Three approaches in API Development

Enis Spahi

About me

Consultant Architect at OpenValue

enis@openvalue.de

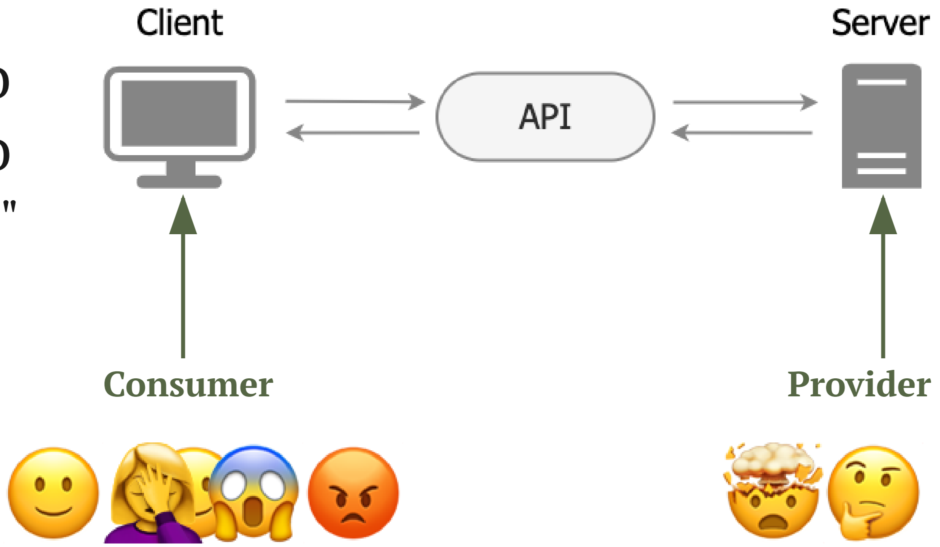
enisspahi

enisspahi

APIs

"An application programming interface (API) is a way for two or more computer programs to communicate with each other."

Source: Wikipedia, "API"



Boundaries

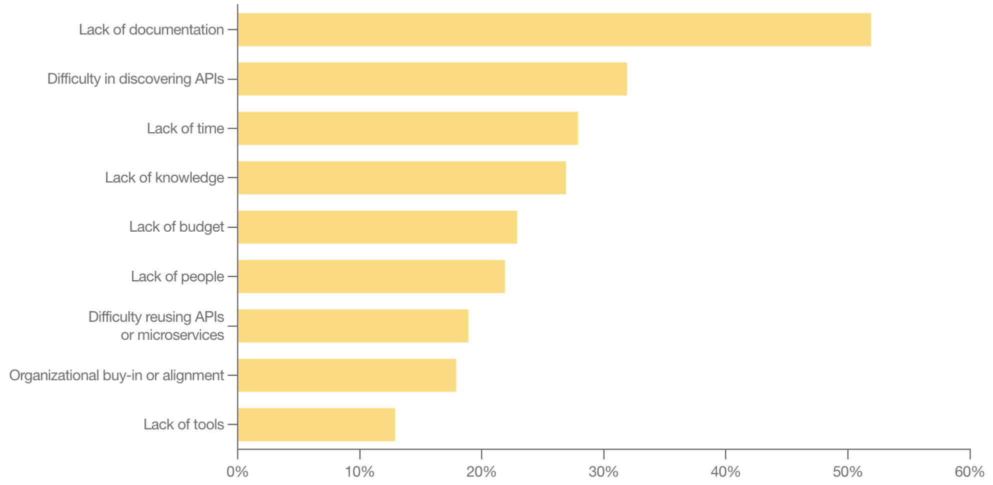
- Private APIs
  - Provider and consumer are developers in the same team or same organisation

- Partner Facing APIs
  - Serving partners (i.e.: Payment Service Providers)
  - Provider and consumer might not communicate directly

- Public APIs
  - Publicly available (i.e… Geo-Location services, LLMs)
  - Communication at scale: Many Consumers ↔ 1 Provider

