

# Membuat Dokumentasi API dengan Format Yaml

---



**“Janganlah pernah menyerah ketika kamu masih mampu berusaha lagi. Tidak ada kata berakhir sampai kamu berhenti mencoba” - Brian Dyson**

# Main Objective

---

Mengenal apa itu swagger

---

Mengenal tools/jenis swagger

---

Mengenal fungsi swagger

---

Mengenal apa itu Api

---

Mengenal jenis-jenis Api

---

Mengenal arsitektur Api

---

Mengenal komponen swagger

---

Mengenal komponen pada object paths

---

Mengetahui step by step pembuatan dokumentasi Api

---

Mengetahui cara test Api pada swagger

# Apa itu Swagger?

Swagger adalah *tools* yang digunakan untuk mendokumentasikan API atau web service yang dibangun.

Ada 3 jenis dokumentasi API, yaitu adalah sebagai berikut :

- **API Documentation**

Dokumentasi API yang diperuntukkan kepada pengguna agar mudah dibaca dan dipahami.

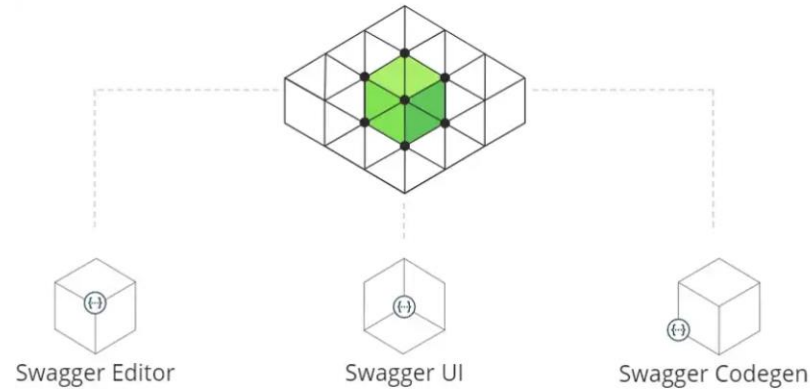
- **API Specification**

Lebih terstruktur serta programmable yang menjelaskan bagaimana suatu API tertentu bekerja, karakteristik serta output yang akan dihasilkan.

- **API Definition**

Dokumentasi API yang ditujukan untuk mesin agar dapat diotomatisasi serta dieksekusi di server.

# Tools Swagger



Sumber : <https://swagger.io/tools/>

**Swagger Editor**  
Editor API untuk mendesain API dengan OpenAPI Specification dan dapat disimpan dalam format yaml atau json.

**Swagger UI**  
UI interaktif yang dapat digunakan di browser untuk memvisualisasikan definisi OpenAPI Specification.

**Swagger Codegen**  
Code generator yang dapat digunakan untuk membuat server stubs dan client SDKs secara langsung dari OpenAPI Specification yang telah kita buat.

# Fungsi Swagger

---

1. API Design  
Swagger membuat desain API menjadi lebih mudah dengan menggunakan *tools* yang mudah digunakan oleh pengembang.
2. API Development  
Swagger menyediakan *tools* untuk membuat prototipe dan membangun fungsionalitas API dengan cepat.
3. API Documentation  
Swagger dapat membuat dokumentasi serta memvisualisasikan dan memelihara dokumentasi API.
4. API Testing  
Swagger dapat digunakan untuk pengujian fungsional, keamanan dan kinerja dari OpenAPI Specification. *Tools* Swagger memudahkan pembuatan, pengelolaan dan pelaksanaan pengujian API dengan cepat.
5. API Mocking and Virtualization  
Swagger dapat mengurangi ketergantungan dalam siklus pengembangan dan pengujian, serta tiruan realistis dan dinamis dengan menggunakan Swagger dan ReadyAPI Virtualization.
6. API Governance  
Swagger dapat mengelola program API secara bersama dengan tim yang terdiri dari lima orang atau lebih.
7. API Monitoring  
Swagger dapat memantau ketersediaan, kecepatan dan fungsionalitas API mulai dari Spesifikasi OpenAPI (OAS) dengan AlertSite untuk OAS. Swagger dapat memonitoring dan lacak perilaku API secara instan untuk memastikan semuanya berjalan dengan lancar.
8. OpenAPI & Swagger  
Swagger menyediakan *tools* yang mudah digunakan untuk memanfaatkan kemampuan Spesifikasi OpenAPI (OAS).

