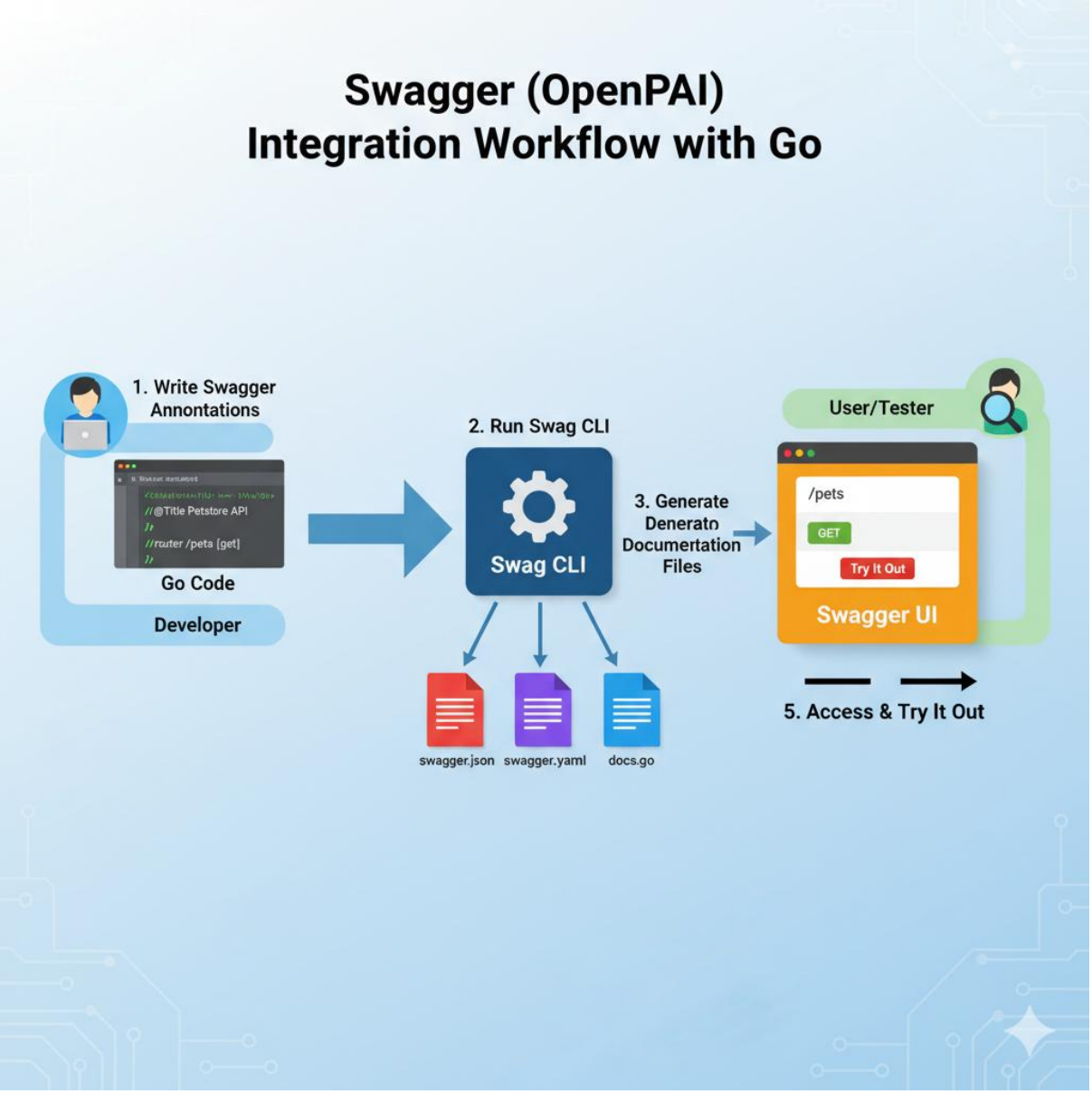
Automatic REST API Documentation with Swagger



