Engineer

API

🗘 あるプログラムが別のプログラムを呼び出すための、仕様を定めた方法

どのようなものか

APIはapplication programming interfaceの略で、ソフトウェアが他のソフトウェアに公開する操作の集合です。各操作の名前、受け取るもの、返すもの、問題の伝え方を定めます。英語の複数形はAPIsです。呼び出し元が知るのは、その表面だけで、背後は知りません。そのため、背後のプログラムを書き直し、別のマシンへ移し、三つに分けても、表面を保てば、利用側は動き続けます。

この語には複数の層があります。ライブラリーのAPIは、そのライブラリーが公開する関数と型であり、一つのプロセスの中から呼び出します。OSのAPIはシステムコールの集合で、POSIXは、その共通部分を標準化します。`open` や `read` を使うプログラムを、複数のUnix系カーネルでビルドできるためです。ウェブAPIは、ほぼ常にHTTPを使い、ネットワーク越しに提供します。現在、単にAPIと言うと、多くの場合これを意味します。サーバーが `/v1/orders` などのアドレスで要求に応答し、通常は本文にJSONを返します。

ウェブAPIには、いくつかの形式があります。REST形式とも呼ぶリソース指向のものは、各対象にアドレスを与え、HTTPメソッドで読み書きします。gRPCなどのRPC形式は、引数と結果を事前に宣言した、名前付きの手続きを公開します。GraphQLは、一つのエンドポイントと型体系を提供し、クライアントが、必要なフィールドを正確に指定するクエリを書きます。Webhookは向きを逆にし、何かが起きたとき、提供側がお客様の登録したアドレスを呼び出します。

近い三つの言葉が、混同されます。HTTPなどのプロトコルは、メッセージの伝わり方を定め、APIは、あるサービスがそのプロトコルで何を伝えるかを定めます。SDKは、特定の言語でAPIを包むライブラリーで、開発者は要求を組み立てる代わりに関数を呼べます。実装はAPIの背後のコードで、呼び出し元は読む必要がありません。

概念を判断に生かす

「APIとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 データの保存先、処理のルール、他のシステムとの接続方法など、利用者の目に見えない部分をつくります。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 ETL/ELTパイプライン6本を設計・運用 — 6 3種類のDBエンジンにまたがるETLパイプライン

ETL/ELTパイプライン6本を設計・運用 お問い合わせ →

なぜ重要なのか

APIは、他の人がその上に構築する部分であり、実装より長く残る約束です。ハンドラーは半日あれば書き直せても、返すものを変えると、顧客のスマートフォンにインストール済みのアプリ、提携先の夜間取り込み、誰も書いたことを覚えていないスクリプトが壊れます。

最初の判断は、対象者です。自分たちのフロントエンドだけが使う非公開APIなら、唯一の利用側と同じプルリクエストで変えられます。公開APIは、非推奨化と予告を通して変更し、公開する各フィールドを、読む人がいる限り支える必要があります。

二つ目は、契約の厳密さです。厳密なAPIは、各入力に型と制限を宣言し、知らないものを拒否し、各失敗に正確なステータスコードを返します。入り口で誤りを捕まえ、文書化、テスト、クライアント生成もしやすくなります。寛容なAPIは、届くものを受け入れ、できる限り処理します。最初の一か月は親切に見えても、その後、実装上の偶然すべてが、誰かの依存する挙動になります。厳密にするには最初に設計の時間がかかります。寛容にすれば、後になって、誰も望まない時点で、さらに大きな負担が生じます。

三つ目は、他社のAPIです。決済事業者、地図サービス、言語モデルは、製品が呼んでも制御できないAPIです。価格が変わり、バージョンが廃止され、割り当てが厳しくなります。五十か所から直接呼ぶと、切り替えは書き直しになります。自分たちが管理する一つのモジュールから呼べば、その変更で済みます。ラッパーを挟むと呼び出しは少し間接的になりますが、乗り換える道を確保できます。

Semantic Versioningは、この約束を表す語彙です。公開APIに非互換の変更があればメジャー番号を上げ、利用側が番号だけで更新の安全性を判断できるようにします。HTTP上では、破壊的変更の示し方が複数あります。HTTP上のプロトコ

