запиту. Ціну сторінка теж називає: крок генерації посеред усього й ланцюг, якому можна довіряти лише настільки, наскільки надійна його найслабше захищена ланка.

Сторінка компанії про готовність до ШІ-агентів застосовує ту саму логіку до агентів. Застосунок, яким можуть користуватися агенти, пропонує описаний API: документ OpenAPI, MCP-сервер там, де агентам слід викликати застосунок напряму, і метадані OAuth, щоб агент міг діяти від імені користувача і ніхто не передавав свого пароля. Кожну рекомендацію для агентів сторінка класифікує, а не виконує наосліп, і оскільки власний сайт компанії не має API, каталог API та метадані OAuth — дві з п'яти, від яких там відмовилися.

Де виникають проблеми

Опис обіцяє те, чого сервер не робить. Опис, який підтримують уручну, відстає: новий параметр запиту, перейменоване поле чи код помилки, про який знає лише сервер. Згенеровані клієнти тоді компілюються на основі вигаданого опису. Генеруйте опис із коду або тримайте його окремо й перевіряйте за ним справжні запити та відповіді в тестах, а також порівнюйте маршрути, які реєструє сервер, зі шляхами, які перелічує опис.

Кожна перегенерація клієнта перейменовує половину методів. Без `operationId` генератори вигадують назви методів зі шляху та методу HTTP, тож перенесення маршруту перейменовує функцію, яку викликає кожен клієнт. Специфікація вимагає, щоб кожен `operationId` був унікальним, і розрізняє в ньому великі й малі літери. Задайте його кожній операції, добирайте так само ретельно, як назву публічної функції, і нехай лінтер зупиняє збірку, коли його бракує.

Поля, що можуть бути null, після оновлення починають відхиляти null. OpenAPI 3.0 позначала їх як `nullable: true`. У 3.1 цього ключового слова вже немає, тип стає списком, а валідатор, що натрапляє на незнайоме ключове слово, зазвичай його пропускає. Схема 3.0, якій просто змінили номер на 3.1, тому відхиляє null усюди, де автор хотів його дозволити. Перетворюйте схеми за посібником OpenAPI Initiative з оновлення, де описано й те, що `exclusiveMinimum` стає числом:

```
# OpenAPI 3.0
type: string
nullable: true
---
# OpenAPI 3.1
type: [string, "null"]
```

Приклади не проходять власну схему. Специфікація каже, що приклад слід узгоджувати з його схемою, а перевірку залишає інструментам. Читач копіює приклад із документації й отримує 422; мок-сервер віддає його, і фронтенд будують на даних, які справжній API відхилить. Перевіряйте кожен приклад у CI так само суворо, як живий трафік.

Публічний опис відкриває карту адміністративного API. Один опис, згенерований з усіх маршрутів і доступний за публічною адресою, показує внутрішні операції та їхні схеми кожному, хто запитає. Публікуйте окремий опис для кожної аудиторії, згенерований без внутрішніх операцій. Специфікація дозволяє залишати об’єкти порожніми або прибирати їх саме для такого контролю доступу.

Що почитати далі

- OpenAPI Specification — остання версія стандарту з визначенням кожного об’єкта й поля та історією кожного випуску.
- JSON Schema Draft 2020-12 — мова схем, якою OpenAPI 3.1 і новіші версії описують тіло кожного запиту й відповіді.
- Upgrading from OpenAPI 3.0 to 3.1 — власний перелік змін у схемах від OpenAPI Initiative, серед них і `nullable`.
- Huma: OpenAPI Generation — як Go-фреймворк, ужитий у проєктах вище, виводить опис із типів обробників.

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 OpenAPI?» 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/openapi/