Pengenalan Swagger

Swag adalah tool yang mengkonversi anotasi Go menjadi dokumentasi Swagger/OpenAPI secara otomatis. Swagger memungkinkan developer untuk mendokumentasikan API dengan cara yang terstruktur dan interaktif yang mempunyai keuntungan sebagai berikut :

- Dokumentasi Otomatis - Generate dokumentasi dari kode tanpa perlu file terpisah
- Interactive UI - Swagger UI memungkinkan testing API langsung dari browser

- Client Generation - Dapat generate client code dari dokumentasi
- Standardisasi - Menggunakan standar OpenAPI yang diakui industri
- Maintainability - Dokumentasi selalu sinkron dengan kode

Sedangkan Swag CLI adalah command line tool yang membaca anotasi Go dan menghasilkan:

- docs.go - Go file yang berisi dokumentasi dalam format Go
- swagger.json - Dokumentasi dalam format JSON (OpenAPI 2.0)
- swagger.yaml - Dokumentasi dalam format YAML

Alur kerja ini mengintegrasikan dokumentasi API ke dalam proyek Go secara otomatis menggunakan Swag CLI dan menampilkannya melalui Swagger UI. Proses dimulai ketika Developer menulis anotasi Swagger langsung di dalam *file Go* (kode sumber) yang mendefinisikan *endpoint* API, skema data, dan informasi API lainnya. Selanjutnya, *developer* menjalankan Swag CLI. *Tool* ini bertugas untuk mem-*parse* anotasi tersebut. Berdasarkan anotasi yang dianalisis, Swag CLI menghasilkan file dokumentasi standar, yaitu: swagger.json (format OpenAPI standar berbasis JSON), swagger.yaml (format alternatif berbasis YAML), dan docs.go (file *import* khusus untuk Go agar dokumentasi bisa diakses *runtime*). Setelah file-file ini dihasilkan, Swagger UI (biasanya di-*hosting* sebagai bagian dari aplikasi atau secara terpisah) membaca file swagger.json. Ini memungkinkan Swagger UI untuk menampilkan dokumentasi secara visual di *browser* dalam format yang mudah dibaca. Selain itu, Swagger UI juga memberikan fitur "Try It Out" yang memungkinkan User atau *developer* untuk melakukan pengujian *endpoint* API secara langsung dari *interface* dokumentasi.

Anotasi Swagger dalam Go menggunakan format komentar khusus:

```
// @[keyword] [parameter]
```

Contoh anotasi yang umum digunakan:

- @Summary - Ringkasan singkat endpoint (wajib)
- @Description - Penjelasan detail endpoint
- @Tags - Kategori endpoint untuk pengelompokan
- @Accept - Format input yang diterima (json, form, xml, dll)
- @Produce - Format output yang dihasilkan (json, xml, dll)
- @Param - Parameter input (query, path, body, header)

`

- @Success - Response success beserta format data
- @Failure - Response error beserta format data
- @Router - Path dan method HTTP endpoint

Di file main.go, tambahkan anotasi khusus untuk dokumentasi API secara keseluruhan:

```
package main

import (
    "fmt"
    "gofiberapi/config"
    _ "gofiberapi/docs"
    "gofiberapi/handlers"
    "gofiberapi/router"

    "log"

    "github.com/gofiber/fiber/v2"
    "github.com/joho/godotenv"
    fiberSwagger "github.com/swaggo/fiber-swagger"
)

// @title User API
// @version 1.0
// @description API untuk mengelola data user dengan MongoDB menggunakan Clean Architecture
// @host localhost:3000
// @BasePath /api/v1
// @schemes http


func main() {
    err := godotenv.Load()
    if err != nil {
        log.Fatal("Gagal memuat file .env")
    }
      //
```

Anotasi yang digunakan:

- @title - Judul API
- @version - Versi API
- @description - Deskripsi lengkap API
- @host - Host server (tanpa /api/v1)

- @BasePath - Base path API
- @schemes - Protocol yang digunakan (http, https)

Instalasi Swag

Instalasi Swag CLI

```
go install github.com/swaggo/swag/cmd/swag@latest

//verifasi install
swag --version
```