# Apa itu API?

API merupakan singkatan dari *Application Programming Interface* yaitu merupakan sebuah *interface* yang menghubungkan antara satu aplikasi dengan aplikasi lainnya. API memperlihatkan sekumpulan data dan fungsi untuk memfasilitasi antar aplikasi serta memungkinkan untuk bertukar informasi. API digunakan sebagai penerjemah komunikasi antara *client* dengan *server*. API bermanfaat untuk memudahkan membangun aplikasi yang fungsionalitas dan kompleks serta dapat mengembangkan aplikasi menjadi lebih efisien.

# Jenis-jenis Api

Berdasarkan penggunaannya, terdapat empat jenis API sesuai dengan hak aksesnya yaitu adalah sebagai berikut:

## 1. Public API

Public API sering disebut juga dengan Open API yang akan menjadi fokus pembahasan kita kali ini. API jenis ini dapat digunakan oleh siapa saja dalam lintas platform yang berbeda.

## 2. Private API

Private API tidak dapat digunakan secara umum, biasanya API jenis ini dibuat untuk penggunaan internal dalam pengembangan aplikasi tertentu.

## 3. Partner API

Partner API dapat digunakan secara umum tetapi hanya sebatas pihak yang sudah memiliki izin penggunaannya.

## 4. Composite API

Composite API merupakan API yang dapat menyimpan data-data dari berbagai server.



# Arsitektur Api

API dibangun menggunakan arsitektur tertentu, terdapat beberapa Arsitektur API yaitu adalah sebagai berikut:

- RPC (Remote Procedure Call)

RPC merupakan arsitektur API yang digunakan untuk membantu kinerja dari client side dan server side dalam hal komunikasi dengan menggunakan konsep yang sederhana. RPC memiliki dua jenis berdasarkan media perpindahan datanya yaitu XML-RPC dan JSON-RPC.

- SOAP (Simple Object Access Protocol)

SOAP merupakan arsitektur API yang dimana data yang digunakan berupa XML.

- REST (Representational State Transfer)

REST merupakan arsitektur API yang sangat populer pada saat ini. Kelebihan dari REST yaitu memiliki bentuk data berupa JSON sehingga lebih ringan serta mudah dalam proses pengembangannya.

# Komponen Swagger

Swagger memiliki object untuk mendokumentasikan API, berikut adalah beberapa object swagger

**Supermarket Swagger - OpenAPI 3.0** 1.0.11 OAS3 → **title**

berisi informasi tentang apa saja barang yang dijual di supermarket seperti buah,sayur,daging → **Description**

Terms of service → **Terms of service**

Contact the developer → **contact**

Apache 2.0 → **license**

Find out more about Swagger → **External docs**

Servers

https://supermarket/api/v3 → **servers** Authorize

**buah** semua tentang buah buahan ada disini Find out more

**sayur** semua tentang sayur sayuran ada disini Find out more

**daging** semua tentang daging-dagingan ada disini

Keterangan dari setiap komponen/object openapi swagger:

1. **info:** Penjelasan tentang dokumentasi swagger yang dibuat yang isinya ada **title** swagger, **description** dari swagger tersebut, **termsOfService** s&k pakai api tersebut, **contact** yg bisa dihubungi, **license** api tersebut serta **version** api tersebut
2. **externalDocs:** info tambahan diluar dari object openapi info
3. **servers:** informasi url yang digunakan oleh api
4. **tags:** grouping/pengelompokan api, contoh pada gambar disamping ada 3 grup api yaitu buah, sayur, daging
5. **paths:** untuk membuat url directory api contoh: /buah(artinya url/path akan masuk ke dalam tags buah)

**tags**

**buah** semua tentang buah buahan ada disini Find out more

- GET** /buah Get semua data buah
- PUT** /buah Update data buah yang sudah ada
- POST** /buah tambah data buah di supermarket
- GET** /buah/{buahId} Find buah by ID
- DELETE** /buah/{buahId} Deletes a buah

**paths**

# Komponen pada paths

Didalam object path memiliki object/komponennya tersendiri diantaranya:

The screenshot shows a Swagger UI interface for a PUT endpoint. Annotations with arrows point to various components: **path object** points to the path `/buah`; **operation path** points to the description `Update data buah yang sudah ada by id`; **parameters** points to the `Parameters` section; **request body** points to the `Request body` section; **required** points to the `required` label; **content** points to the `application/json` dropdown; and **schema** points to the JSON example value.

```
PUT /buah Update data buah yang sudah ada
```

Update data buah yang sudah ada by id

Parameters

No parameters

Request body **required** content application/json

Update data buah yang sudah ada di supermarket

```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ]
}
```

The screenshot shows a Swagger UI interface for a response. Annotations with arrows point to: **response** pointing to the `response` section; **description** pointing to the `Data berhasil di perbaharui` description; **content** pointing to the `application/json` dropdown; and **schema** pointing to the JSON example value.