Statistics

Obstacles to consuming APIs

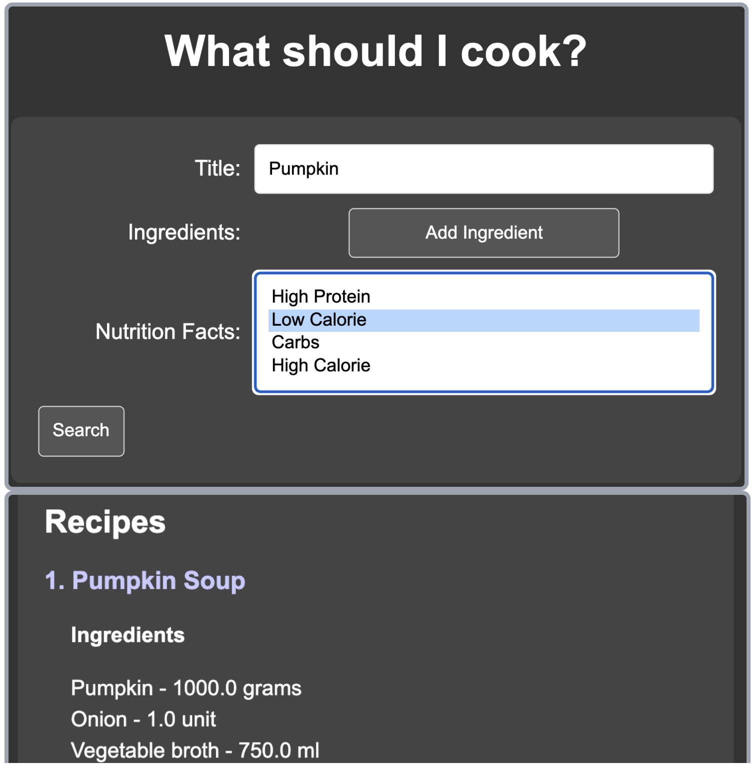


Source: Postman, "2023 State of the API Report"

Let’s build an API

Recipes API

Client



What should I cook?

Title: Pumpkin

Ingredients: Add Ingredient

Nutrition Facts: High Protein / Low Calorie / Carbs / High Calorie

Search

Recipes

1. Pumpkin Soup

Ingredients

Pumpkin - 1000.0 grams
Onion - 1.0 unit
Vegetable broth - 750.0 ml

Server

```
curl 'http://localhost:8080/recipes?title=Pumpkin&nutritior
```

```
[
    {
        "title": "Pumpkin Soup",
        "ingredients": [
            {
                "name": "Pumpkin",
                "quantity": 1000.0,
                "unit": "grams"
            },
            {
                "name": "Onion",
                "quantity": 1.0,
                "unit": "unit"
            },
            {
                "name": "Vegetable broth",
                "quantity": 750.0,
                "unit": "ml"
```

Find a common language

OPENVALUE

OpenAPI Specification

- Technology agnostic standard to describe Rest APIs
- Formerly Swagger, OpenAPI as of version 3
- Written as JSON or YAML
- Great tooling for code and documentation generation
- https://openapi.tools/

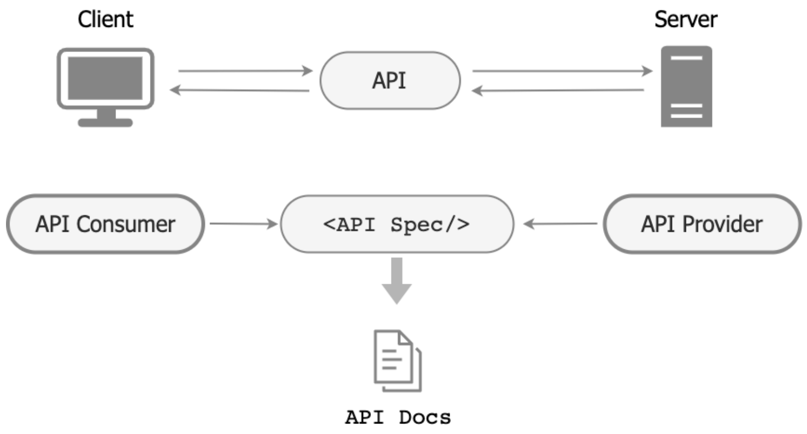
OpenAPI Specification

```yaml
openapi: 3.0.3
info:
  title: Recipes API
  ...
servers:
- url: http://localhost:8080
paths:
  /recipes:
    get:
      summary: List all recipes
      ...
      responses:
        ...
        "200":
          description: OK
          content:
            'application/json':
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Recipe'
```

