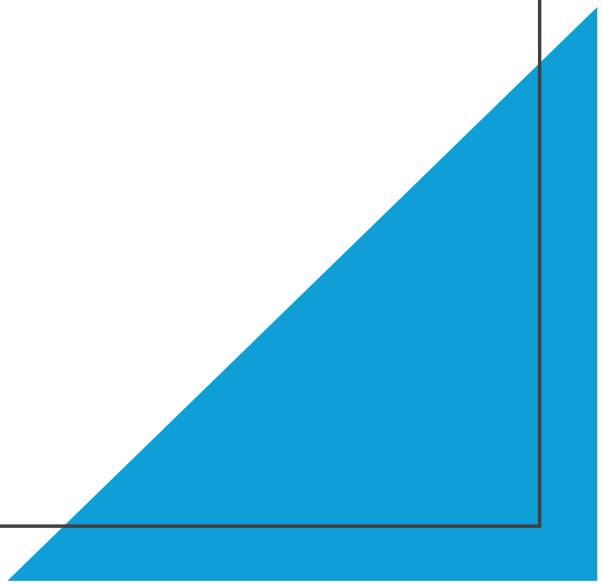
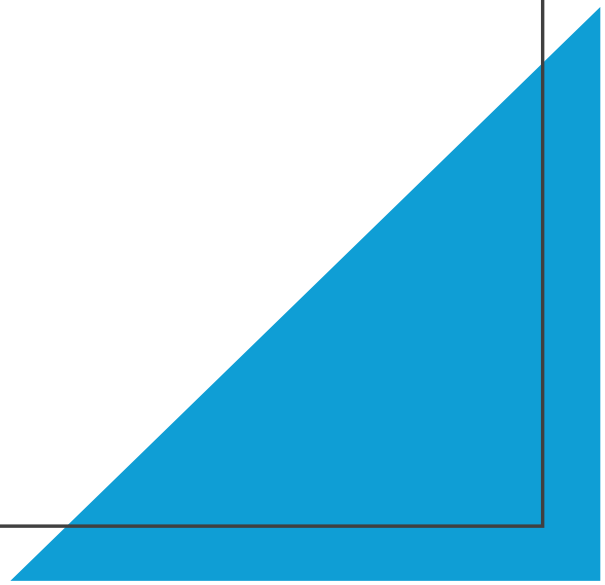
スキーマ駆動開発について



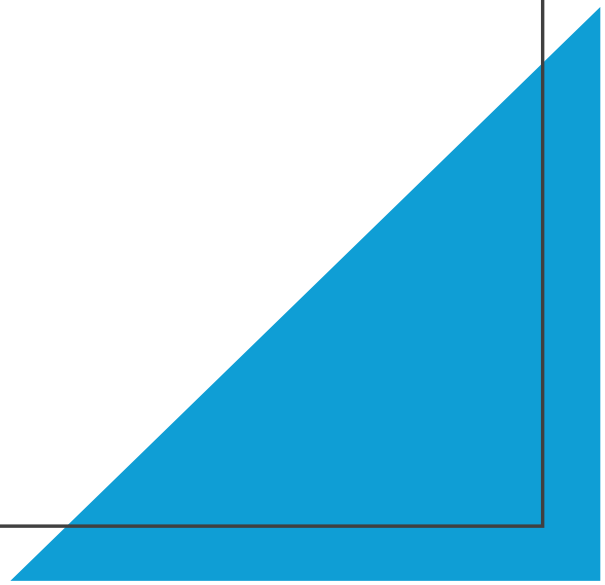
目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- まとめ



目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- まとめ



スキーマ駆動開発について

スキーマ駆動開発とは？

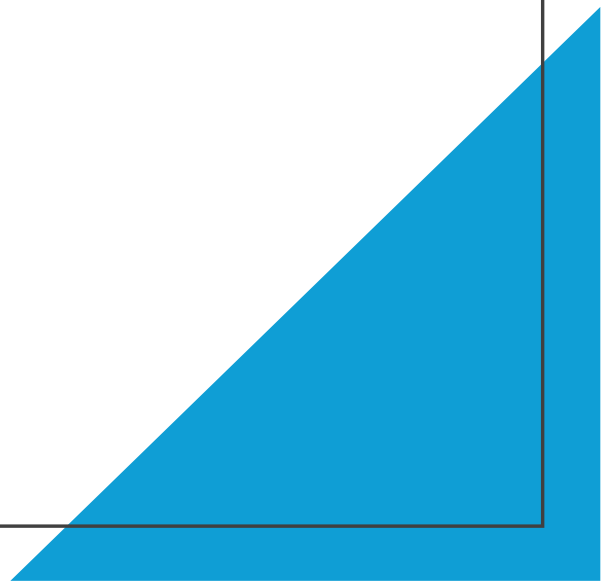
- APIやWebアプリケーションの開発を、「データ構造や形式を定義する設計図」から始める開発手法のこと。

- 「データ構造や形式を定義する設計図」
  → 「スキーマ」と定義

先に約束事を決めておく開発手法
→ “API開発”において真価を発揮

目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- まとめ



`OpenAPI`とは

（公式より）

> OpenAPI Specification (formerly Swagger Specification) is an API description format for REST APIs.

> API specifications can be written in YAML or JSON.

↓　簡略　↓

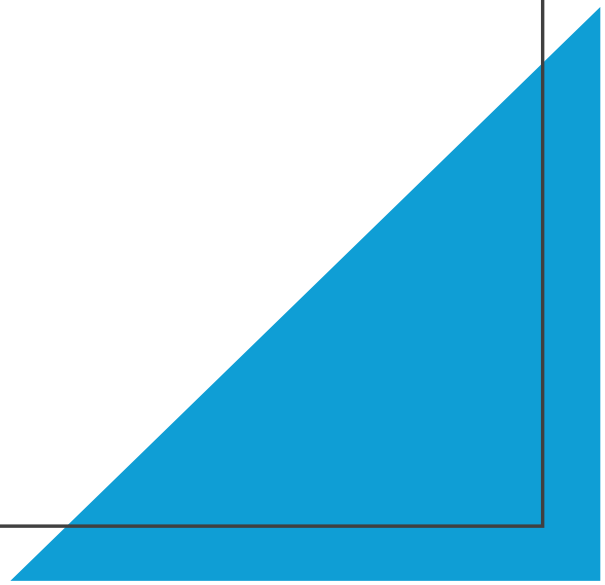
・スキーマ駆動開発を効率的に進めるための規約、ツール

・REST APIのフォーマットを、YAMLやJSON形式で記述できる！

プロジェクト関係者間の認識統一

目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- まとめ



`OpenAPI`導入に伴うメリット

（1）API記述方法の標準化

（2）ドキュメント、リクエスト・レスポンス定義の自動生成

（3）作業のスケールアウト

`OpenAPI`導入に伴うメリット

(1)API記述方法の標準化

- `YAML` or `JSON`形式の記述

```yaml
get:
  summary: サンプルユーザー情報取得
  operationId: getSampleUsers
  tags:
    - SAMPLE
  responses:
    "200":
      description: A list of users
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: "../../openapi.yaml#/components/schemas/SampleUserResponse"
```