Responses response

Code	Description	Links
200	Data berhasil di perbaharui	No links
400	Bad request	No links

Media type: application/json

Example Value | Schema

```
{
  "id": 10,
  "name": "salah banget",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

# Keterangan komponen paths

**operation path:** semacam http method karena operasi ini terdiri dari get,put,post,patch,delete dan lain-lain

**summary:** ringkasan dari url path

**path object:** sebuah url yang digunakan untuk mengelompokkan operation path. Contoh: kita membuat path objek dengan nama **/buah** dan **/buah/{buahId}**, setiap kita membuat method yang urlnya mengandung **/buah** maka method tersebut masuk ke dalam grup url **/buah**. Dalam satu grup url **/buah** bisa mempunyai satu atau lebih dari satu method seperti get,post,put.

**description:** deskripsi dari path yang dibuat

**parameters:** bersifat optional tergantung kebutuhan, biasanya digunakan pada method GET dan memiliki komponen/objeknya sendiri. Misal id, nama

**request body:** request data yang diperlukan API, bersifat optional tergantung kebutuhan, biasanya digunakan pada method POST,PUT,PATCH. Memiliki objeknya sendiri seperti:

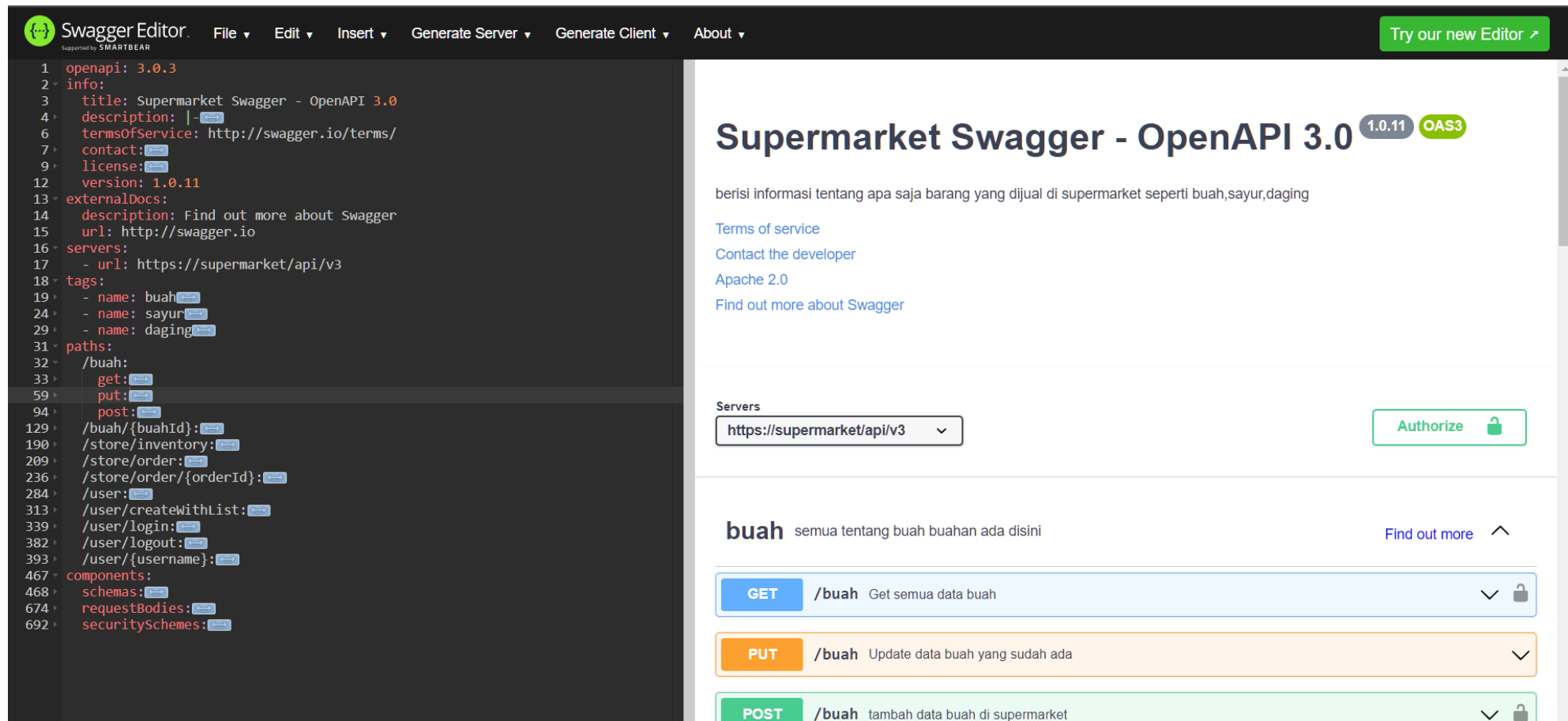
- **Content** : bentuk/konten data pada request body tapi yang paling familiar adalah application/json dan application/xml
- **Required:** untuk memvalidasi request body wajib diisi atau tidak boleh kosong. Required bersifat optional digunakan tergantung kebutuhan
- **Description:** deskripsi dari request body
- **Schema:** skema atau contoh request data yang diperlukan API

**response:** hasil balikan yang dikirim API dari request yang sudah dilakukan. Response memiliki komponennya sendiri diantaranya:

- http status code: kode yang digunakan sebagai informasi dari status request yang dilakukan kepada API. Contohnya 200 jika statusnya berhasil
- Description: deskripsi dari response
- Content: bentuk/konten data pada response tapi yang paling familiar adalah application/json dan application/xml
- Schema: skema atau contoh hasil response data dari API

# Cara membuat dokumentasi Api di Swagger

1. Buka <https://editor.swagger.io/>. tampilan awalnya akan seperti ini. Sudah ada contoh yang disiapkan oleh swagger dalam bentuk yaml



The screenshot displays the Swagger Editor interface. On the left, a dark-themed code editor shows a YAML file for an OpenAPI 3.0 specification. The file includes metadata such as title, description, version (1.0.11), and external documentation. It also defines several API endpoints under the 'paths' section, including a '/buah' endpoint with GET, PUT, and POST methods, and other endpoints like '/store/inventory', '/store/order', and '/user'. On the right, the rendered API documentation page is visible. It features a title 'Supermarket Swagger - OpenAPI 3.0' with version '1.0.11' and 'OAS3' badge. Below the title, there is a brief description and links for 'Terms of service', 'Contact the developer', and 'Apache 2.0'. A 'Servers' section shows the base URL 'https://supermarket/api/v3' with an 'Authorize' button. The 'buah' endpoint is highlighted, with a sub-header 'buah semua tentang buah buahan ada disini' and a 'Find out more' link. Below this, three API methods are listed: GET (Get semua data buah), PUT (Update data buah yang sudah ada), and POST (tambah data buah di supermarket).

2. Kemudian kita bisa langsung membuat dokumentasi API. Disini kita akan membuat dokumentasi API supermarket swagger, memiliki 3 kategori yaitu buah,sayur dan daging. Contohnya seperti gambar dibawah ini

