**MEMBUAT DOKUMENTASI API DENGAN FORMAT JSON MENGGUNAKAN VISUAL STUDIO CODE**

***"Jika kamu menginginkan sesuatu yang belum pernah dimiliki dalam hidupmu. Kamu harus melakukan sesuatu yang belum pernah dilakukan sebelumnya" - JD Houston***

1. **Main Objective**

* Mengetahui apa itu json
* Mengetahui fungsi dan struktur dasar json
* Mengetahui tipe data yang bisa digunakan pada json
* Mengetahui cara konsumsi data json
* Mengetahui object pada openapi
* Belajar step by step membuat dokumentasi api

1. **Json (JavaScript Object Notation)**
2. **Pengertian Json**

JSON (JavaScript Object Notation) adalah sebuah format data yang digunakan untuk pertukaran dan penyimpanan data.

JSON merupakan bagian (subset) dari Javascript. JSON bisa dibaca dengan berbagai macam bahasa pemrograman seperti C, C++, C#, Java, Javascript Perl, Python, dan banyak lagi.

1. **Fungsi Json**

* JSON sebagai format untuk bertukar data client dan server atau antar aplikasi. Contoh: RESTful API;
* JSON sebagai tempat menyimpan data, contoh: Database Mongodb;
* JSON digunakan untuk menyimpan konfigurasi project, contoh: file composer.json pada project PHP dan package.json pada Nodejs;
* JSON digunakan untuk menyimpan konfigurasi dan penyimpanan data pada Hugo;
* JSON digunakan untuk menyimpan konfigurasi project pada Nodejs;
* JSON digunakan untuk menyimpan data menifest;
* dan masih banyak lagi.

1. **Struktur Dasar Json**



JSON selalu dimulai dengan tanda kurung kurawal { dan ditutup dengan kurung }.

Lalu di dalam kurung kurawal, berisi data yang format key dan value. Jika terdapat lebih dari satu data, maka dipisah dengan tanda koma dan di data terakhir tidak diberikan koma. Lalu key dan valude dipisah dengan titik dua.

Kita bisa memberikan tipe data apa pun. Bahkan juga bisa kita isi dengan array dan objek.

1. **Tipe data yang didukung Json**

Setidaknya ada enam tipe data yang didukung dan dapat digunakan sebagai value Json yaitu:

* String

String adalah data yang terdiri dari karakter unicode, seperti “Tiara” . contoh: “nama”:”Tiara”

* Object

Object adalah sepasang key dan value, object dibuka dan ditutup dengan kurung kurawal. Jika ada lebih dari satu object, masing-masing dipisahkan oleh koma dan spasi.

“karyawan”: {“nama”:”tiara”, “asal”:”jakarta”}

Pada contoh di atas, semua yang ada di dalam kurung kurawal adalah object yang merupakan value dari “karyawan”.

* Array

Array adalah kumpulan object. Kumpulan ini dibuka dan ditutup dengan kurung siku [].

"karyawan":[

{"nama":"tiara", "asal":"jakarta"},

{"nama":"fitri", "asal":"surabaya"},

{"nama":"ana", "asal":"tangerang"}

]

* Boolean

Boolean adalah jenis data yang hanya berisikan pernyataan benar atau salah. Anda hanya tinggal memasukkan true atau false, seperti pada contoh berikut ini:

“karyawan tetap”:”false”

* Number

Seperti namanya, jenis data ini berupa angka saja. Dengan catatan, angkanya harus merupakan integer atau angka bulat. Artinya, 21,8 atau √2 tidak dapat dijadikan sebagai value . Contoh penggunaannya seperti di bawah ini:

“usia”:“29”

* Null

Jika sebuah key tidak memiliki value, Anda dapat mengetikkan null. Contohnya seperti pada kode berikut:

“golonganDarah”:”null”

1. **Cara konsumsi data Json**

Setiap Bahasa pemrograman memiliki cara masing-masing untuk membuat dan mengakses data json. Contoh pada javascript kita bisa gunakan fungsi JSON.stringify() untuk membuat json dari objek javascript contoh:

// objek javascript

*var* person = {

name: "Dian",

age: 23

}

// string JSON

*var* jsonString = JSON.stringify(person);

// maka akan menghasilkan:

// {"name":"Dian","age":23}

1. **Object OpenApi**

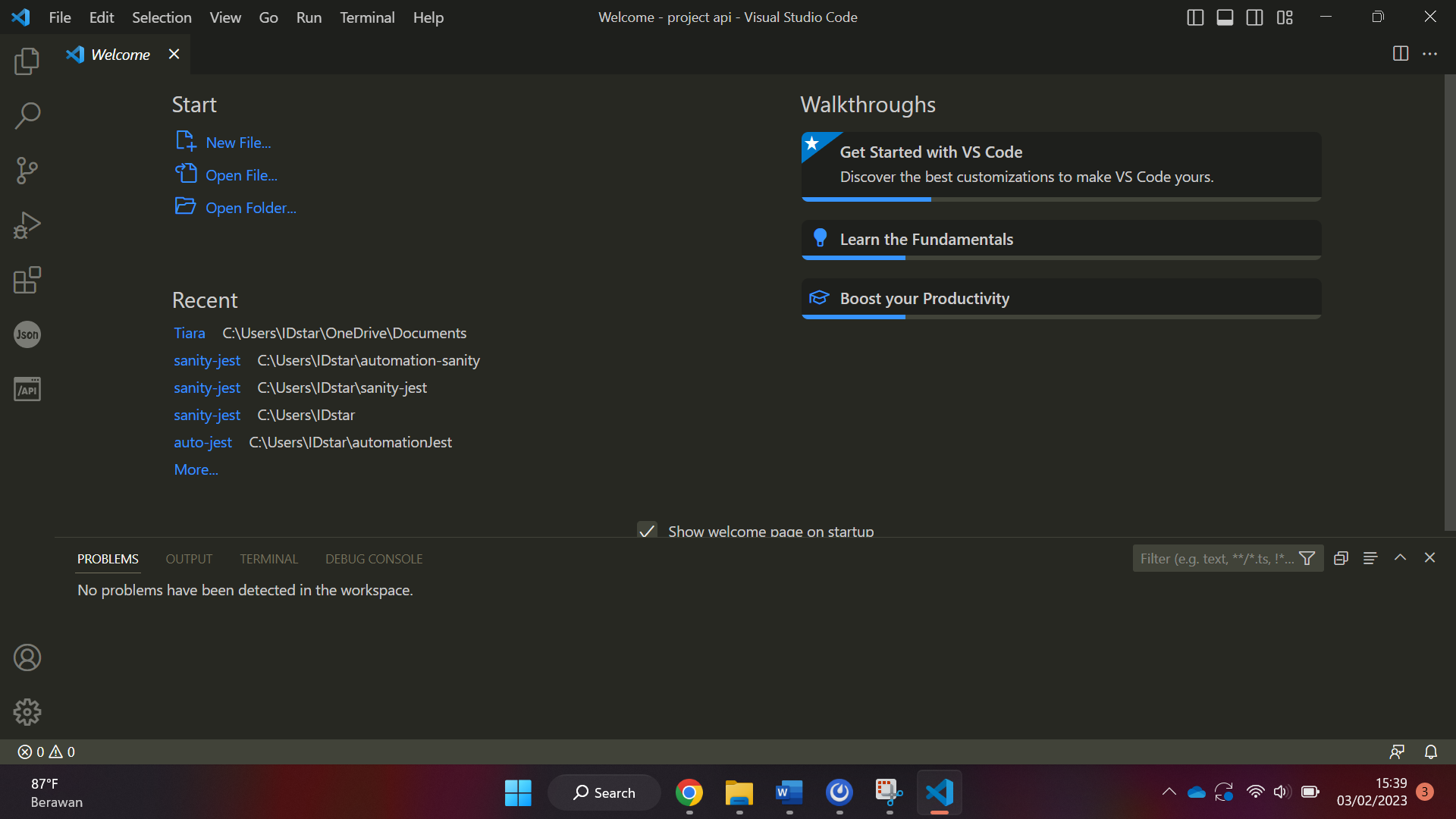
OpenApi digunakan untuk mendokumentasikan Api, terdapat 3 format yaitu json, xml dan yaml. OpenApi memiliki beberapa object atau komponen diantaranya:

* 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
* externalDocs: info tambahan diluar dari object openapi info
* servers: informasi url yang digunakan oleh api
* 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)

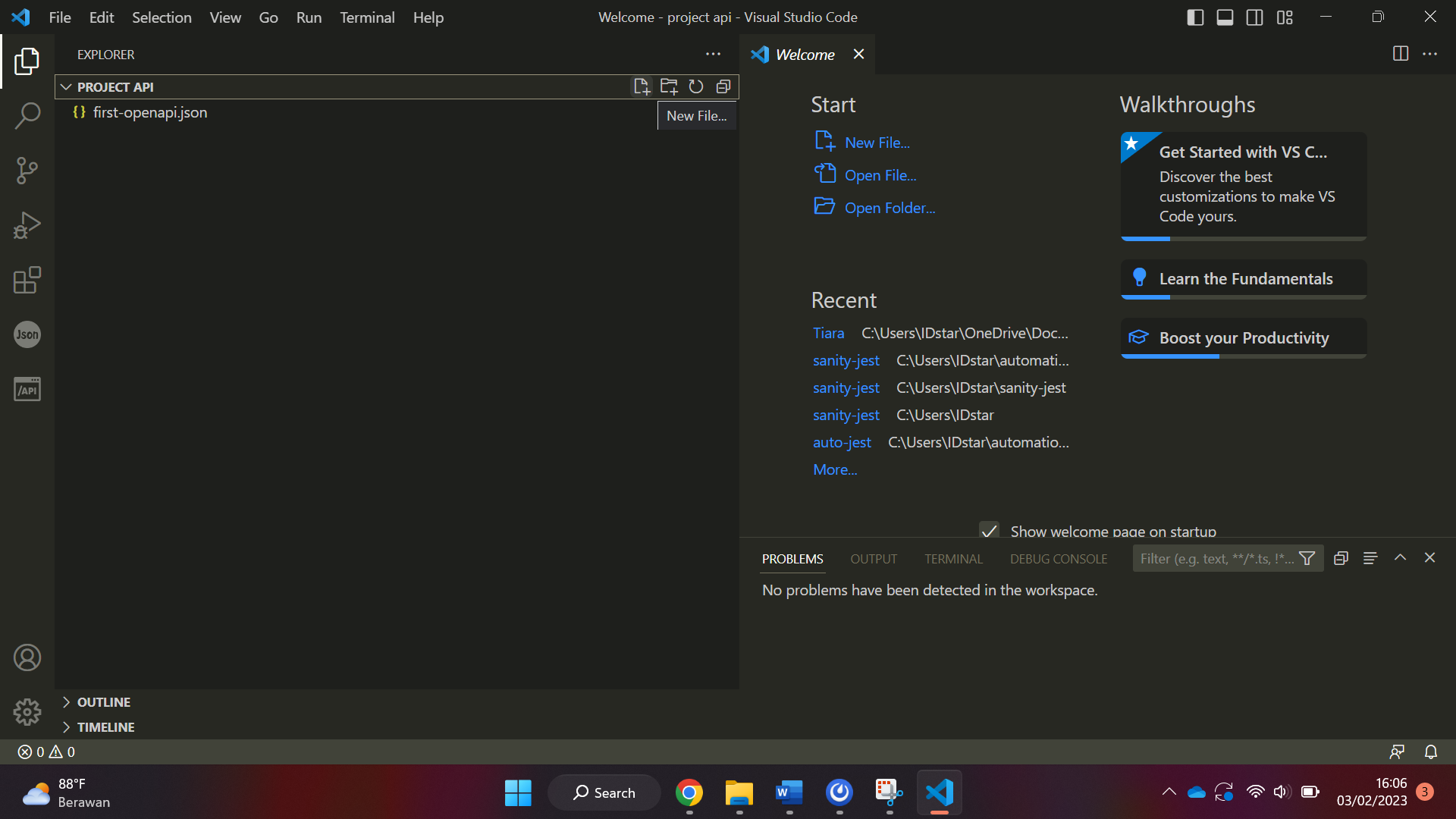
1. **Step by Step membuat dokumentasi API**