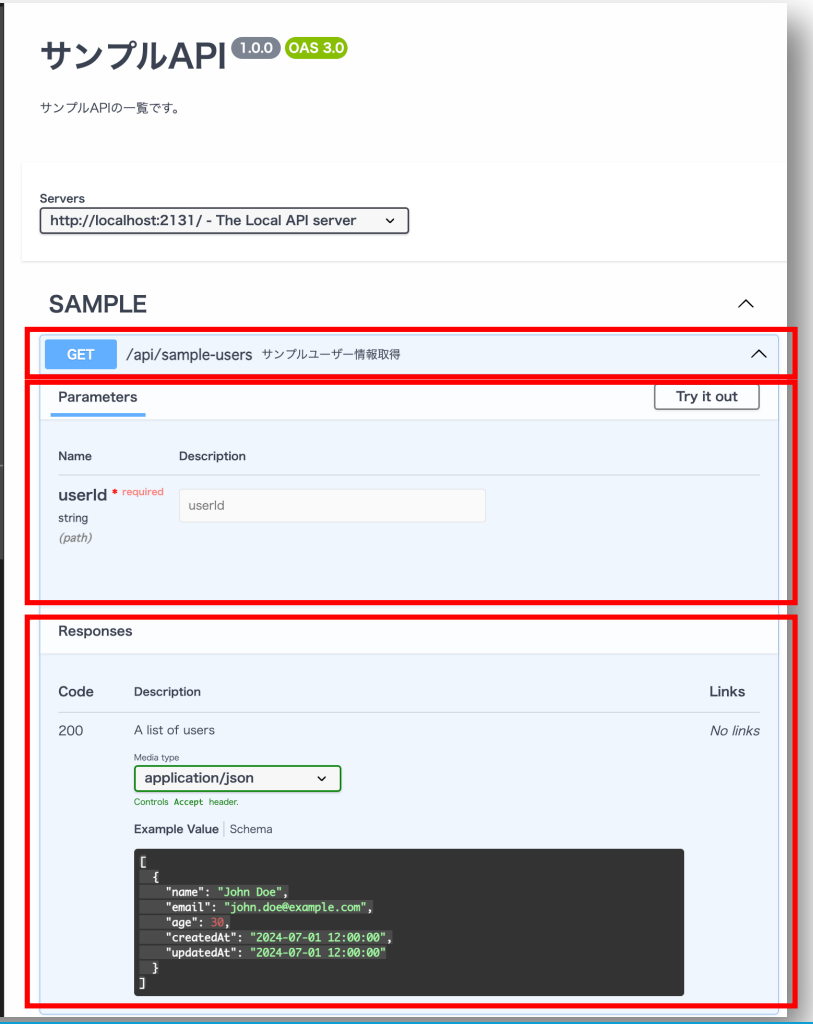
`OpenAPI`導入に伴うメリット

（2）ドキュメント、リクエスト・レスポンス定義の自動生成

→ 次ページに実際のサンプル

ドキュメント

YAML



エンドポイント

リクエスト

レスポンス

自動生成コードを利用したAPIコール

③
エディタ上で
ツールチップ参照

```
(method) SAMPLEApi.getSampleUserById(userId: string, options?: RawAxiosRequestConfig): Promise<AxiosResponse<SampleUserResponse, any>>
@summary — サンプルユーザー情報取得
@param userId
@param options — Override http request option.
@throws — {RequiredError}
@memberof — SAMPLEApi
```

```
import { SAMPLEApi } from '@/types/OpenAPI/api';

const api = new SAMPLEApi(undefined, '', axiosInstance);
const response = await api.getSampleUserById(id);

console.log(response.data);
```

①
Import対象は
自動生成されたファイルのみ

②
呼び出したいAPIの
メソッドを記述するだけ

`OpenAPI`導入に伴うメリット

（3）作業のスケールアウト

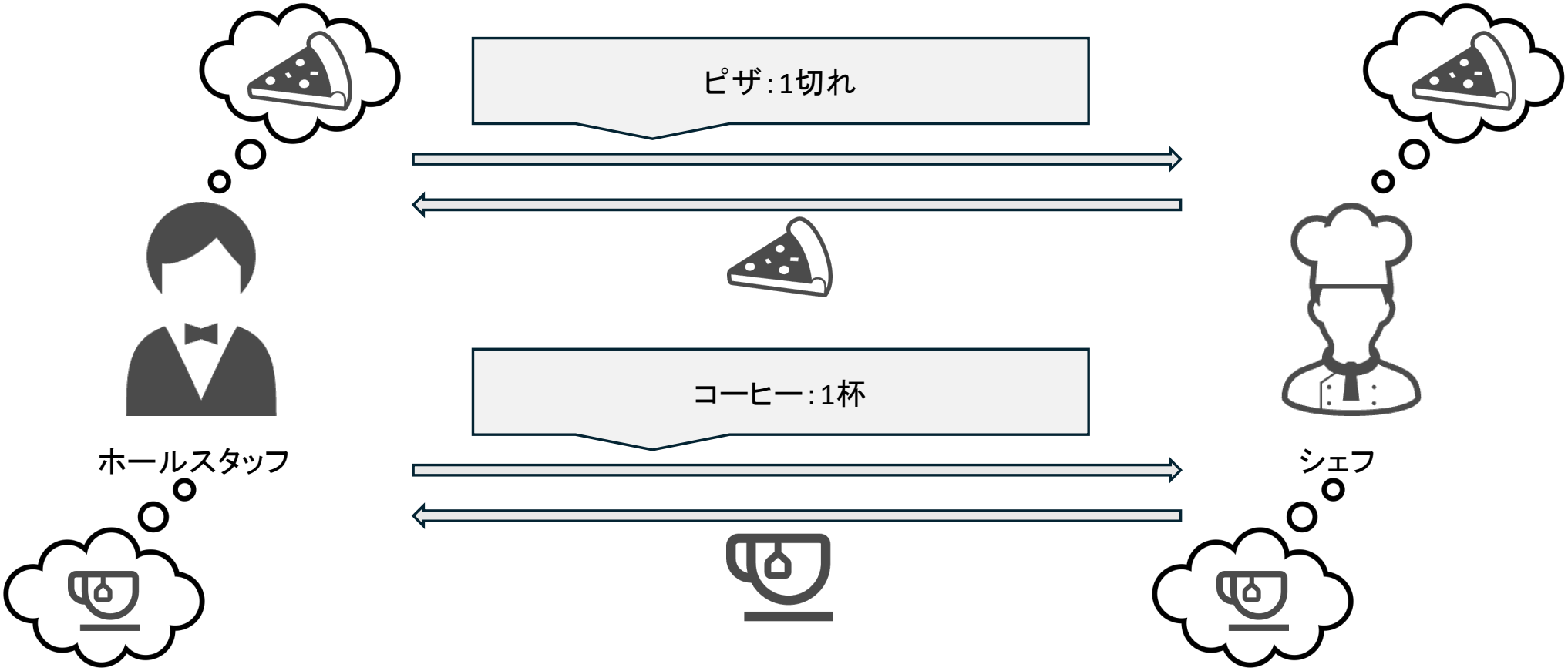
- OpenAPIさえ確定すれば、「 Frontend 」「 Backend 」の開発順序は問わない。