```
Swagger Editor
Supported by SMARTBEAR
File Edit Insert Generate Server Generate Client
1 openapi: 3.0.3
2 info:
3   title: Supermarket Swagger - OpenAPI 3.0
4   description: |
6   termsOfService: http://swagger.io/terms/
7   contact:
9   license:
12  version: 1.0.11
13 externalDocs:
14   description: Find out more about Swagger
15   url: http://swagger.io
16 servers:
17   - url: https://supermarket/api/v3
18 tags:
19   - name: buah
24   - name: sayur
29   - name: daging
31 paths:
32   /buah:
33     get:
59     put:
94     post:
129  /buah/{buahId}:
190  /store/inventory:
209  /store/order:
236  /store/order/{orderId}:
284  /user:
313  /user/createWithList:
339  /user/login:
382  /user/logout:
393  /user/{username}:
467 components:
468   schemas:
674   requestBodies:
692   securitySchemes:
```

---

**openapi:** wajib diisi, isinya adalah versi openapi latest

---

**info:** wajib diisi, sebagai informasi mengenai api yang kita dokumentasikan seperti judul,deskripsi,contact person,dll

---

**externalDocs:** bersifat optional, digunakan jika kita ingin membuat informasi tambahan

---

**servers:** digunakan sebagai tempat URL Api yang akan digunakan

---

**tags:** grouping/pengelompokan api, contoh pada gambar disamping ada 3 grup api yaitu buah, sayur, daging

---

**paths:** untuk membuat url directory api contoh: /buah(artinya url/path akan masuk ke dalam tags buah)

---

**components:** digunakan sebagai tempat untuk membuat skema, request body, security schema yang bisa digunakan oleh objek lain dan objek komponen itu sendiri

### 3. Jika sudah, kita bisa run dan hasilnya akan seperti ini

**Supermarket Swagger - OpenAPI 3.0** 1.0.11 OAS3

berisi informasi tentang apa saja barang yang dijual di supermarket seperti buah,sayur,daging