Pada study kasus ini kita akan mendokumentasikan API kendaraan. Memiliki 2 jenis kendaraan yaitu motor dan mobil. Berikut adalah Langkah-langkahnya:

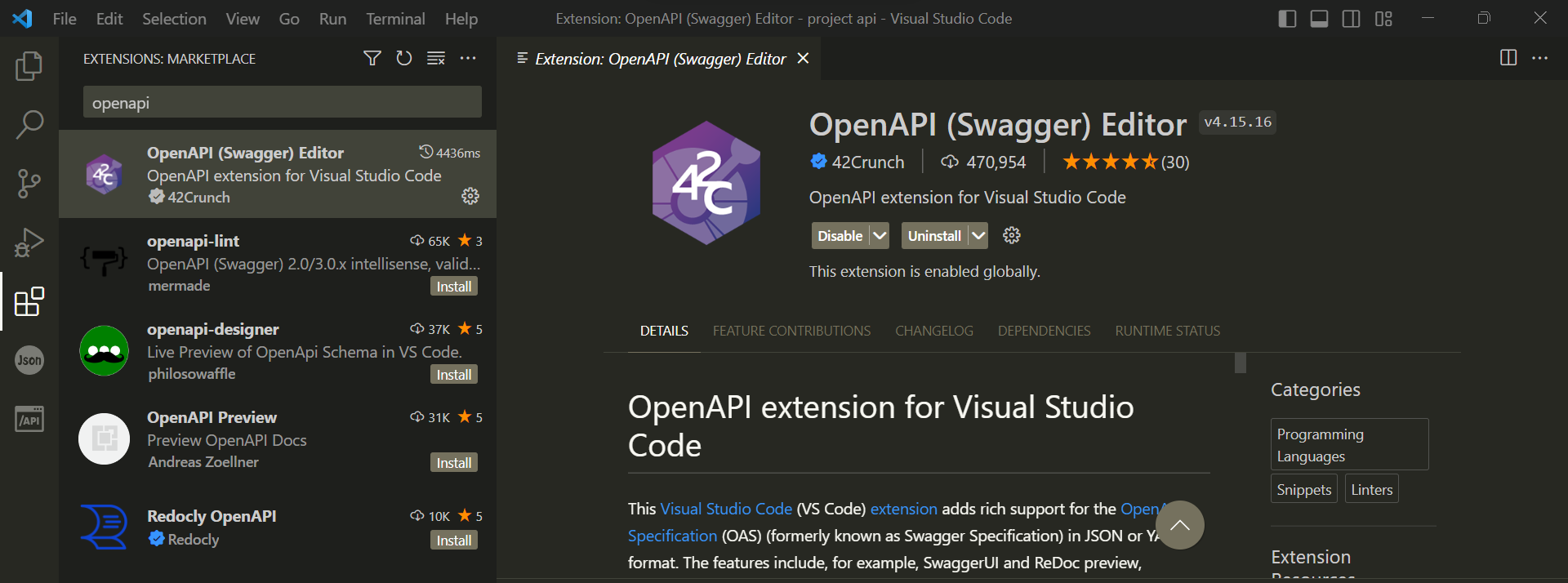
1. Pertama kita harus install visual studio code terlebih dahulu. Untuk install vs code kita bisa download di <https://code.visualstudio.com/download> kemudian pilih sesuai OS laptop/computer kita. Setelah berhasil di download Langkah selanjutnya adalah install vs code.
2. Setelah berhasil terinstall, buka visual studio code dan akan tampil tampilan awal seperti ini



1. Sebelum install exstension yang diperlukan, saya sarankan untuk membuat folder terlebih dahulu untuk menyimpan file json yang akan kita buat, setelah itu pilih folder yang sudah kita buat tadi dan folder tersebut akan tampil pada sisi kiri atas seperti gambar dibawah ini.