Instalasi Dependencies di Project

```
go get -u github.com/swaggo/gin-swagger
go get -u github.com/swaggo/files
go get github.com/google/uuid
go get -u github.com/gofiber/fiber/v2
go get -u github.com/mongodb/mongo-go-driver
```

Struktur folder yang diperbarui:

```
project/
├── main.go
├── config/
│   └── database.go
├── models/
│   └── user.go
├── repository/
│   └── user_repository.go
├── services/
│   └── user_service.go
├── routes/
│   └── user_routes.go
├── docs/
│   ├── docs.go
│   ├── swagger.json
│   └── swagger.yaml
├── go.mod
├── go.sum
└── .env
```

services/user_service.go

```
// HandleGetAllUsers godoc
// @Summary Dapatkan semua user
// @Description Mengambil daftar semua user dari database
// @Tags Users
// @Accept json
// @Produce json
// @Success 200 {array} models.User
// @Failure 500 {object} models.ErrorResponse
// @Router /users [get]
func (s *userService) HandleGetAllUsers(c *fiber.Ctx) error {
    // implementasi
}

// HandleGetUserByID godoc
// @Summary Dapatkan user berdasarkan ID
// @Description Mengambil data user spesifik berdasarkan ID
// @Tags Users
// @Accept json
// @Produce json
// @Param id path string true "User ID"
// @Success 200 {object} models.User
// @Failure 400 {object} models.ErrorResponse
// @Failure 404 {object} models.ErrorResponse
// @Router /users/{id} [get]
func (s *userService) HandleGetUserByID(c *fiber.Ctx) error {
    // implementasi
}

// HandleCreateUser godoc
// @Summary Buat user baru
// @Description Membuat user baru di database
// @Tags Users
// @Accept json
// @Produce json
// @Param body body models.CreateUserRequest true "User data"
// @Success 201 {object} models.User
// @Failure 400 {object} models.ErrorResponse
// @Failure 500 {object} models.ErrorResponse
// @Router /users [post]
func (s *userService) HandleCreateUser(c *fiber.Ctx) error {
    // implementasi
}

// HandleUpdateUser godoc
// @Summary Update user
// @Description Memperbarui data user berdasarkan ID
```

```go
// @Tags Users
// @Accept json
// @Produce json
// @Param id path string true "User ID"
// @Param body body models.UpdateUserRequest true "User data"
// @Success 200 {object} models.User
// @Failure 400 {object} models.ErrorResponse
// @Failure 404 {object} models.ErrorResponse
// @Router /users/{id} [put]
func (s *userService) HandleUpdateUser(c *fiber.Ctx) error {
    // implementasi
}

// HandleDeleteUser godoc
// @Summary Hapus user
// @Description Menghapus user berdasarkan ID
// @Tags Users
// @Accept json
// @Produce json
// @Param id path string true "User ID"
// @Success 200 {object} models.UserResponse
// @Failure 400 {object} models.ErrorResponse
// @Failure 404 {object} models.ErrorResponse
// @Router /users/{id} [delete]
func (s *userService) HandleDeleteUser(c *fiber.Ctx) error {
    // implementasi
}
```

Anotasi yang digunakan:

- @Summary - Ringkasan singkat endpoint
- @Description - Penjelasan detail endpoint
- @Tags - Kategori endpoint (untuk pengelompokan di Swagger UI)
- @Accept - Format input yang diterima
- @Produce - Format output yang dihasilkan
- @Param - Parameter input (path/query/body)
- @Success - Response sukses dengan status code dan model
- @Failure - Response error dengan status code dan model
- @Router - Path dan method HTTP

```
project/
│
├── main.go
│   │
│   ├── // @title User API                    ANOTASI GLOBAL
│   ├── // @version 1.0
│   ├── // @description ...
│   ├── // @host localhost:3000
│   ├── // @BasePath /api/v1
│   ├── // @schemes http
│   │
│   └── func main() {
│       // import _ "yourmodule/docs"
│       // app.Get("/swagger/*", swagger.HandlerDefault)
│       }
│
└── services/user_service.go
    │
    ├── // HandleGetAllUsers godoc           ANOTASI PER ENDPOINT
    ├── // @Summary Dapatkan semua user
    ├── // @Tags Users
    ├── // @Router /users [get]
    ├── func (s *userService) HandleGetAllUsers(c *fiber.Ctx) error
    │
    ├── // HandleCreateUser godoc            ANOTASI PER ENDPOINT
    ├── // @Summary Buat user baru
    ├── // @Tags Users
    ├── // @Router /users [post]
    ├── func (s *userService) HandleCreateUser(c *fiber.Ctx) error
    │
    ├── // HandleGetUserByID godoc           ANOTASI PER ENDPOINT
    ├── // @Summary Dapatkan user by ID
    ├── // @Tags Users
    ├── // @Router /users/{id} [get]
    ├── func (s *userService) HandleGetUserByID(c *fiber.Ctx) error
    │
    ├── // HandleUpdateUser godoc            ANOTASI PER ENDPOINT
    ├── // @Summary Update user
    ├── // @Tags Users
    ├── // @Router /users/{id} [put]
    ├── func (s *userService) HandleUpdateUser(c *fiber.Ctx) error
    │
    └── // HandleDeleteUser godoc            ANOTASI PER ENDPOINT
        // @Summary Hapus user
        // @Tags Users
        // @Router /users/{id} [delete]
        func (s *userService) HandleDeleteUser(c *fiber.Ctx) error
```

Generate Dokumentasi Swagger

Setiap kali mengubah atau menambah:

- Anotasi Swagger di method handler
- Main function documentation
- Struktur response/request model
- Endpoint baru

Jalankan perintah di root project:

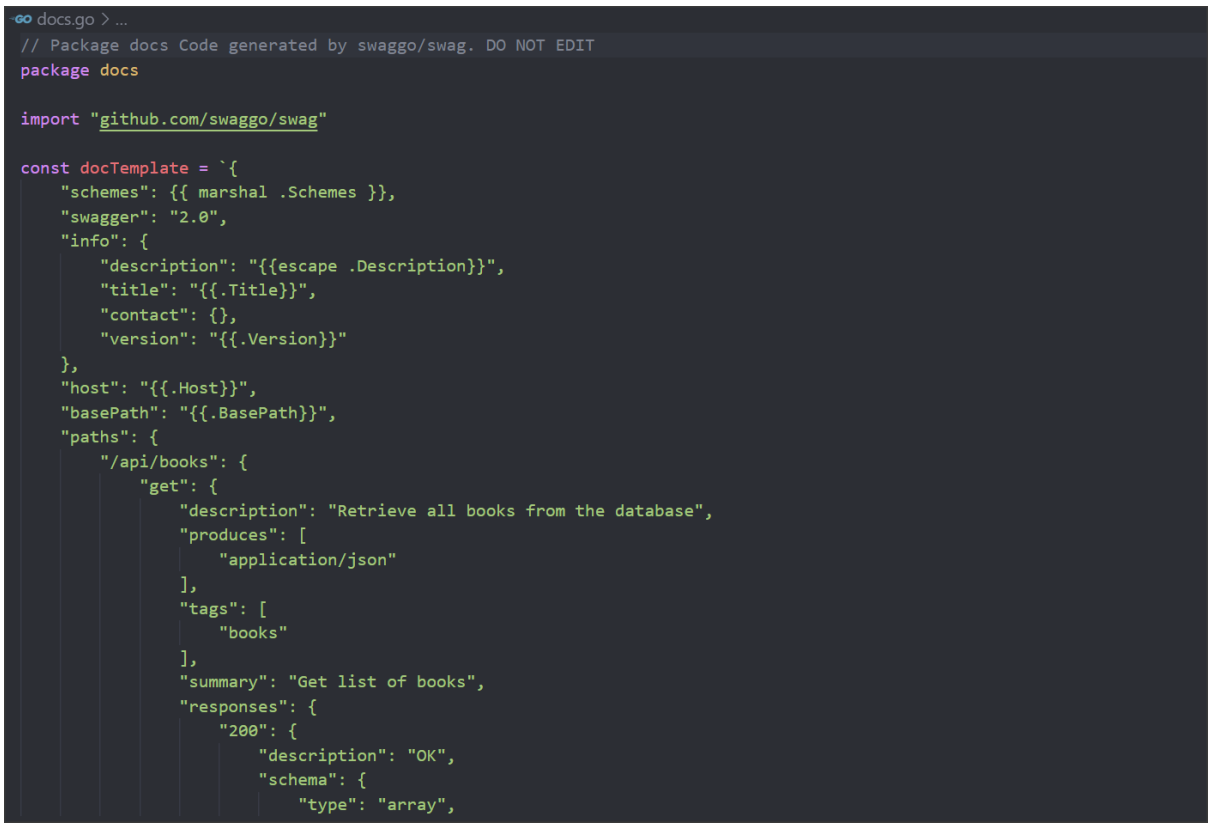
swag init

Output yang Dihasilkan

Perintah ini akan membuat atau update folder docs/ dengan file:

docs.go

adalah jembatan antara anotasi Swagger di kode Go dan Swagger UI. File ini membuat dokumentasi API menjadi bagian integral dari aplikasi Go Anda, sehingga mudah didistribusikan dan diakses tanpa ketergantungan file eksternal.



swagger.json

swagger.json adalah file yang dihasilkan oleh swag (Swag for Go) dan memiliki beberapa fungsi penting:

1. Dokumentasi API Otomatis

File ini berisi spesifikasi lengkap API Anda dalam format OpenAPI (Swagger). Ini mencakup semua endpoint, method HTTP, parameter, request body, response, dan error handling yang telah Anda anotasi dalam kode Go.

2. Standar Industri

swagger.json mengikuti standar OpenAPI 3.0 (atau Swagger 2.0), sehingga dapat dibaca oleh berbagai tools dan platform yang mendukung format ini.

3. Basis untuk Swagger UI

File ini digunakan oleh Swagger UI untuk menghasilkan antarmuka web interaktif. Dengan Swagger UI, developer dapat melihat dan menguji endpoint API langsung dari browser tanpa perlu tools eksternal seperti Postman.

4. Integrasi dengan Tools Lain

swagger.json dapat diintegrasikan dengan berbagai tools seperti:

   - Code generators (menghasilkan client SDK, server stubs)
   - API documentation generators
   - Testing tools
   - API gateway dan mock servers

