**Implementasi Swagger UI pada Azure Function v3 dengan Menggunakan Swashbuckle**

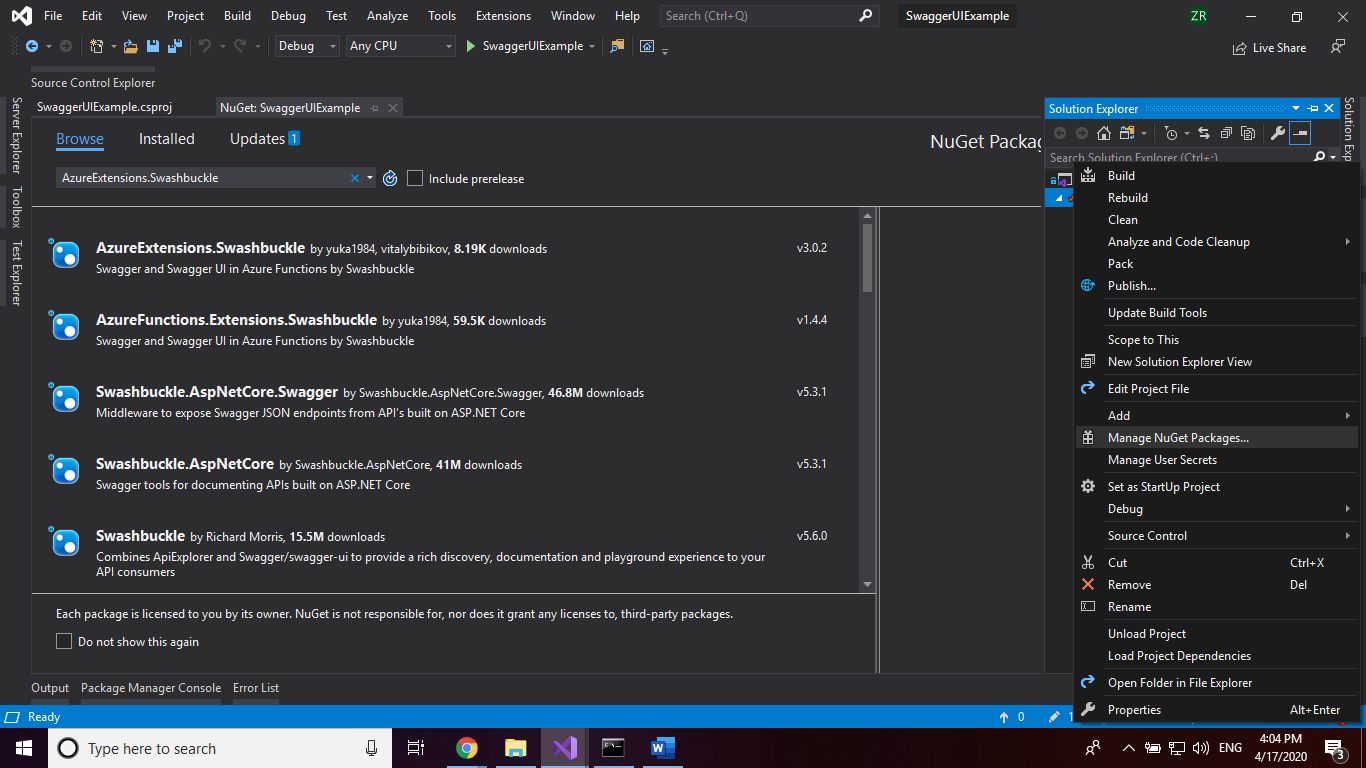
# Requirements:

* Azure Function V3
* .NET Core v3.1
* NuGet Package: AzureExtensions.Swashbuckle