- **Openapi:** Spec Version
- **Info:** General API information as metadata
- **Servers:** Connectivity information about target servers
- **Paths:** Paths to the endpoints with their expected request, response and errors.
- **Components:** Holds the schemas for the request, response and errors for referencing

OPENVALUE

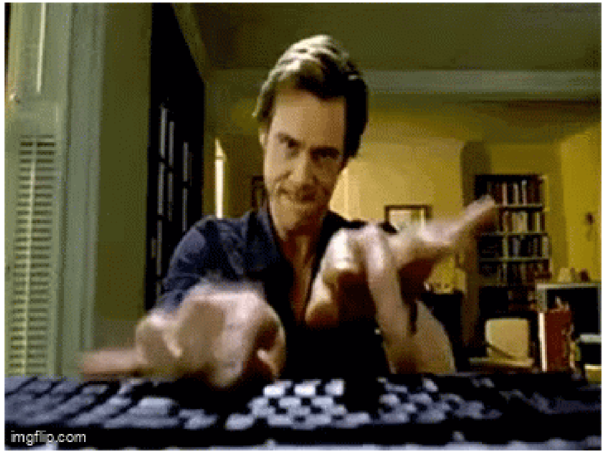
Standardized API Communication



- Common Language for API discovery
- Foundation for tooling
  - Code generation
  - Documentation
- Community

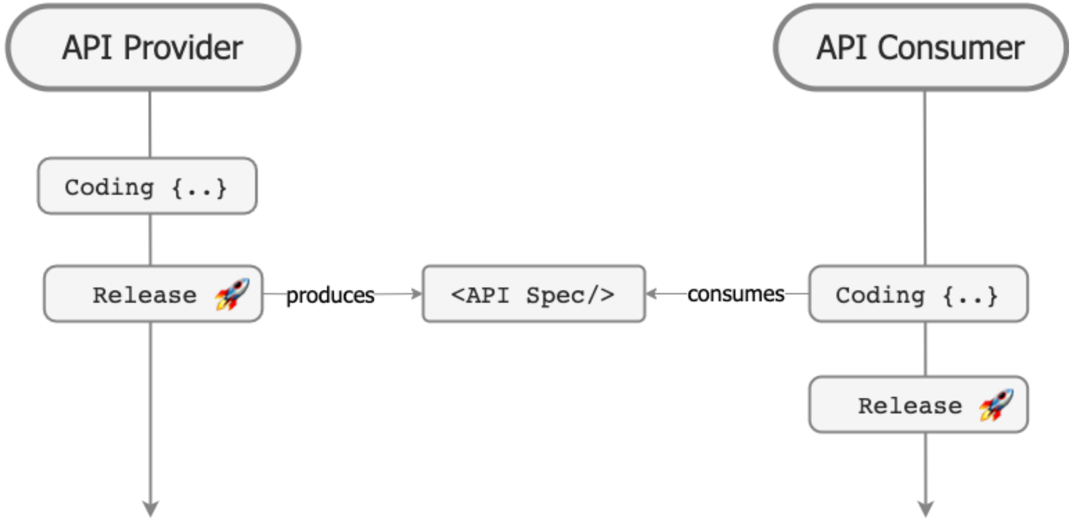
3 Approaches

1. Code First



Code First

Communicate API specification once coding has been done



Code First

Communicate API specification once coding has been done

Advantages:

- Focus on coding
- Flexibility to change the API design
- Client code generation
- Documentation generation

Disadvantages:

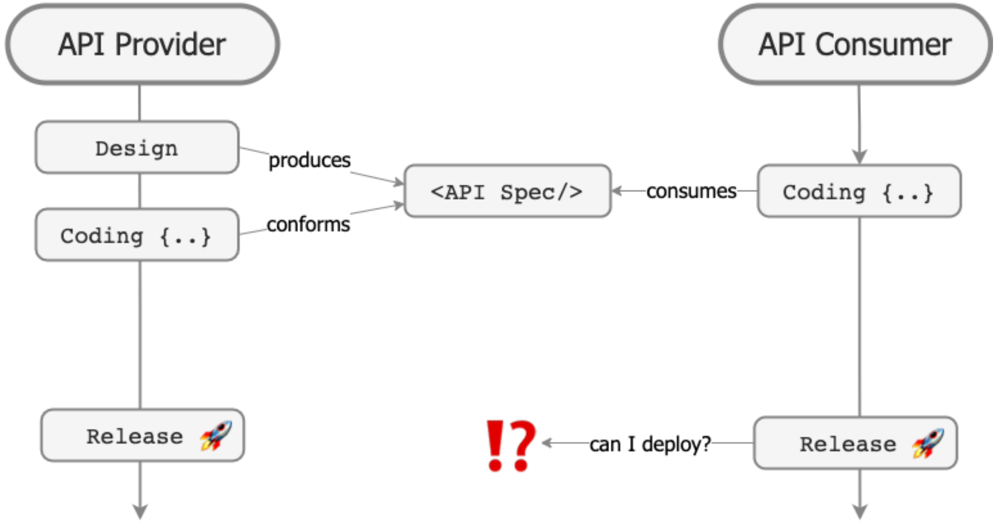
- Late communication with the consumer
- Does not enable development in parallel
- Annotations

2. API First

OPENVALUE

API First

Communicate API specification before coding. Prioritizes API design over implementation.



OPENVALUE

API First

Communicate API specification before coding. Prioritizes API design over implementation.

Advantages:

- Early communication with the consumer
- Enables development in parallel
- Documentation thought ahead
- Client and server code generation

Disadvantages:

- Less flexibility to change the API design
- Sometimes bureaucratic for providers
- Requires Integration Testing
- "Can I deploy?" challenge

3. Consumer First

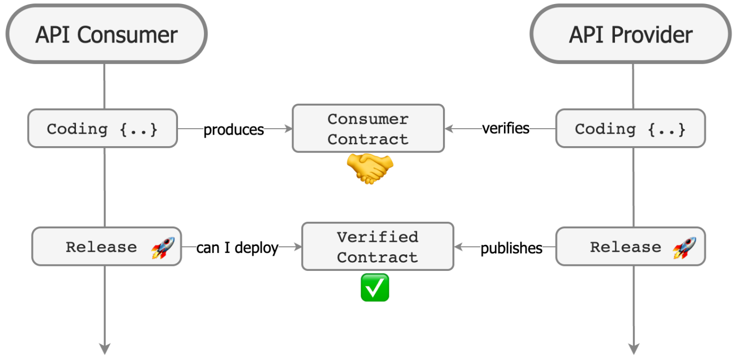
Consumer First

Consumer dictates the expected API behavior to the provider.

Pact: A Code-first consumer contract testing tool that enables consumer driven API development.

Process:

- Consumer produces a pact
- Provider verifies it's API implementation
- Server / Client deployments synced



OPENVALUE

The right methodology?

The right methodology?

When to use Code first?	Provider initially focuses on coding speed Flexibility to change the design
When to use API first?	API design over implementation Early communication with the consumer → Documentation Utilize code generation Large number of consumers
When to use Consumer first?	Provider should conform to consumer needs API consumer and provider test their applications independently To sync provider and consumer deployments Small number of consumers
When to mix & match?	When API first alone is not sufficient to match consumer needs

Message-driven APIs?

AsyncAPI Specification

- Technology agnostic standard to describe message-driven APIs
- An adaptation of the OpenAPI specification
- Written as JSON or YAML
- Protocols: AMQP, HTTP, JMS, Kafka, but not only
- https://www.asyncapi.com/tools

AsyncAPI Specification

```yaml
        $ref: '#/components/messages/Pong'

components:
  messages:
    Ping:
      name: Ping
      ...
      payload:
        $ref: '#/components/schemas/PingPayload'
    Pong:
      name: Pong
      ...
      payload:
        $ref: '#/components/schemas/PongPayload'

  schemas:
    PingPayload:
      type: object
      ...
    PongPayload:
      type: object
      ...
```

- **Asyncapi:** Spec Version
- **Info:** Metadata information about the API
- **Servers:** Connectivity information about servers (i.e. Kafka brokers)
- **Channels:** Messages exchange between provider and consumer
- **Components:** Defines the reusable objects such as schemas or messages which could be referenced.

Thank you for your attention!

https://github.com/enisspahi/3approaches-apis

OPENVALUE