5. Client Generation

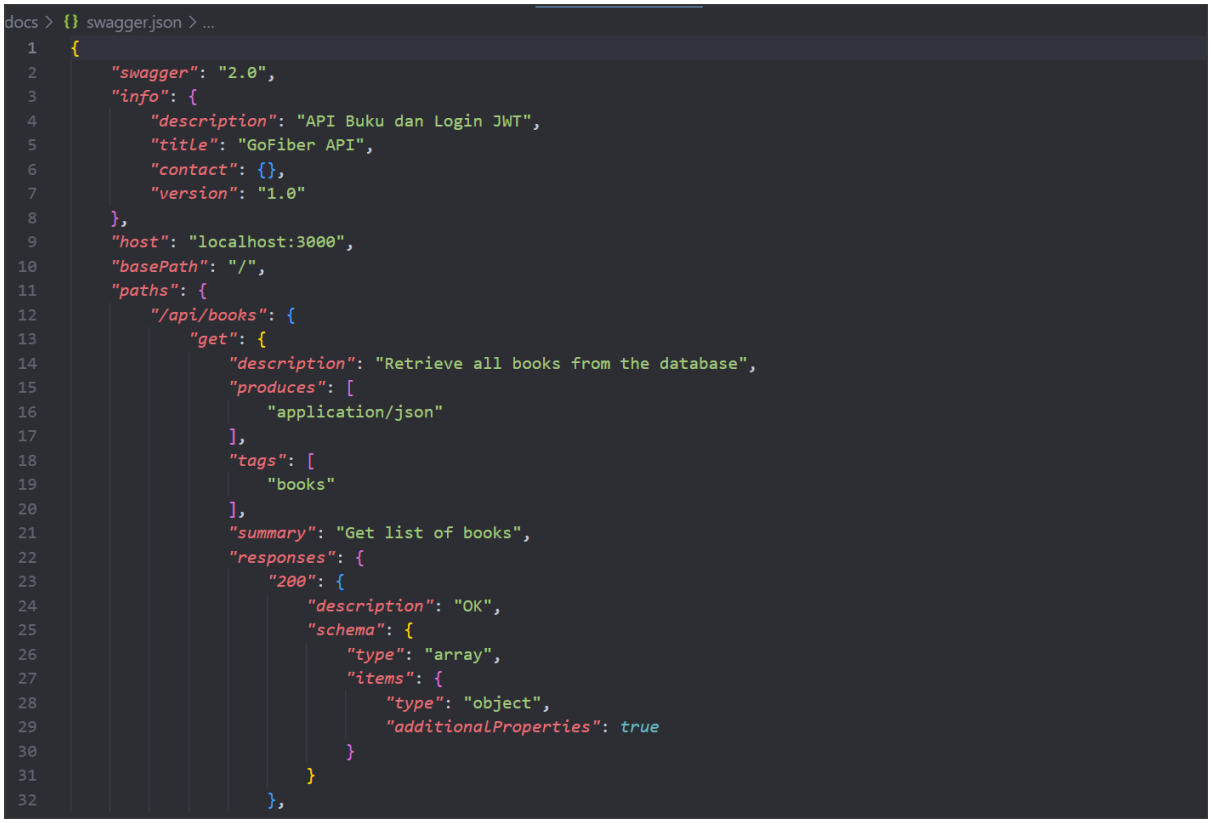
Dari file ini, Anda dapat membuat client library otomatis untuk berbagai bahasa pemrograman (JavaScript, Python, TypeScript, dll) menggunakan tools seperti OpenAPI Generator.

6. Validasi dan Kontrak API

File ini berfungsi sebagai "kontrak" antara backend dan frontend, memastikan kesesuaian antara implementasi API dan konsumsinya.

Secara ringkas, swagger.json adalah jembatan yang membuat API Anda lebih mudah didokumentasikan, diuji, dan diintegrasikan dengan sistem lain.RetryClaude can make mistakes. Please double-check responses.

`



swagger.yaml

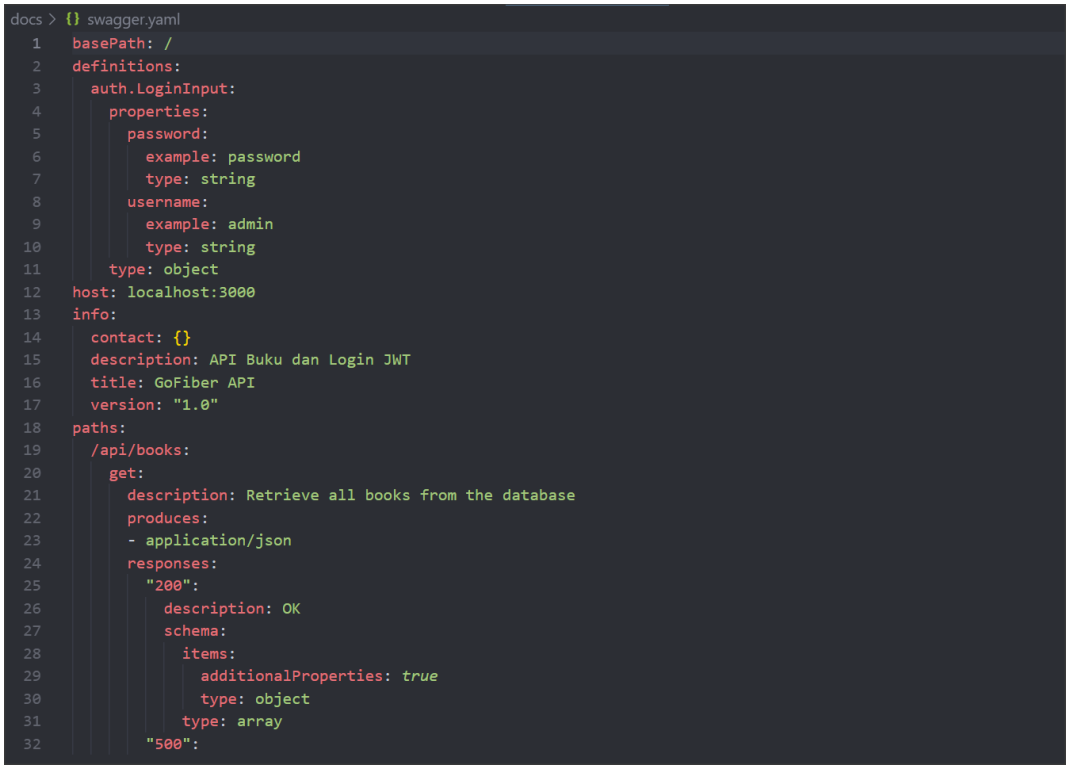
swagger.yaml pada dasarnya memiliki fungsi yang sama persis dengan swagger.json, hanya berbeda dalam format file. Berikut penjelasannya:

1. Format Alternatif untuk OpenAPI

swagger.yaml adalah representasi spesifikasi API yang sama dengan swagger.json, tetapi menggunakan format YAML (YAML Ain't Markup Language) daripada JSON. Keduanya berisi informasi identik tentang API Anda.

2. Lebih Readable dan Mudah Diedit

YAML menggunakan indentasi dan syntax yang lebih sederhana, sehingga swagger.yaml lebih mudah dibaca dan diedit secara manual dibandingkan JSON:



3. Fungsi yang Sama dengan swagger.json

   Dokumentasi API otomatis
   Input untuk Swagger UI
   Basis untuk code generation
   Kontrak API antara backend dan frontend
   Integrasi dengan berbagai tools

4. Pilihan Format

Swag dapat menghasilkan kedua format (swagger.json dan swagger.yaml). Anda bisa memilih format mana yang lebih sesuai dengan preferensi tim Anda. Beberapa organisasi lebih suka YAML karena lebih human-friendly.

Aspek	JSON	YAML
Format	Menggunakan kurung kurawal {} dan tanda kurung siku []	Menggunakan indentasi dan struktur hierarki
Readability	Kurang Readable - Lebih verbose dan sulit dibaca	Sangat Readable - Lebih bersih dan mudah dipahami
Sintaks	Strict - Memerlukan tanda petik ganda untuk key dan value string	Fleksibel - Tidak memerlukan tanda petik pada kebanyakan kasus
Komentar	Tidak Support - Tidak bisa menambahkan komentar	Support - Bisa menambahkan komentar dengan #
Editing Manual	Sulit - Rawan kesalahan sintaks saat edit manual	Mudah - Lebih aman untuk edit manual
Ukuran File	Lebih besar karena banyak karakter khusus	Lebih kecil karena lebih ringkas
Parsing	Cepat - Parser JSON sangat cepat dan efisien	Sedikit lebih lambat dibanding JSON
Kompatibilitas	Universal - Didukung di semua bahasa pemrograman	Didukung di kebanyakan bahasa, tapi tidak seuniversal JSON
Tools Support	Sempurna - Semua tools dan IDE support	Baik - Mayoritas tools support
Penggunaan	API responses, System integration, Production environment	Configuration files, Documentation, Manual editing

Akses swagger UI

Swagger UI adalah antarmuka web interaktif yang memungkinkan Anda melihat dan menguji API secara langsung dari browser. Berikut cara mengaksesnya:

```
//Jalankan Aplikasi
bashgo run main.go
//Buka Swagger UI
//Buka browser dan akses:
http://localhost:3000/swagger/index.html
```