1. Setelah itu kita perlu menginstall plugin atau extension untuk membuat openapi. Plugin yang diperlukan adalah “openapi”. Pertama klik tombol berbentuk kotak seperti dadu yang ada dibawah sebelah kiri dan kalau kita hover akan keluar text “extensions” setelah itu ketik “openapi” dan pilih extensions yang ada pada gambar dibawah ini kemudian install. Setelah berhasil install plugin “openapi” otomatis akan tampil pada bar sebelah kiri seperti gambar dibawah ini (nomor 3)



**3**

**2**

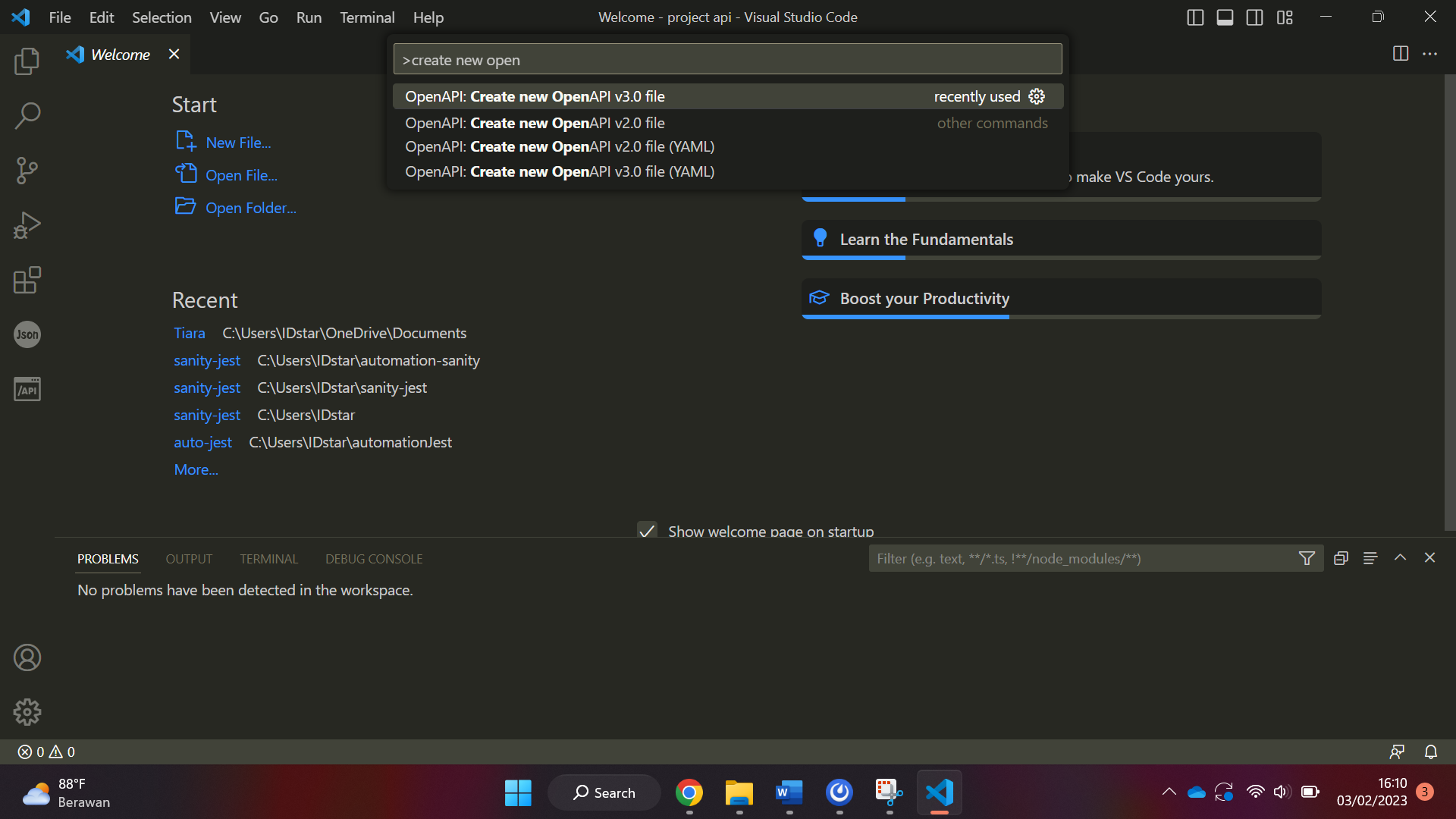
**1**

1. Untuk cek plugin atau extension yang telah kita install kita bisa klik tombol extensions->installed seperti pada gambar dibawah ini

A screenshot of a computer

Description automatically generated with low confidence

1. Untuk membuat file openapi kita bisa tekan tombol Ctrl+Shift+P. setelah popupnya tampil masukkan keyword yang ada unsur “openapi”nya atau bisa langsung ketik “create new open” dan akan muncul list openapi. Ada 2 format yang disediakan yaitu json dan xml, dan ada 2 versi openapi yaitu 2.0 dan 3.0. karena kita akan membuat dokumentasi api dengan json maka disini kita akan menggunakan file openapi dengan format json dengan versi latest yaitu versi 3.0



1. Setelah berhasil dibuat akan tampil file openapi beserta contoh sourcenya yang sudah otomatis tercreate seperti gambar dibawah ini. Saya akan menjabarkan secara detail dari fungsi objek openapi yang akan kita buat untuk dokumentasi api kendaraan ini.

Graphical user interface, text, application

Description automatically generated

1. Pertama adalah “openapi”. Disini kita perlu mengisi versi openapi yang akan kita gunakan dan sifatnya wajib diisi, biasanya diisi dengan versi openapi paling terbaru, karena diawal tadi kita sudah create openapi dengan versi paling terbaru jadi kita tidak perlu mengisinya lagi karena sudah terisi otomatis.