スキーマ駆動開発について

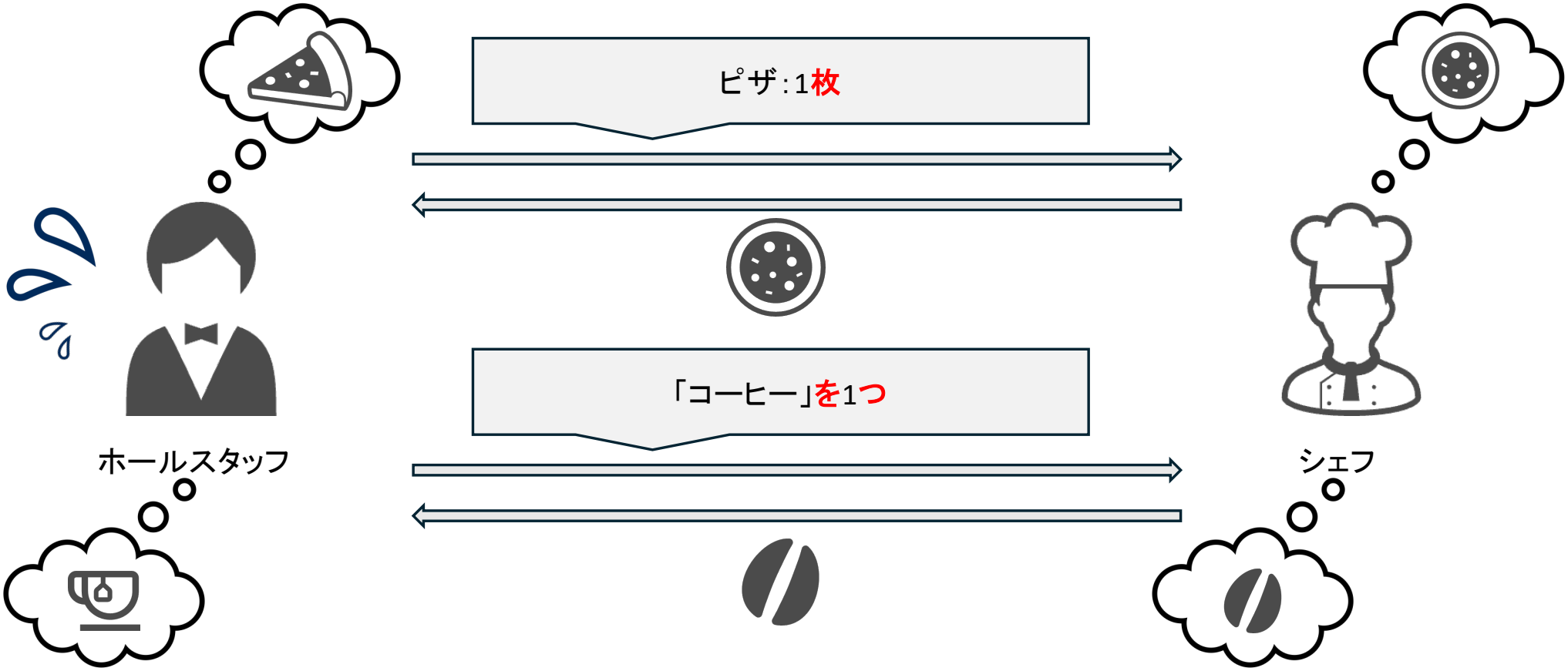
現実世界に置き換えると...