[Terms of service](#)  
[Contact the developer](#)  
[Apache 2.0](#)  
[Find out more about Swagger](#)

Servers  
https://supermarket/api/v3 Authorize

**buah** semua tentang buah buahan ada disini [Find out more](#) ∨

**sayur** semua tentang sayur sayuran ada disini [Find out more](#) ∨

**daging** semua tentang daging-dagingan ada disini ∨

[Find out more about Swagger](#)

Servers  
https://supermarket/api/v3 Authorize

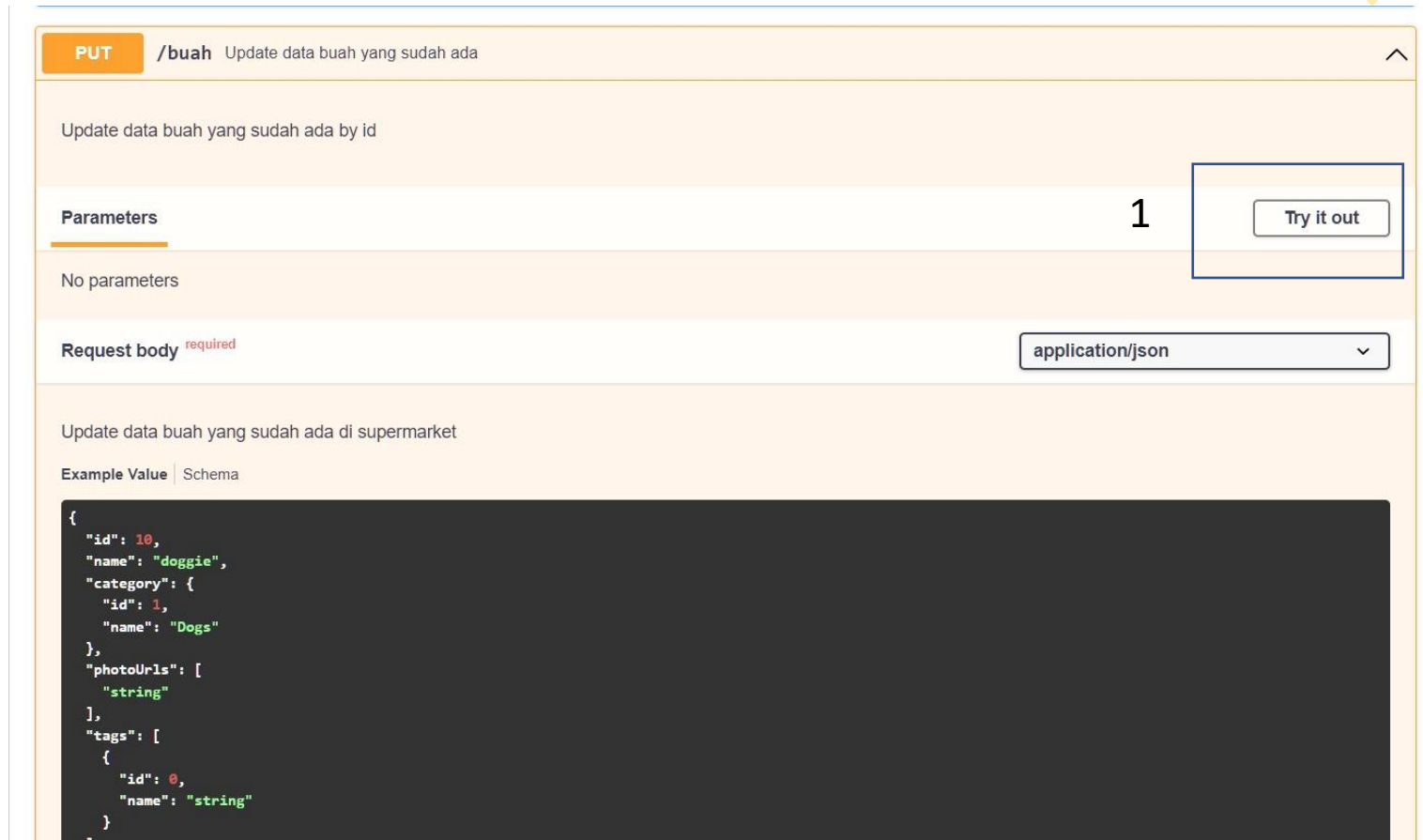
**buah** semua tentang buah buahan ada disini [Find out more](#) ∧

- GET** /buah Get semua data buah ∨ 🔒
- PUT** /buah Update data buah yang sudah ada ∨
- POST** /buah tambah data buah di supermarket ∨ 🔒
- GET** /buah/{buahId} Find buah by ID ∨ 🔒
- DELETE** /buah/{buahId} Deletes a buah ∨ 🔒

# Step by Step Test API dengan Swagger

Disini saya akan menjelaskan cara test api dengan swagger menggunakan API dummy yang sudah disiapkan oleh swagger.

Pertama kita pilih dulu method yang ingin kita test, disini kita akan test method PUT, klik button **Try it Out** untuk menjalankan method yang ingin kita test

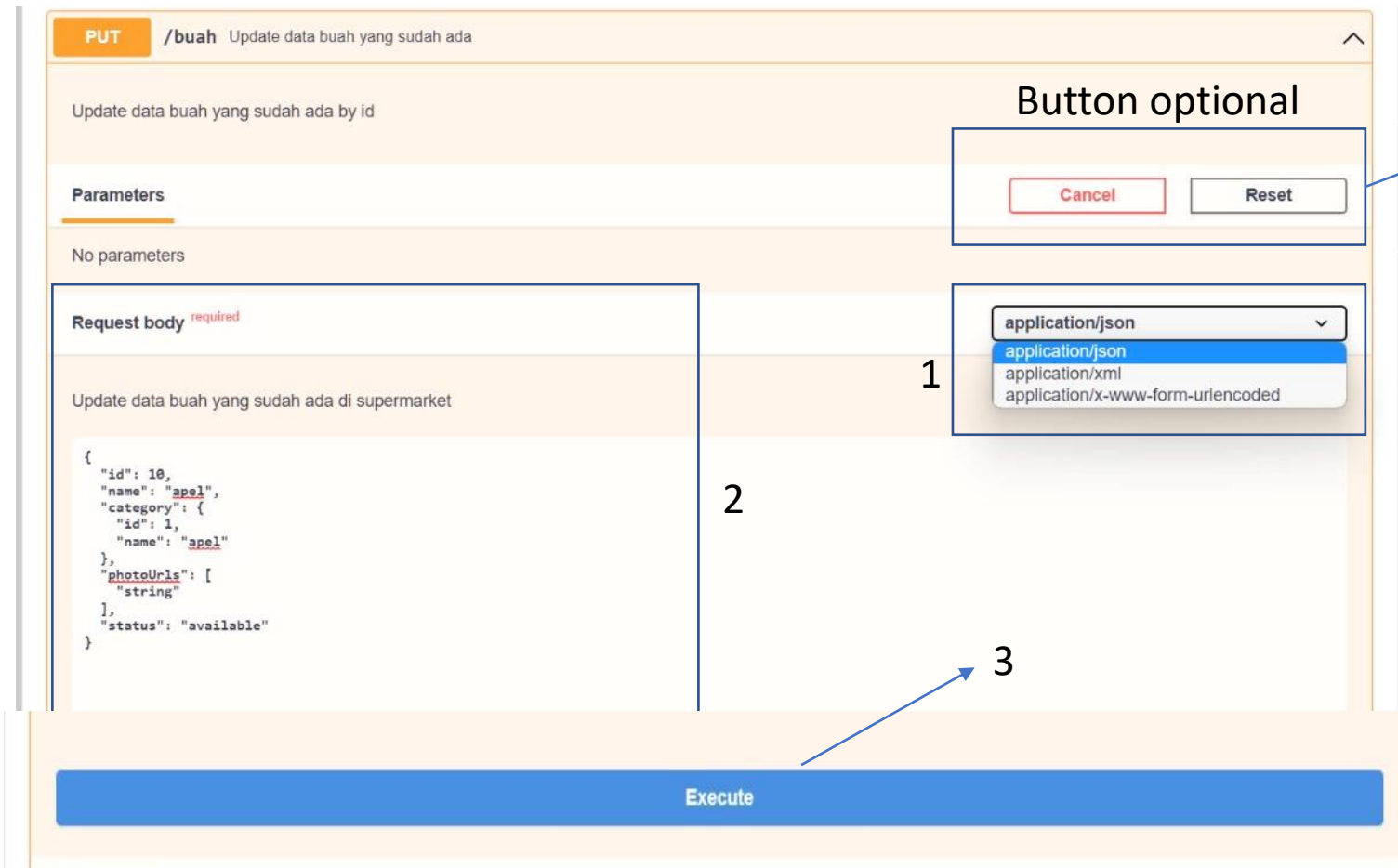


The screenshot shows the Swagger UI interface for a PUT endpoint. The endpoint is labeled 'PUT /buah' with the description 'Update data buah yang sudah ada'. Below the endpoint, there is a 'Parameters' section with a count of '1' and a 'Try it out' button. The 'Request body' section is marked as 'required' and has a dropdown menu set to 'application/json'. Below the request body, there is an 'Example Value' section with a 'Schema' tab. The example value is a JSON object representing a fruit item with a category and tags.