A screenshot of a computer

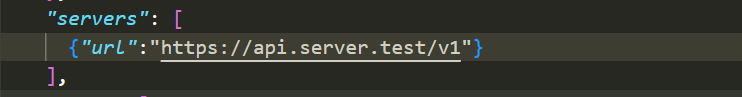
Description automatically generated with low confidence

1. Kedua adalah “info”. Kita wajib mengisi objek info jika tidak akan terjadi error. Objek info berbentuk objek dan memiliki attribute sendiri seperti title, description, contact dan lain-lain. Tapi yang paling sering digunakan adalah title untuk membuat judul dokumentasi, description untuk memberikan keterangan tentang dokumentasi api yang dibuat, contact untuk info contact person yang bisa dihubungi, di object contact mempunyai attribute name,email dan url tapi disini kita hanya menggunakan name dan email saja.

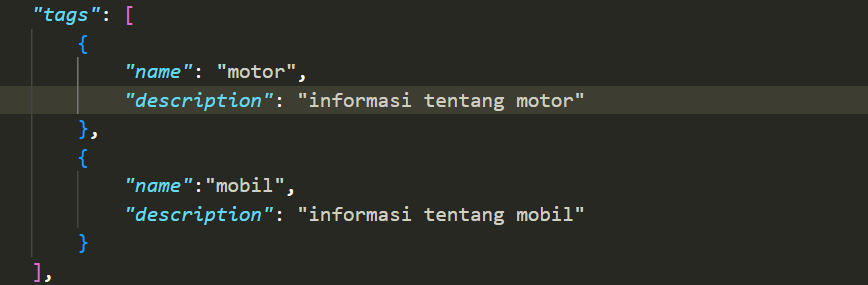
Text

Description automatically generated

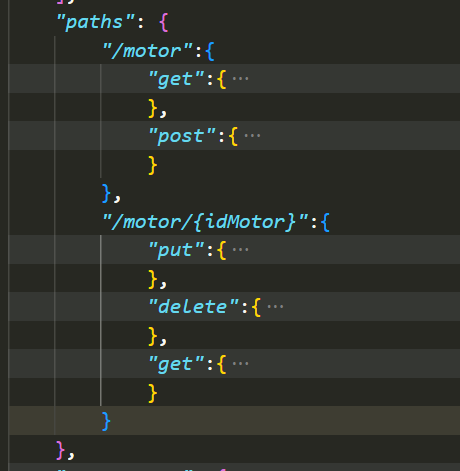
1. Ketiga adalah “servers”, servers ini berbentuk array dan bisa diisi lebih dari satu value dalam bentuk object. Didalam objek tersebut kita bisa mengisi url server yang akan kita gunakan, description server untuk memberikan keterangan tentang server yang kita gunakan.



1. Keempat adalah “tags”, tags ini bentuknya array dan bisa diisi lebih dari satu value dalam bentuk object. Digunakan untuk mengelompokkan api, misal pada gambar dibawah ini kita mempunyai api dengan nama kendaraan dan kendaraan memiliki 2 jenis yaitu motor dan mobil, maka tags berguna untuk memisahkan api yang digunakan untuk mengakses data motor menjadi satu kelompok dan api yang digunakan untuk mengakses data mobil menjadi satu kelompok. Didalam object terdapat beberapa attribute diantaranya name untuk nama grup dan description untuk memberikan deskripsi tentang grup tags tersebut



L. kelima adalah “paths”, sebuah url yang digunakan untuk pengelompokkan method yang akan kita buat. Contoh: kita membuat path dengan nama **/motor** dan **/motor/{motorId}**, setiap kita membuat method yang urlnya mengandung **/motor** maka method tersebut masuk ke dalam grup url **/motor.** Dalam satu grup url **/motor** bisa mempunyai satu atau lebih dari satu method seperti get,post,put. Dibawah ini adalah gambaran struktur umum pada objek paths. Saya akan menjabarkan secara detail apa saja attribute yang ada pada paths di gambar-gambar selanjutnya



m. dibawah ini adalah struktur detail pada objek path dan attribute yang paling sering digunakan pada objek path diantaranya:

- “put” adalah http method, fungsinya untuk mengupdate data

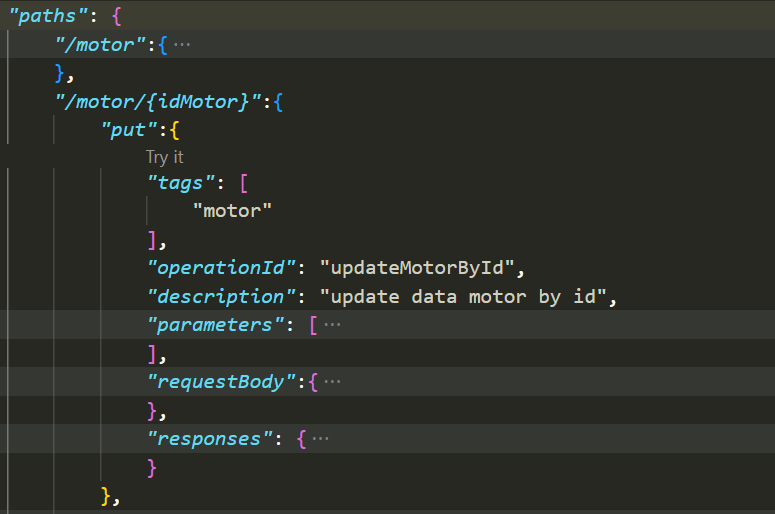
- “tags” digunakan untuk mengelompokkan url /motor/{idMotor} dan method put pada kelompok motor