レストラン？？

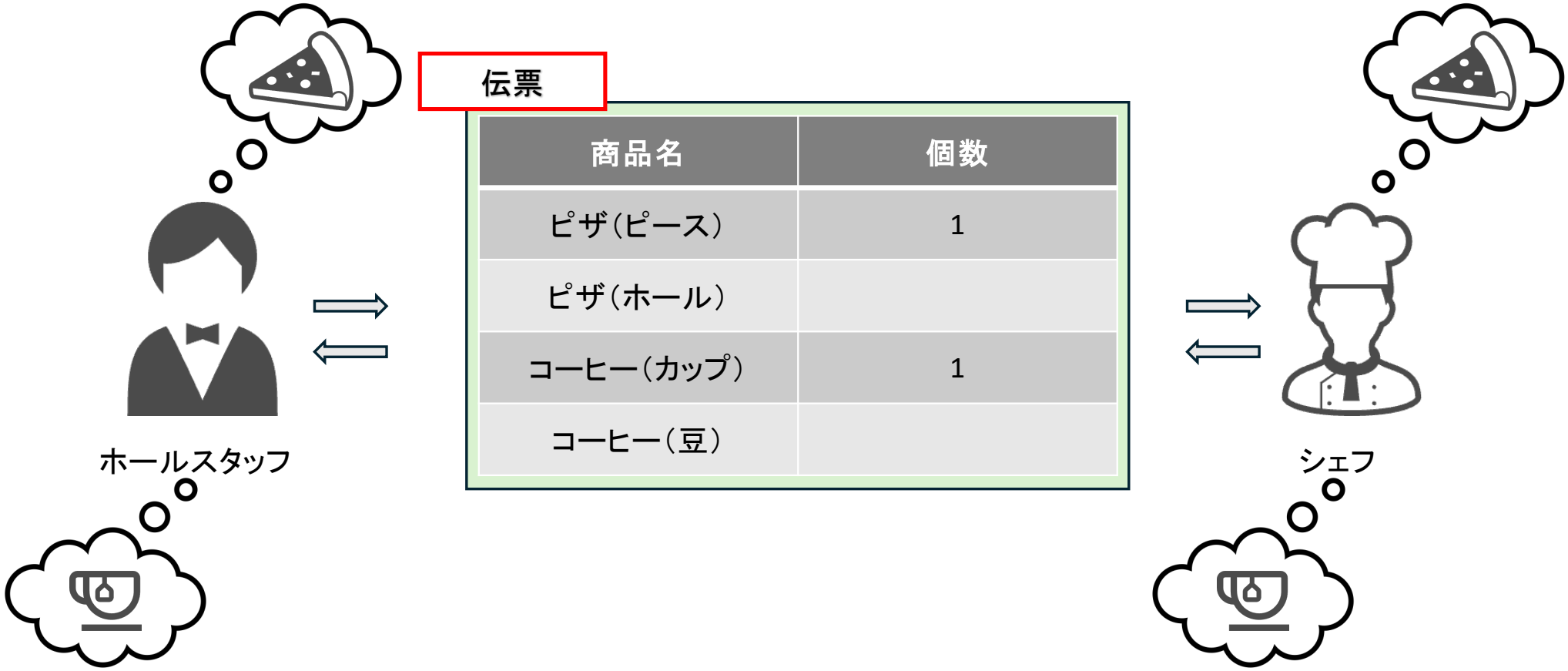
現実世界で考えると...