ル設計に関するIETFの指針RFC 9205は、新しいリンク関係、メディアタイプ、ヘッダーフィールドなどを挙げています。

実際の活用例

このサイトの複数のプロジェクトでは、何をAPIに置き、何を他へ置くかという問いに沿って、APIを設計しました。

NextMarinerは、ビジネス上の人脈づくり、企業レビュー、スポンサーの支援による教育を扱う海事プラットフォームです。創業者によるアーキテクチャーへの貢献には、Go APIを意図して薄く保つことが含まれます。Fiber上のHumaで構築し、経路の振り分け、要求の検証、セキュリティ上の絞り込み、処理の調整を行い、業務ロジックは一切持ちません。ルールはPostgreSQL関数に置き、一つのSQL層で変更します。境界を文書化し、レビューで守りました。こうした構成により、APIはよい意味で平凡なアダプターになります。

関連するプロジェクトは、応答の成形もAPIから外しました。PostgreSQL関数が各応答をJSONとして組み立て、Goハンドラーは `json.RawMessage` として受け取り、小さな `QueryJSON` 補助関数で、そのまま転送します。階層化したAPIで増えがちなDTO、マッパー、応答構造体が不要になり、ハンドラーは大幅に短くなり、フロントエンドもGoに依存しなくなりました。代償も明示すべきです。API応答の形はSQLで定義するため、レビューで契約を読む場所もSQLになります。

誰でも到達できるAPIには、自衛も必要です。企業と専門職の公開名簿の前に、保護する経路ごとに構築した22のレートリミッターを置き、状態をRedisに保持して、複数インスタンス間でも制限を保ちます。Redisに到達できなければ、安全側に倒して拒否します。Cloudflare Turnstile、オリジンの制限、名簿向けの独自チャレンジ、再試行の書き込みを二重にしない351行の冪等性ミドルウェアで、防御を補っています。そのページは代償も明確にしています。Redisの問題を請求の問題にするより、利用者に見える問題にすることを選びました。

問題が起きるところ

APIの問題は、利用側とデータが増えた後に現れがちです。

フィールド名の変更で、知らなかった利用側が壊れます。 公開すると、名前、一覧の順序、エラーの言葉など、観測可能な細部すべてに誰かが依存します。`customer_name` を `name` へ変えると、提携先の連携は次の要求で壊れます。代わ

りに追加で変更します。古いものと並べて新しいフィールドを加え、古いものを非推奨とし、誰も読まなくなるまでログを確認し、新しいメジャーバージョンでだけ削除します。

一覧のエンドポイントが毎月遅くなります。 一致する全行を返し、公開時の五十件なら問題なくても、二十万件では時間切れになります。最初の公開からページ分割し、標準件数と、利用側が超えられない上限を設けます。大きい、または変化の多い一覧では、オフセットよりカーソルを優先します。オフセットでは、二つの要求の間に挿入された行で、別の行が次ページへ押され、二度読まれるためです。

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

ある顧客が、別の顧客の注文を読みます。 ログインだけを確認し、`/v1/orders/42` をIDだけで読み込むと、アドレスの番号を変えるだけで他人のデータを巡れます。認証は呼び出し元を答えましたが、注文42の所有者かは何も問いませんでした。すべての検索をクエリ自体で所有者に絞り、一方のアカウントでログインして他方の記録を要求するテストを残します。

公開するつもりのない列を応答に含めます。 データベース行やORMオブジェクトをそのままシリアライズし、パスワードハッシュ、内部フラグ、仕入れ原価がすべての利用側へ出ます。翌年に追加する列も、作成当日に公開されます。入力で同じ誤りをすると、本文に含めるだけで `role` や `is_admin` を設定できます。各エンドポイントに、許すフィールドを列挙した明示的な入力・出力型を与え、公開を人の判断にします。

参考資料

- RFC 9110：HTTP Semantics：すべてのウェブAPIの基となるメソッド、ステータスコード、ヘッダーを、IETFが定義しています。
- RFC 9205：Building Protocols with HTTP：進化の方法を含む、HTTP上のアプリケーション設計の現行の推奨実践です。
- RFC 9457：Problem Details for HTTP APIs：利用側が失敗を同じ方法で解析できる、エラー応答の標準JSON形式です。
- RFC 6585：Additional HTTP Status Codes：429 Too Many Requestsと、制限された利用側への `Retry-After` を定義しています。
- Semantic Versioning 2.0.0：非互換のAPI変更を、バージョン番号で示すルールです。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「APIとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/api/