Engineer

# バックエンド

利用者のブラウザーが読み込まない側のシステム

## どのようなものか

バックエンドは、運営者が管理するマシンで動く部分です。データを保持し、誰に何を許すかのルールを適用し、ブラウザー、アプリ、他のサービスからの要求に答えます。英語ではback end、back-endとも書き、同じ領域をサーバー側とも呼びます。

HTTPでは、RFC 9110が「指定した対象リソースについて、権威ある応答を生成できるプログラム」とするオリジンサーバーに当たります。W3CのArchitecture of the World Wide Webは、利用者側の見方を加えます。URIがリソースを識別し、ブラウザーへ届くのは、その状態の情報を符号化した表現です。顧客には注文ページが見えますが、注文、在庫、支払いを保持し、返答に入れるものを決めるのはバックエンドです。

通常、フロントエンド、モバイルアプリ、他社のシステムが呼べる、HTTPエンドポイント群のAPIを公開します。APIの仕様は、経路、入力、出力、エラーを定めます。OpenAPI Specificationは、言語に依存せず書き、クライアント、文書、テストを生成できる、一般的な記述方法です。背後では、要求処理、業務ロジック、保存などの層に分けます。

概念を判断に生かす

「バックエンドとは？」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

最初の問いは、要求のたびに動くバックエンドが、そもそも必要かです。全訪問者に同じページなら、一度生成してファイルで提供できます。注文、ログイン、非公開データの検索など、何かを変える、または要求者によって答えが変わる場合に、役立ちます。インターネットの要求に答えるソフトウェアには、防御、修正、料金が必要で、継続費用も生じます。

二つ目は信頼です。管理下のマシンで動くため、ルールを強制できる唯一の場所になります。直接要求を送る人は、フロントエンドの確認を回避できます。そのため、価格から権限まで、重要な各ルールを、ログインした利用者と具体的な記録に対して、サーバーで確認します。OWASP Application Security Verification Standardは、それを、実装・検査できる要件にしています。

三つ目はロジックの置き場所です。一つのルールがハンドラー、サービスクラス、データベーストリガーに散れば、変更先も、ずれるコピーも三つになります。コードにルールを置き、データベースを保存だけに使うチームも、データベースへルールとデータ成形を移してAPIを薄くするチームもあります。各ルールの置き場所が一つに定まっていれば、どちらも機能します。

四つ目は利用側とのインターフェース仕様です。HTTPは各応答に意味の定まったステータスコードを与え、利用側、プロキシ、監視が従います。422 Unprocessable Contentは、構文と形式は正しいが、過去の日付の予約など、意味に問題があることを示します。RFC 9457は、problem detailsという標準のエラー本文を加えます。type、title、status、detail、instanceを持つ小さなJSONを `application/problem+json` として送り、文章を解析せず失敗を区別できるようにします。

## 実際の活用例

通常、データベースからブラウザーまでは変換のリレーです。APIが行を構造体に読み、組み替え、JSONへ戻します。あるプラットフォームでは、データベースが応答を組み立て、そのまま渡します。データのある場所でPostgreSQL関数が全体をJSONにし、Goハンドラーは `json.RawMessage` として受け、そのまま送ります。小さな `QueryJSON` 補助関数によって、それが最も簡単な経路になりました。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、関数の出力変更は、ハンドラーを編集せず利用側まで届きます。

入力にも、データベースから外へつながるAPI仕様で、同じ一つの正本を使います。スキーマと関数がデータの形を、Go型がAPIを定義します。Humaが型からOpenAPIを出力し、そこから44,076行の型付きTypeScriptで書かれた、フロントエンド用のクライアントを生成します。画面でフォームに適用する制限も、バリデーターに制限値を手入力し直すのではなく、同じ場所から取得します。各段階の検査で、一方だけのフィールド変更は、数分後のビルドで失敗します。そのために必要な手間は、再生成です。バックエンドの形を変えるたび、後続の各段階を生成し直し、変更を伝える必要があります。

記録された設計段階では、このサイトは逆の選択をしました。すべてビルド時に生成し、リクエストを処理するアプリケーションのバックエンドを必要としませんでした。三言語での動作と、別料金のアプリケーションサービスを使わないことを含む四つの要件から、この方式を選びました。リクエスト時に動くアプリケーションのバックエンドには、防御、更新、プライバシーページでの説明が必要になるためです。記録にあるポートフォリオのフィルター用の限定的なJavaScript例外を除き、対話的な機能はビルド時に解決するか、省く必要がありました。これは記録された設計段階の説明であり、現在のサイトの構成や実装を示すものではありません。ホスティングに費用がかからなかったという意味でもありません。

## 問題が起きるところ

想定外の要求が来るまで、静かなままになりがちです。

**他人の記録を変更できます。** ログインだけを確認し、URLのIDを読み込むと、要求の `/invoices/1041` を `/invoices/1042` へ変えるだけで済みます。全ハンドラーが通る一か所で、触れる全オブジェクトの所有権・権限を確認し、他のアカウントの記録に403か404を期待するテストを残します。

**操作が二度起きます。** GETで取り消すリンクを、クローラーや先読みするブラウザーがたどり、クリックなしで動きます。RFC 9110がGET、HEAD、OPTIONS、TRACEを安全とするのは、クローラーや先読みが害を恐れず使えるためです。別の経路は再試行で、決済が時間切れになり、再送して二重払いします。状態変更はPOST、PUT、PATCH、DELETEに置き、再試行する書き込みには、サーバーが覚えるキーを与え、二つ目には最初の結果を返します。

**全エラーが500、または本文にエラーを含む200です。** 共通ハンドラーが同じ返し方をし、監視は入力ミスを障害と数え、再試行は必ず失敗する要求を繰り返します。入力は400・422、本人確認と権限は401・403、リソースが見つからない

場合は404、競合は409、サーバー自身の問題だけを5xxへ対応させます。詳細はproblem detailsで送ります。

```
{
    "type": "https://example.com/problems/booking-in-past",
    "title": "Booking date is in the past",
    "status": 422,
    "detail": "The date 2026-10-01 is before today.",
    "instance": "/bookings/8812"
}
```

RFC 9457は、problem detailsは「内部実装のデバッグツールではない」とするため、スタックトレースとSQLはサーバーのログに残します。

**データ増加でページが遅くなります。** 注文一覧を取得し、各顧客を個別に問い合わせると、50件の注文があるページでは、データベースとの往復が51回必要です。開発の十行では速くても、本番の一万行では遅くなります。結合で一度に取得するか、全応答をデータベースで組み立て、応答時間と同じように、要求ごとのクエリ数を監視します。

## 参考資料

- Architecture of the World Wide Web, Volume One：リソース、表現、安全・非安全な操作のW3C勧告です。
- RFC 9110：HTTP Semantics：メソッド、ステータス、安全性・冪等性のIETF標準です。
- OpenAPI Specification：人とコード生成器が読む、言語に依存しないHTTP APIの記述形式です。
- RFC 9457：Problem Details for HTTP APIs：機械可読のエラー応答の、標準JSON形式です。
- OWASP Application Security Verification Standard：ウェブアプリの構築とテストのためのセキュリティ要件です。

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

「バックエンドとは？」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/backend/