現実世界で考えると…

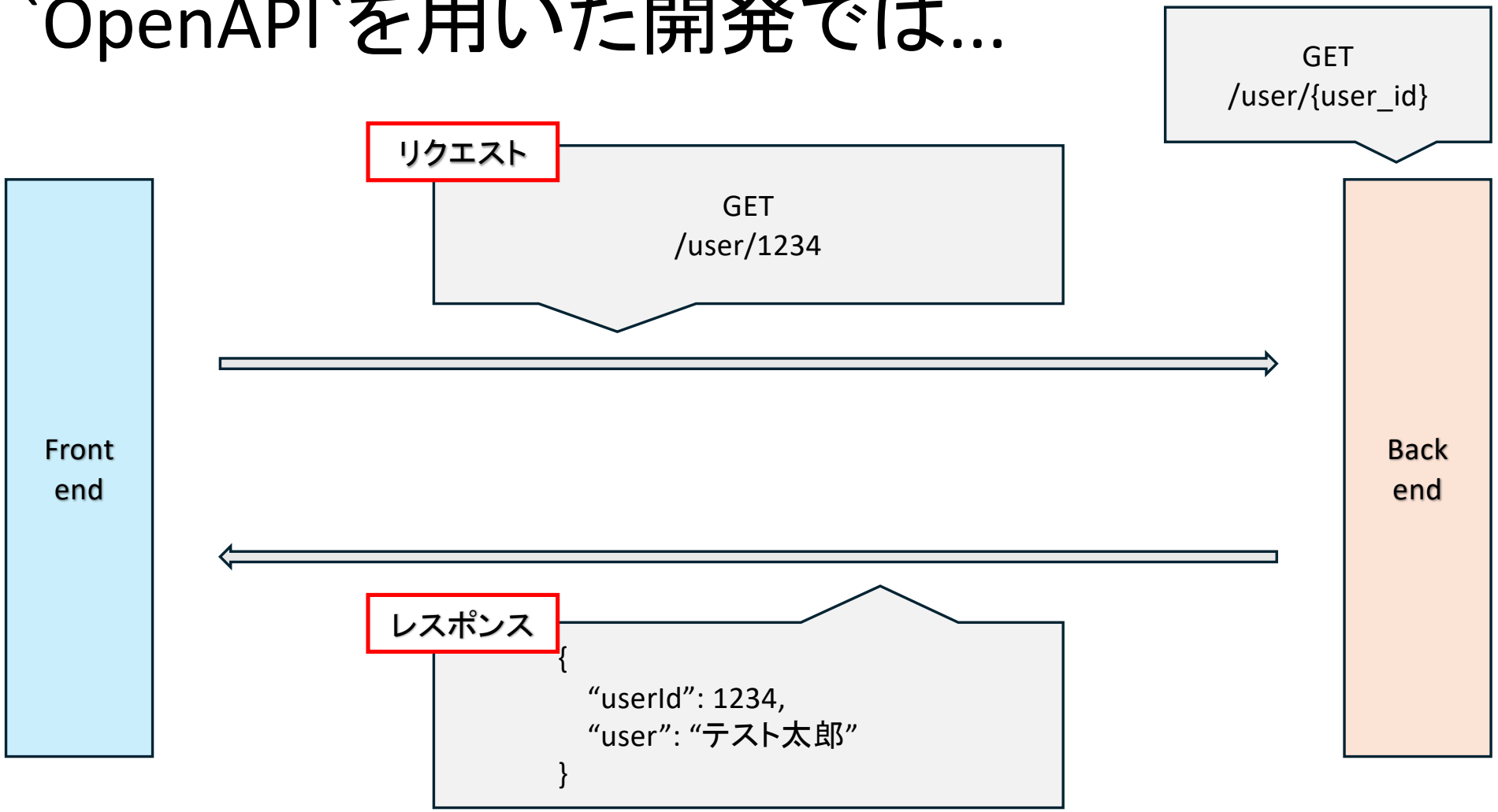


現実世界で考えると…

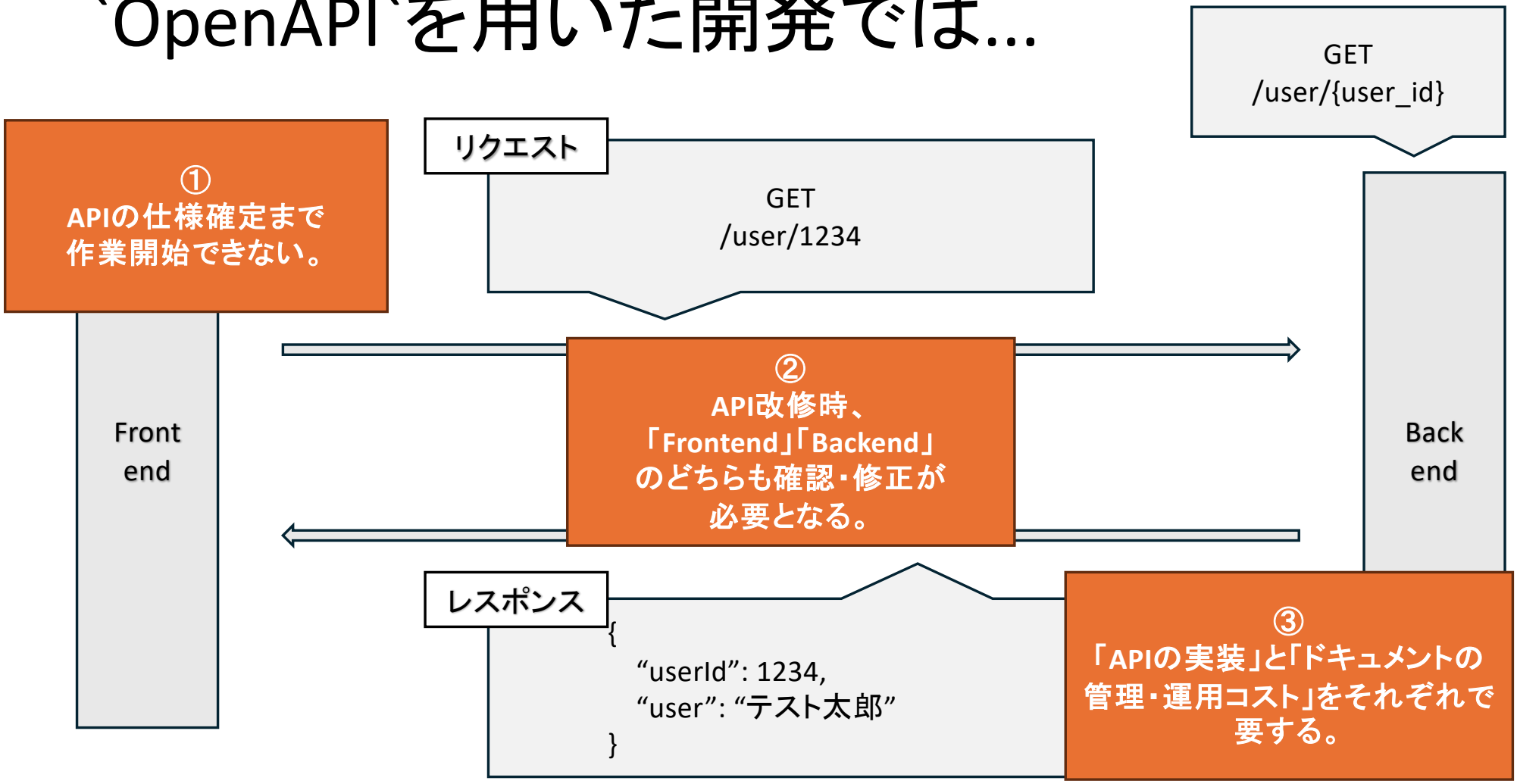


現実世界で考えると…

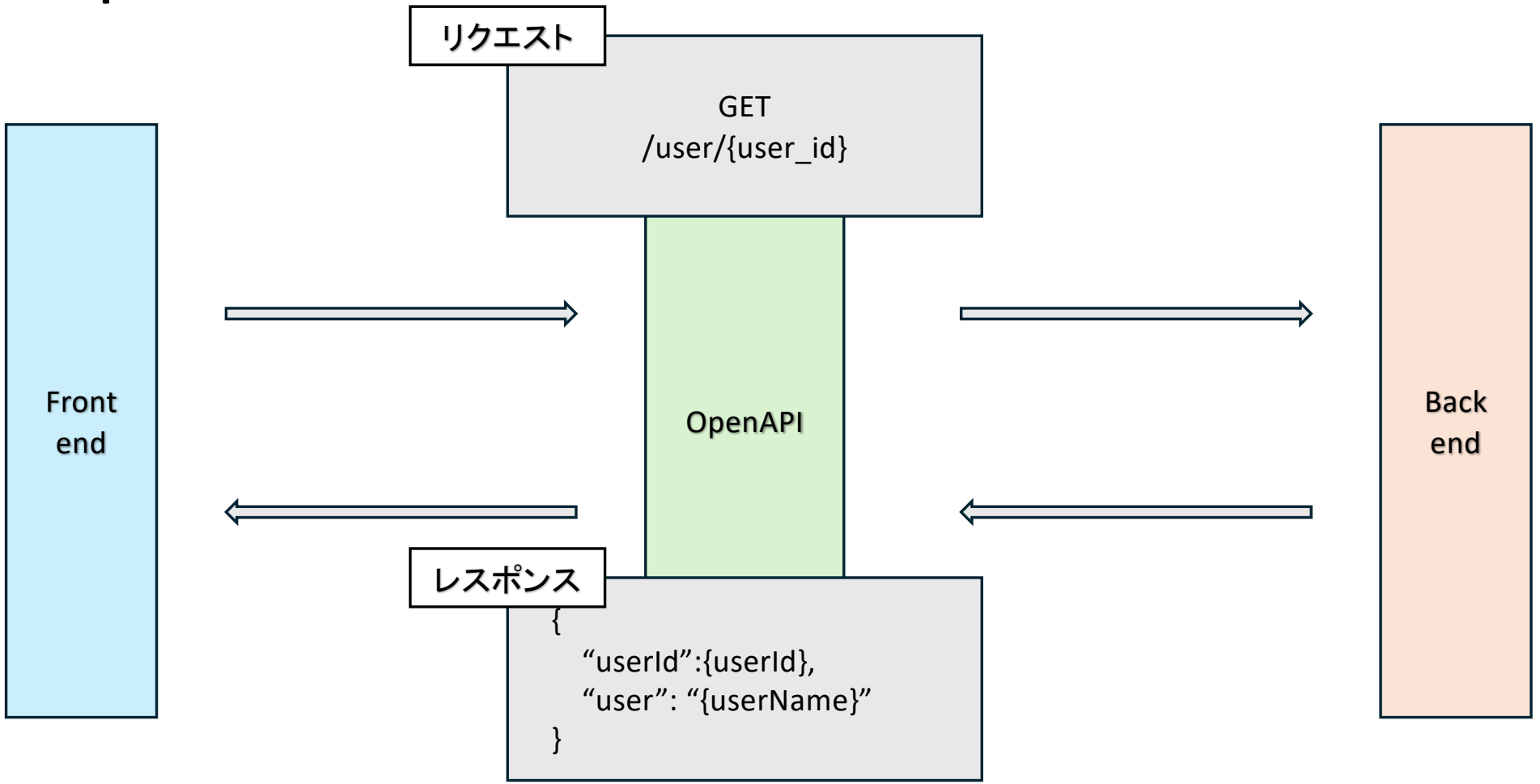
`OpenAPI`を用いた開発では…



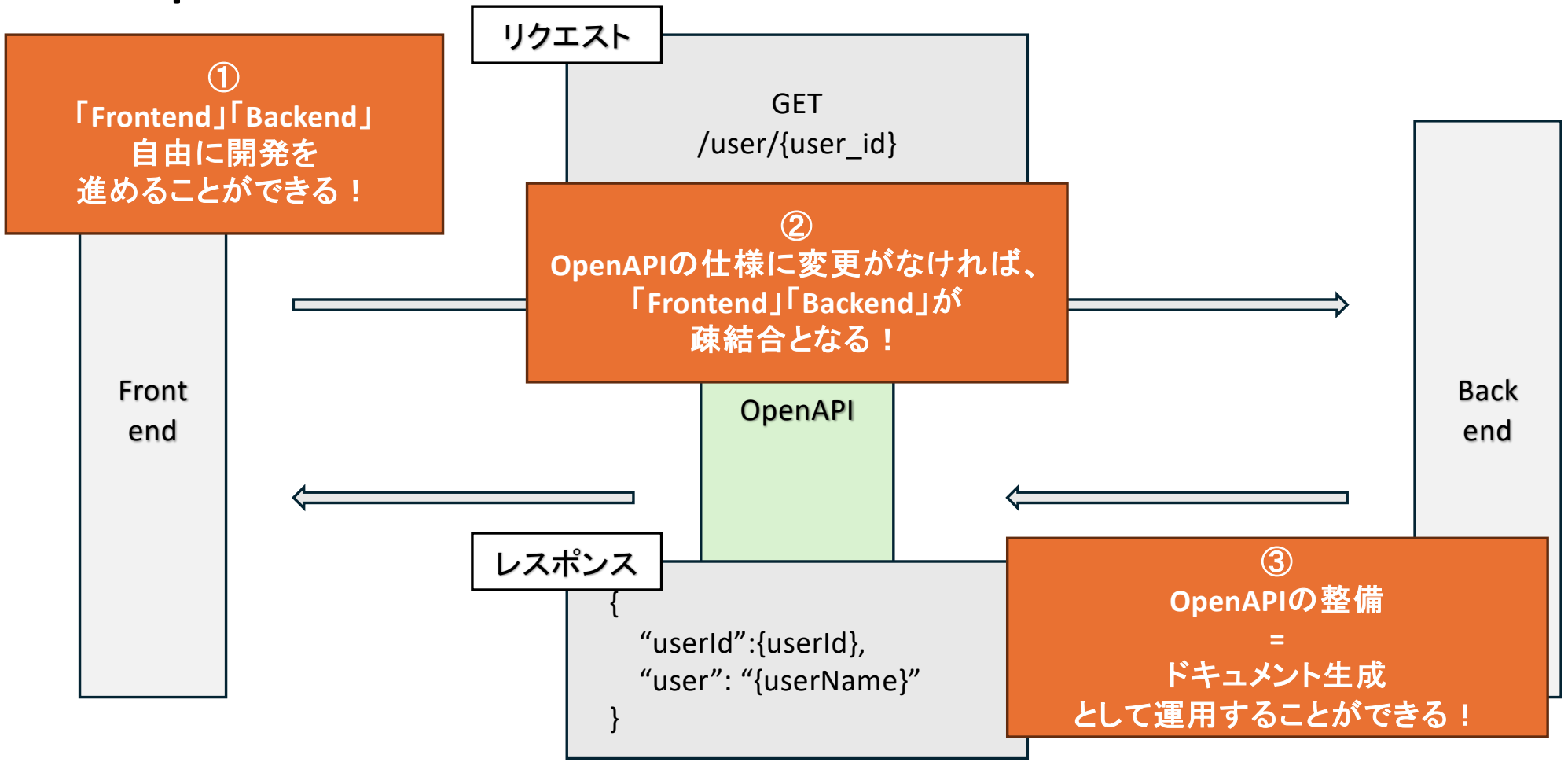
`OpenAPI`を用いた開発では…



`OpenAPI`を用いた開発では…

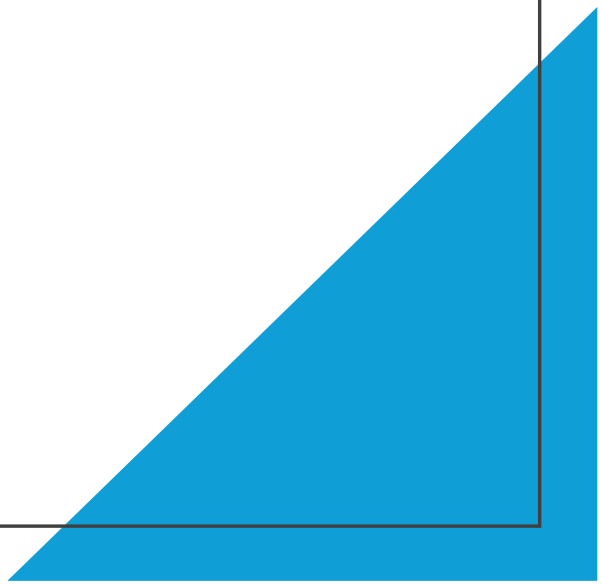