-“operationId” bentuknya harus unique, digunakan sebagai value pembeda antar method

-“parameters” bentuknya array, digunakan jika api memerlukan parameter dan bisa diisi lebih dari satu value

-“requestBody” sebagai tempat untuk menampilkan dan memberikan gambaran request yang diperlukan oleh API dan digunakan jika api memerlukan requestBody, biasanya formatnya json

-“response” digunakan untuk memberikan hasil keluaran yang diberikan oleh API



n. Parameters bentuknya array, digunakan jika api memerlukan parameter dan bisa diisi lebih dari satu value. Parameters memiliki attribute:

-name sebagai nama dari parameter,name yang ada pada attribute parameters harus sama dengan url yang ada pada path, misal namenya idMotor berarti urlnya harus /motor/{idMotor}, kalau namenya berubah parameter pada url pun harus berubah kalau tidak akan terjadi error.

-in digunakan untuk menentukan parameter masuk kategori mana dan ditempatkan dimana, kategori yang paling sering digunakan yaitu path dan query. Kalau kita pilih path urlnya akan menjadi /motor/{idMotor} dan kalau query urlnya akan menjadi /motor?idMotor=1

-required digunakan untuk membuat parameter menjadi wajib diisi, attribute required harus digunakan jika kita memilih in path

-description digunakan untuk memberikan deskripsi tentang parameter

-schema digunakan untuk memberikan gambaran tentang value parameter tersebut, terdiri dari tipe datanya apa formatnya apa dan lain-lain



Hasil parameter

Graphical user interface, application

Description automatically generated

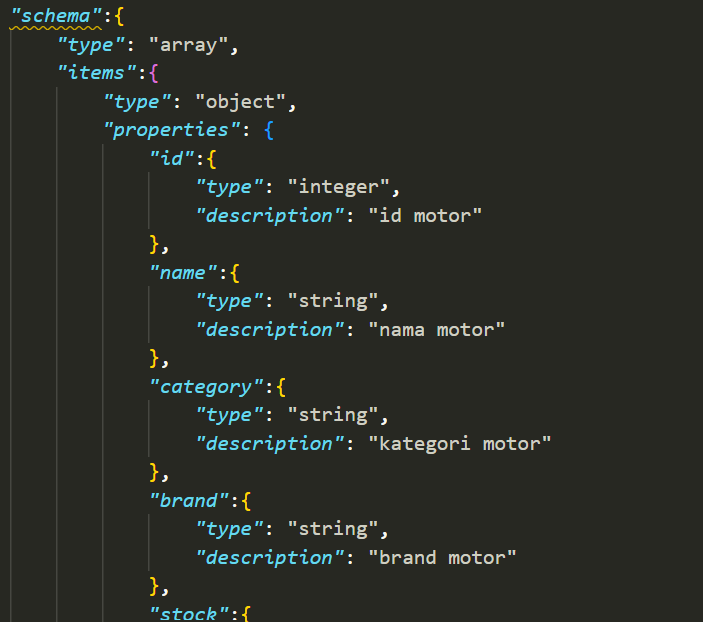
o. requestBody sebagai tempat untuk menampilkan dan memberikan gambaran request yang diperlukan oleh API dan digunakan jika api memerlukan requestBody, biasanya formatnya json. requestBody memiliki attribute misal content digunakan untuk membuat konten dari request body tersebut, ada 2 tipe pada content yang paling sering digunakan yaitu application/json dan application/xml, description untuk menjelaskan deskripsi pada content dan content memiliki attribute tersendiri diantaranya:

- example: digunakan untuk memberikan gambaran request yang diperlukan api seperti apa contoh pada gambar dibawah ini didalam example ada id,name,category dan lain-lain

-schema: digunakan untuk memberikan gambaran pada api field apa saja yang dibutuhkan, tipe datanya apa dan lain-lain. Pada gambar dibawah ini schema tersimpan didalam object components dan cara aksesnya adalah dengan “$ref” ke dalam objek yang menyimpan schema tadi



Contoh source “schema”



Contoh request body example value

Rectangle

Description automatically generated

Contoh request body schema

Graphical user interface, application

Description automatically generated

p. responses digunakan untuk memberikan hasil keluaran yang diberikan oleh API. Untuk attribute responses kurang lebih sama dengan attribute pada requestBody yang membedakan adalah pada response kita harus menyertakan http status code sebelum membuat content. Attribute response diantaranya:

-http status response digunakan untuk memberikan status respon dari API misal 200 artinya berhasil