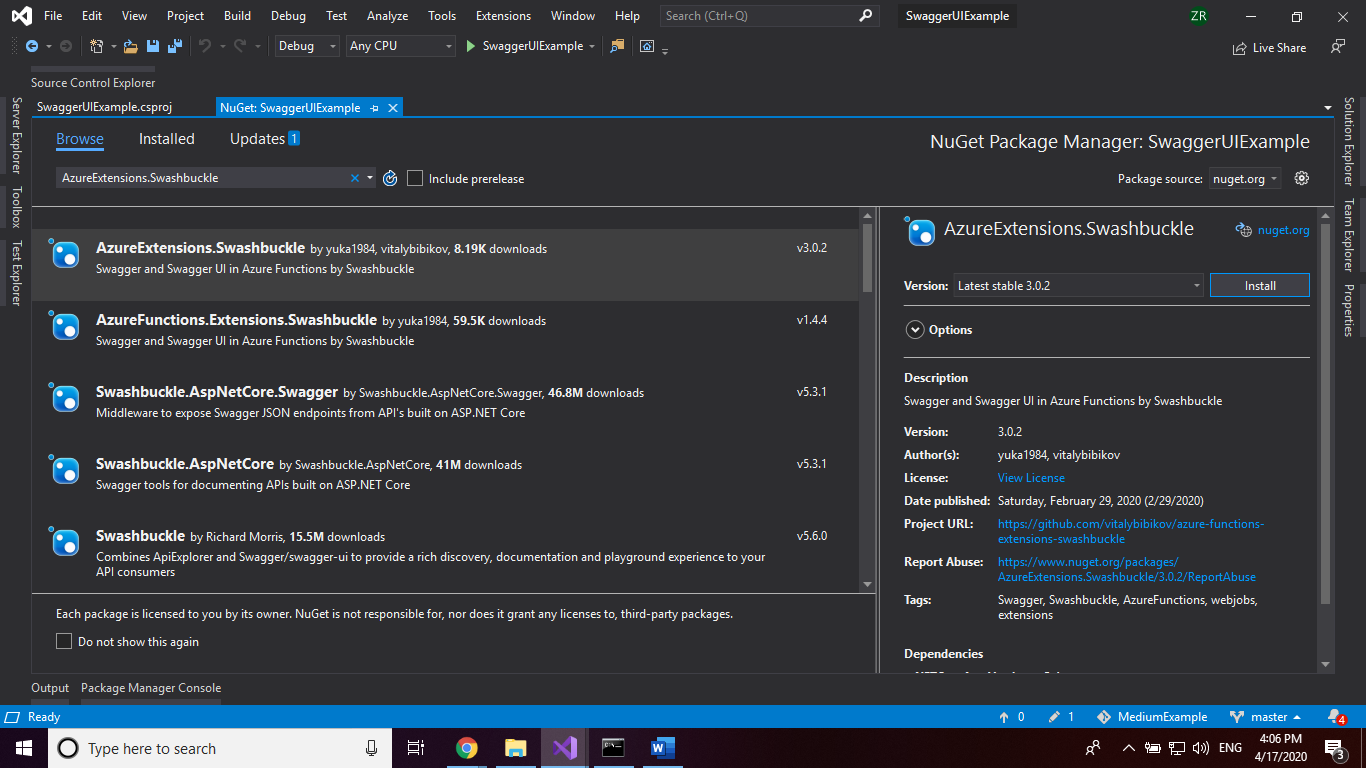
# Steps:

## Instalasi NuGet Package: AzureExtensions.Swashbuckle

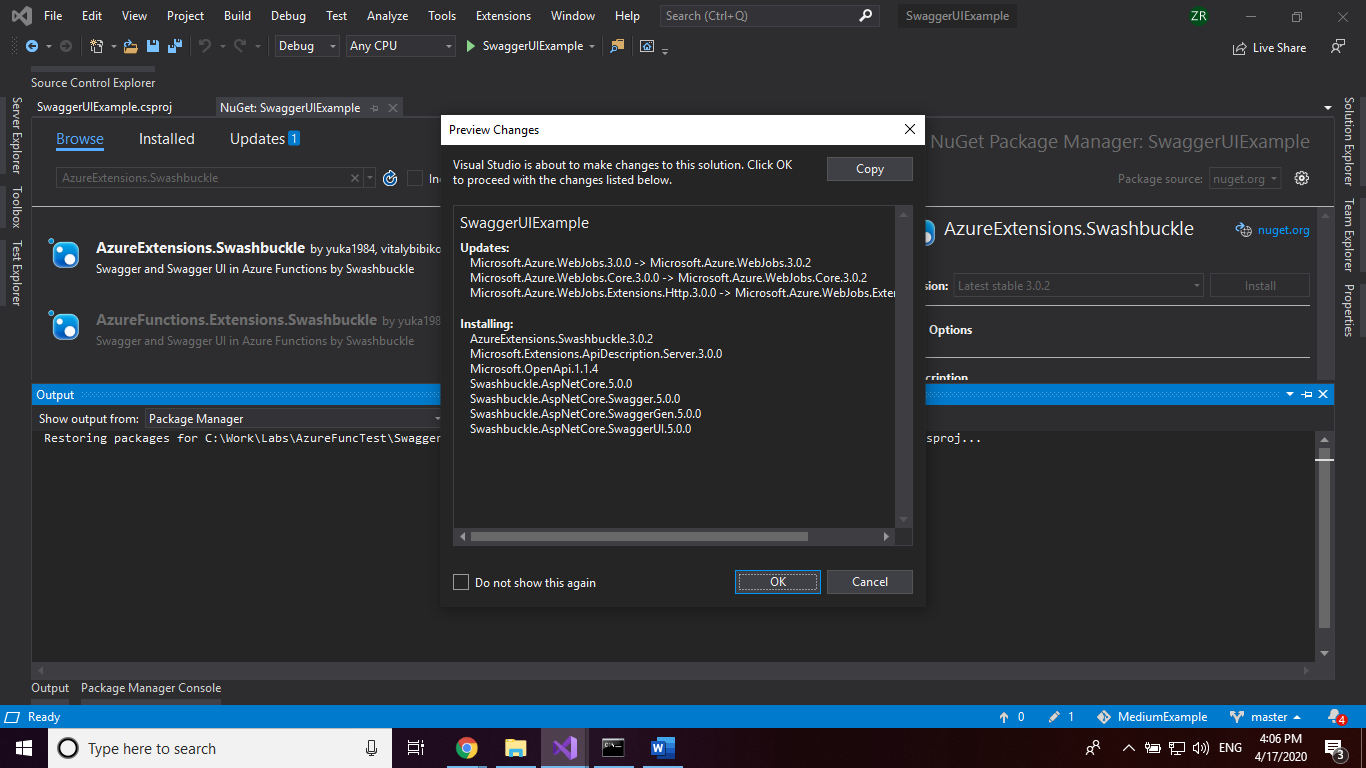
1. Pada project yang akan menggunakan Swagger UI, klik kanan dan pilih “Manage NuGet Pakages..”