`OpenAPI`を用いた開発では…



目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- `まとめ



プロジェクトへの導入目的

(1)プロジェクト関係者間の認識を統一

- OpenAPIを通じて、プロジェクト関係者全員が API の動作やインターフェースを通じて理解

(2)開発の効率化

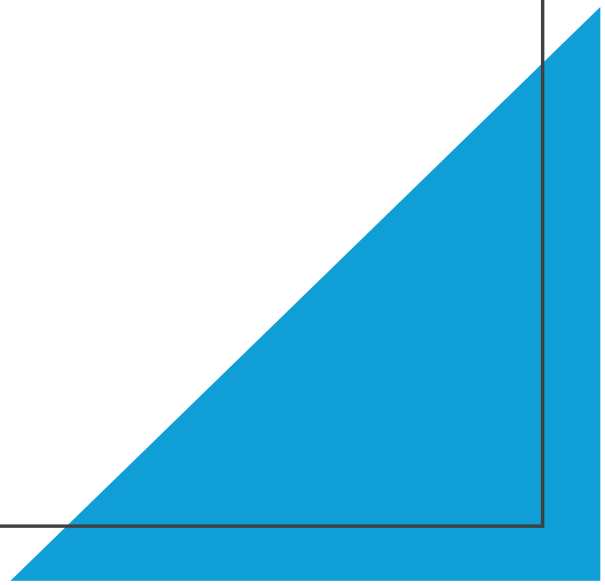
- API 仕様が明確に定義することで、FrontendとBackendの順序を問わない開発を推進