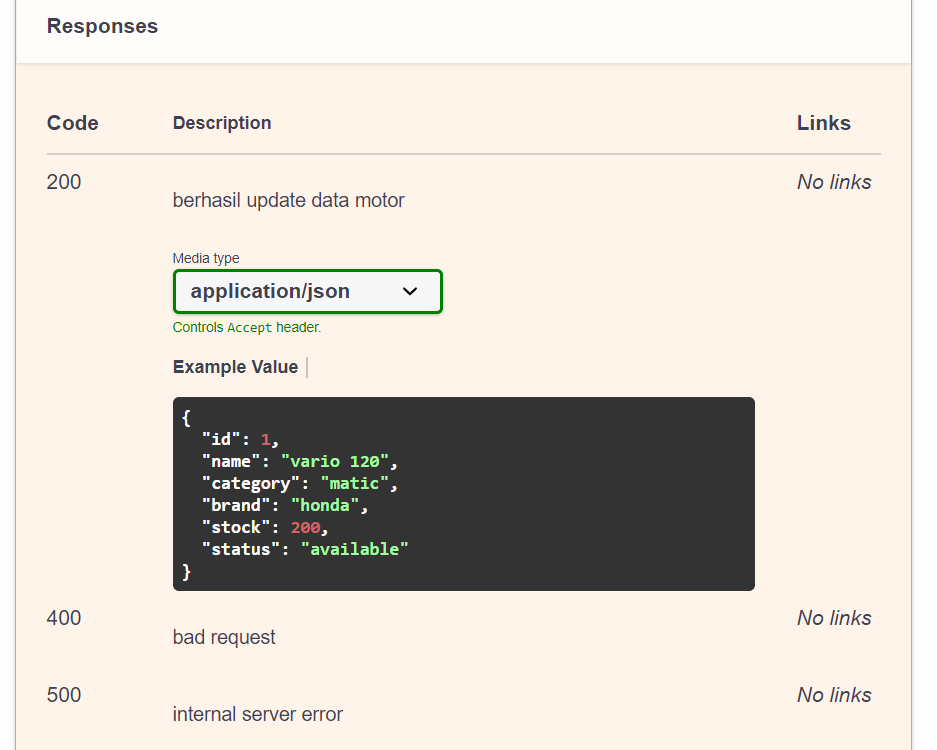
-description digunakan untuk memberikan deskripsi dari respon

- content digunakan untuk membuat konten dari response tersebut, ada 2 tipe pada content yang paling sering digunakan yaitu application/json dan application/xml. Didalam content memiliki attribute yang sama seperti pada requestBody diantaranya schema,example dan lain-lain

Text

Description automatically generated

Contoh responses



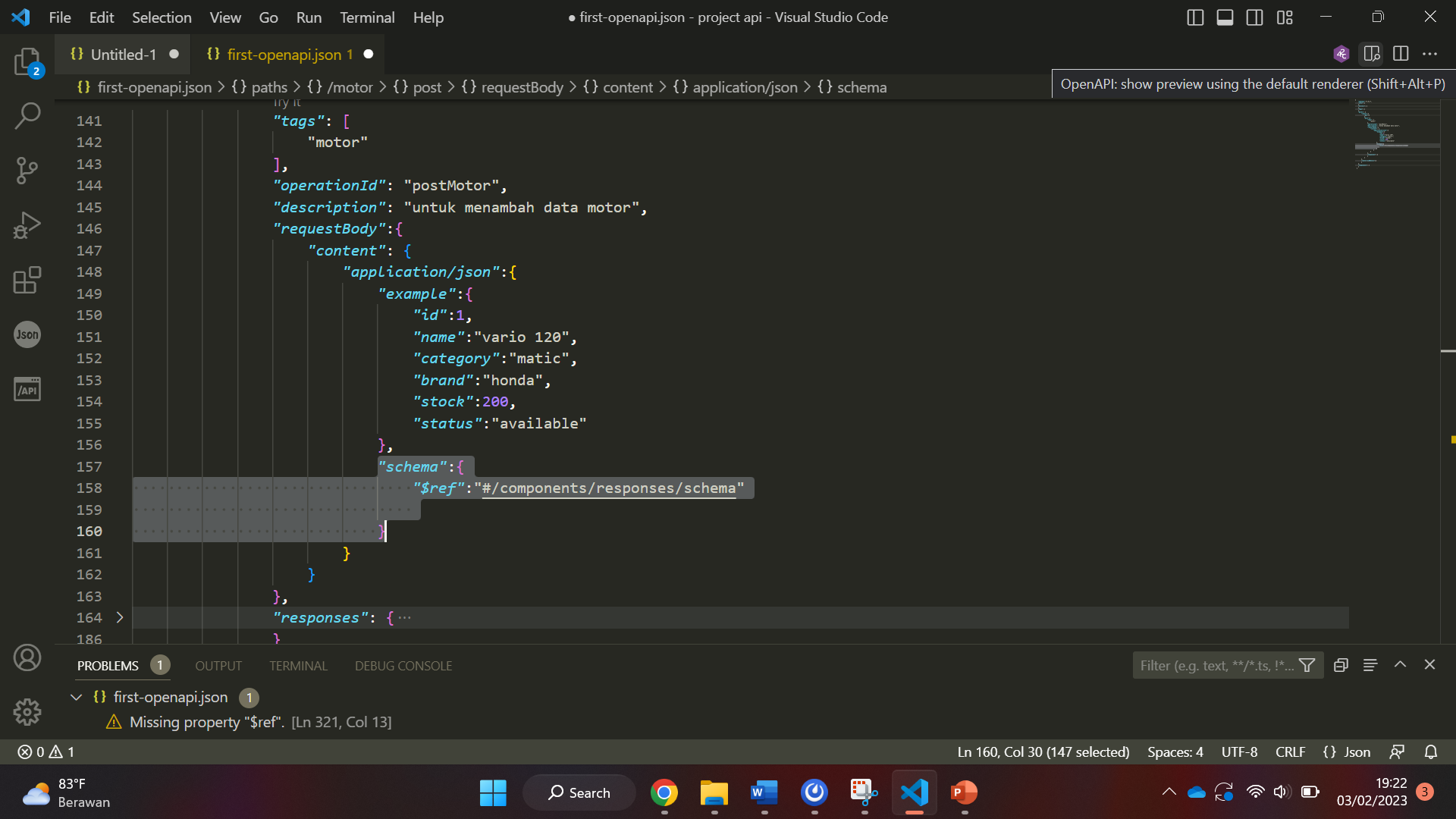
q. Kelima adalah components, digunakan sebagai variable atau tempat menyimpan source yang sama sehingga kita tidak perlu membuat ulang source yang sama cukup satu source dan kita simpan didalam components. Components memiliki beberapa attribute seperti attribute responses untuk menyimpan value response. Kita bisa mengakses component dengan cara mengaksesnya menggunakan “$ref” pada attribute yg ingin mengakses data components tersebut, contoh "schema":{