```
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ]
}
```



Setelah itu Langkah pertama kita bisa menentukan content/bentuk data yg mau kita coba apa misal application/json tapi secara default sudah terisi application/json. Pada kolom request body biasanya sudah disediakan requestnya jadi kita tinggal tentukan saja field mana yang mau kita update valuenya setelah itu klik tombol execute



Button cancel digunakan untuk membatalkan test API dan button Reset untuk mengatur ulang request body ke bentuk semula

Setelah di execute swagger akan mengeluarkan response yang terdiri dari curl,request curl, server response, http status code serta schemanya.

Pada contoh dibawah ini response yang keluar adalah 200 yang artinya data berhasil diupdate

Responses

Curl

```
curl -X 'PUT' \
  'https://supermarket/api/v3/buah' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 10,
    "name": "apel",
    "category": {
      "id": 1,
      "name": "apel"
    },
    "photoUrls": [
      "string"
    ],
    "status": "available"
  }'
```

Request URL

https://supermarket/api/v3/buah

Server response

Code	Details
Undocumented	Failed to fetch. Possible Reasons: <ul style="list-style-type: none"><li>• CORS</li><li>• Network Failure</li><li>• URL scheme must be "http" or "https" for CORS request.</li></ul>

200

Data berhasil di perbaharui No links

Media type

application/json

Controls Accept header

Example Value | Schema