(3)ドキュメントメンテナンスの容易化

- API仕様変更に伴うドキュメント整備工数の削減

目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- まとめ



実施内容

（1）OpenAPI記載のAPIを正とした開発の実施

- OpenAPIファイルに記載してあることのみを正とする。
  - 設計書の記載内容に優先

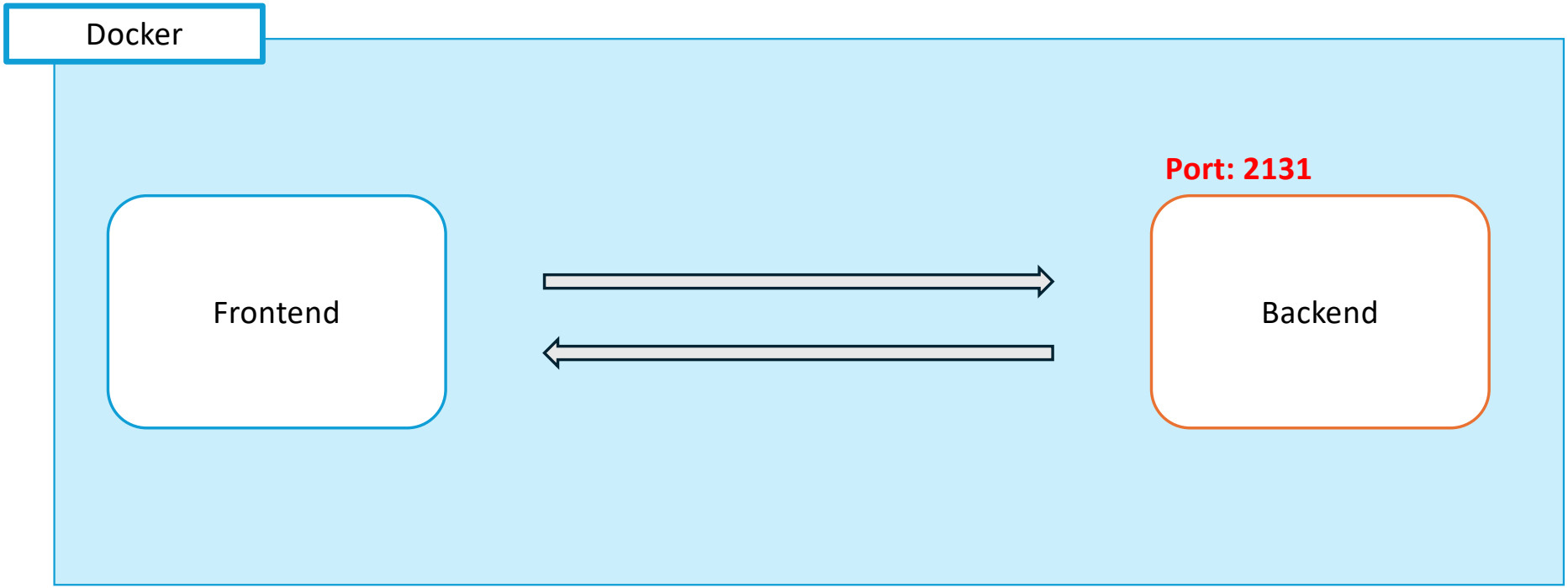
（2）開発はOpenAPI定義の反映から

- APIの新規作成・修正を行う際は、必ずOpneAPIファイルの更新から始める。
  - Frontendのリクエスト処理・Backendのバリデーション・レスポンス定義は、必ずOpenAPIファイルから自動生成されたソースコードを利用