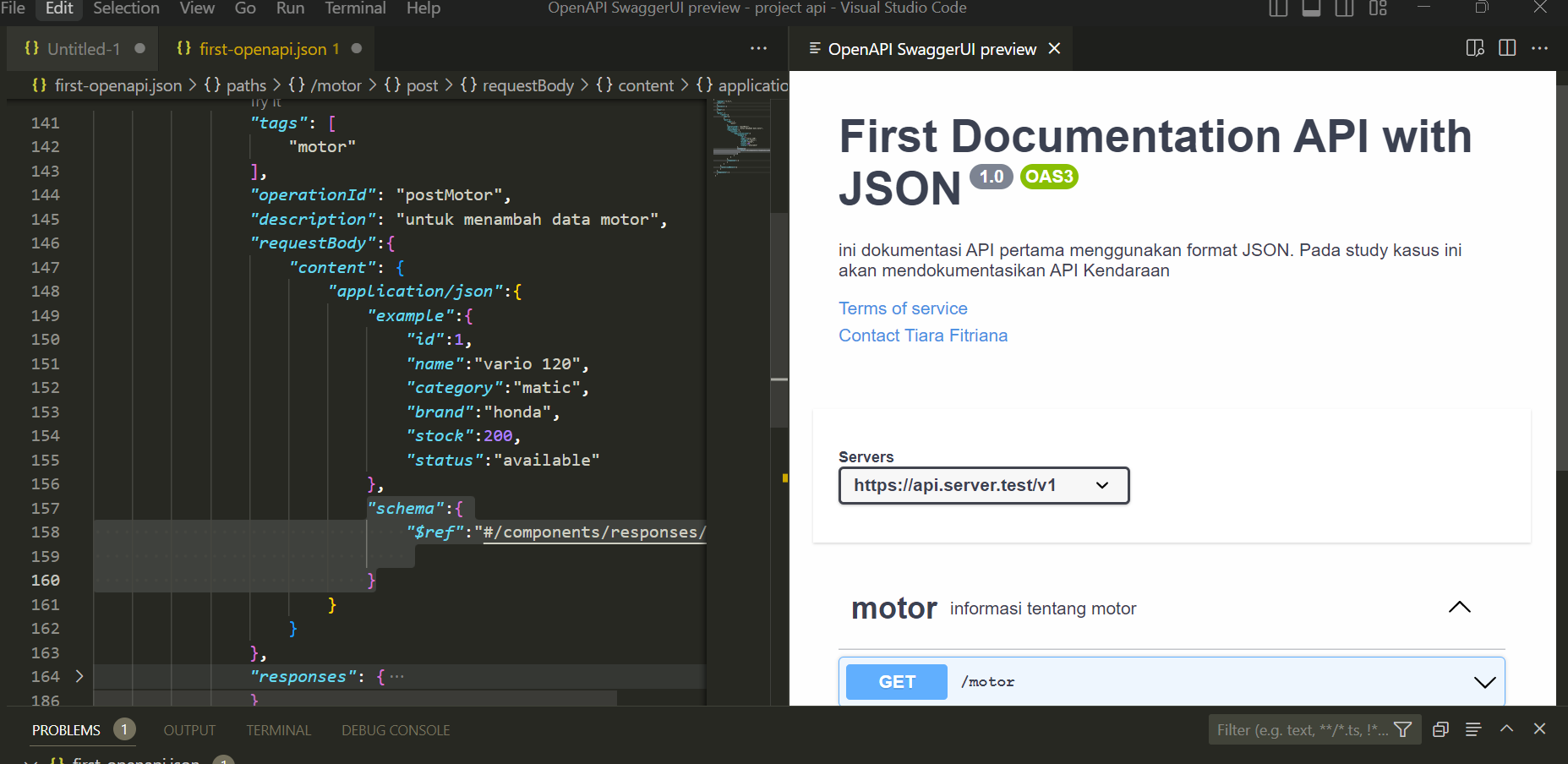
"$ref":"#/components/responses/schema" }

Artinya adalah kita mengakses object components->variable response->variable schema yang isinya schema dari response



r. setelah itu untuk melihat preview dari dokumentasi api dalam bentuk visual kita bisa menekan tombol Shift+alt+P atau klik icon pada gambar dibawah ini dan akan tampil seperti ini. ini adalah hasil dari dokumentasi api yang sudah kita buat tadi





1. **Conclussion**

* Student mengetahui apa itu json
* Student mengetahui fungsi dan struktur dasar json
* Student mengetahui tipe data yang bisa digunakan pada json
* Student mengetahui cara konsumsi data json
* Student mengetahui object pada openapi
* Student belajar step by step membuat dokumentasi api

1. **Source**

[**https://spec.openapis.org/oas/v3.1.0**](https://spec.openapis.org/oas/v3.1.0)

[**https://www.niagahoster.co.id/blog/json-adalah/**](https://www.niagahoster.co.id/blog/json-adalah/)

[**https://code.visualstudio.com/download**](https://code.visualstudio.com/download)