```
{
  "id": 10,
  "name": "success",
  "category": {
    "id": 1,
    "name": "apel"
  },
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

400

Bad request No links

404

Data tidak ada No links

500

Internal sever error No links

# Conclussion

---

Student mengetahui apa itu swagger

---

Student mengenal tools/jenis swagger

---

Student mengenal fungsi swagger

---

Student mengenal apa itu Api

---

Student mengenal jenis-jenis Api

---

Student mengenal arsitektur Api

---

Student mengenal komponen swagger

---

Student mengenal komponen pada object paths

---

Student mengetahui step by step pembuatan dokumentasi Api

---

Student mengetahui cara test Api pada swagger

# Source

---

<https://editor.swagger.io/>

---

<https://spec.openapis.org/oas/v3.1.0#response-object>

---

<https://spec.openapis.org/oas/v3.1.0#pathItemObject>

---

<https://spec.openapis.org/oas/v3.1.0#operationObject>

---

<https://spec.openapis.org/oas/v3.1.0#componentsObject>

---

<https://spec.openapis.org/oas/v3.1.0#requestBodyObject>

---

<https://rinaldiudrasuhanda.medium.com/apa-itu-swagger-a9c8157ed725>

---

<https://rinaldiudrasuhanda.medium.com/apa-itu-open-api-1b2cd2adac20>