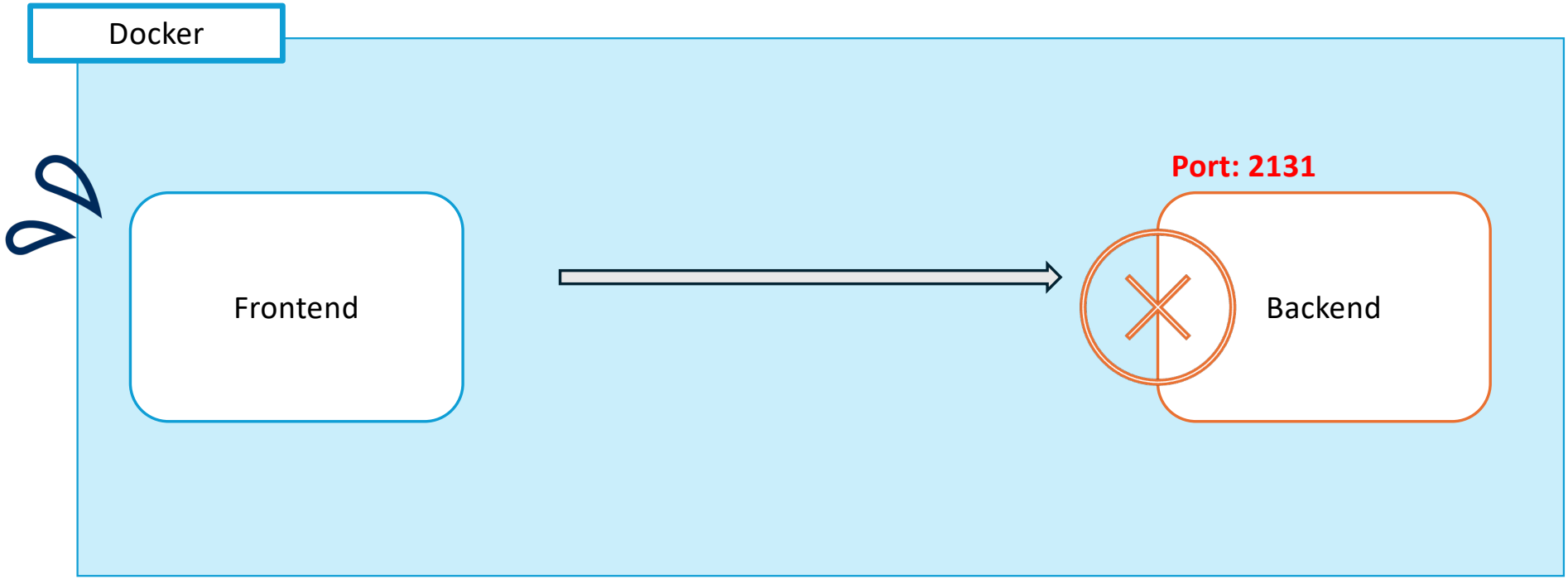
（3）OpenAPIファイルを用いたMockサーバーの利用

- Frontend開発、及び、Backend開発の順不同を実現

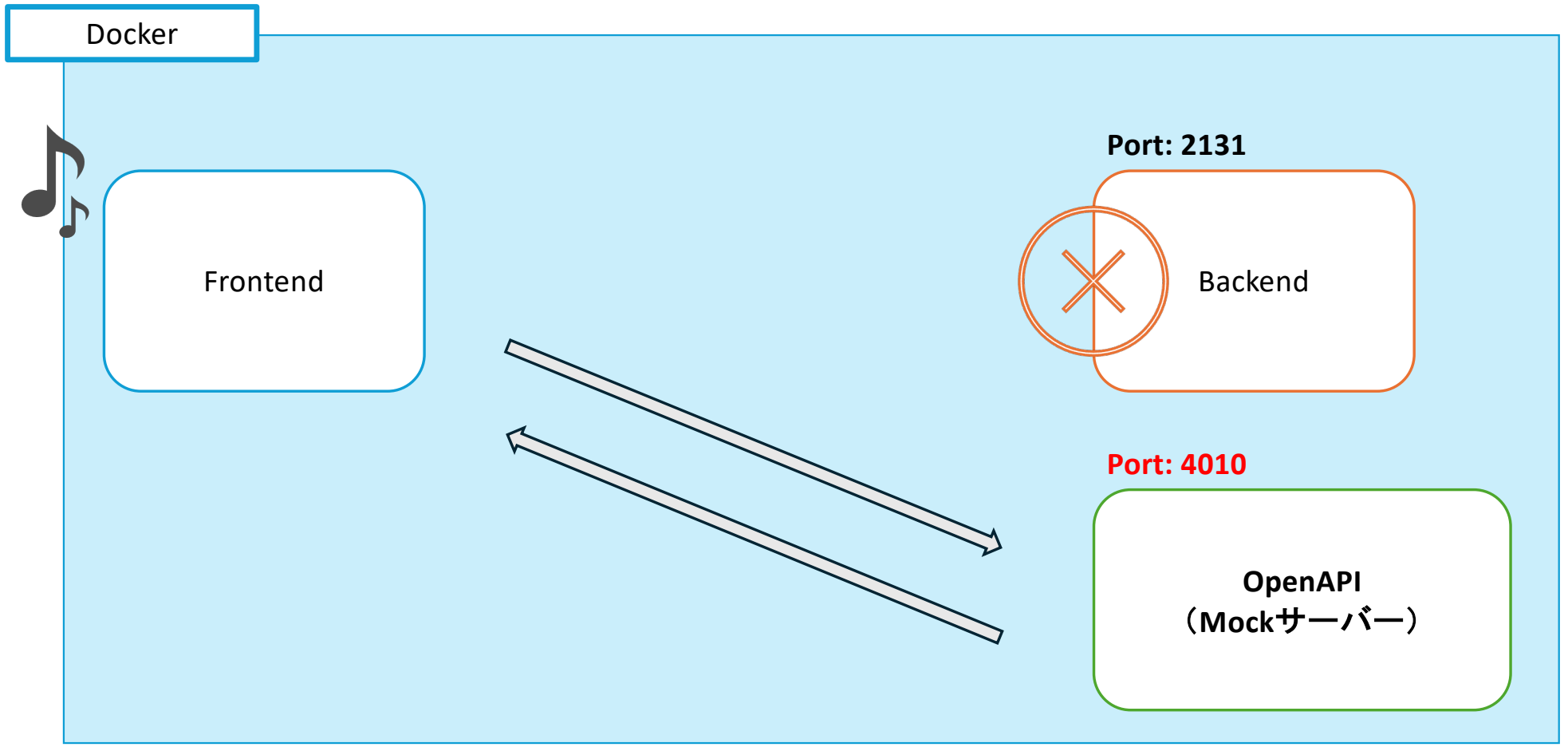
OpenAPIファイルを用いたMockサーバーの利用



OpenAPIファイルを用いたMockサーバーの利用

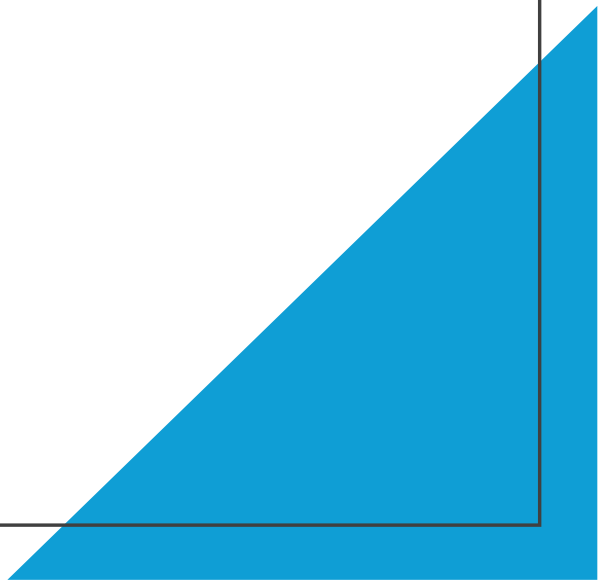


OpenAPIファイルを用いたMockサーバーの利用



目次

- スキーマ駆動開発について
- `OpenAPI`とは
- `OpenAPI`導入に伴うメリット
- プロジェクトへの導入目的
- 実施内容
- `OpenAPI`導入に伴うチームメンバーの評価
- まとめ



まとめ

1. OpenAPIを導入することで、開発工数を削減できる
2. OpenAPIを使用したことがないメンバーのフォローは必須
3. OpenAPIの真価が発揮されるのは、保守・運用開始後