1. Selanjutnya, pada kotak pencarian, cari “AzureExtensions.Swashbuckle”, pilih lalu klik install

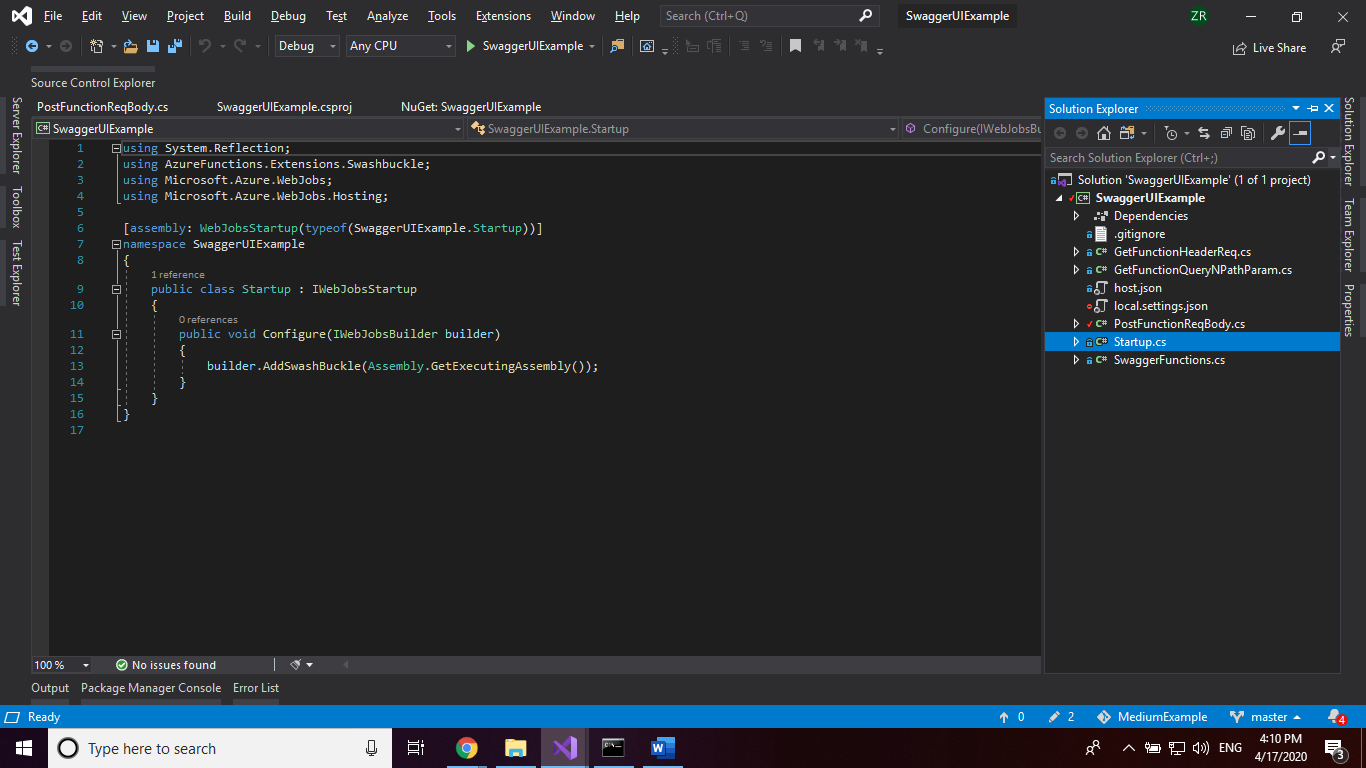


1. Jika ada pemberitahuan “Preview Changes”, klik “OK” dan instalasi package dimulai

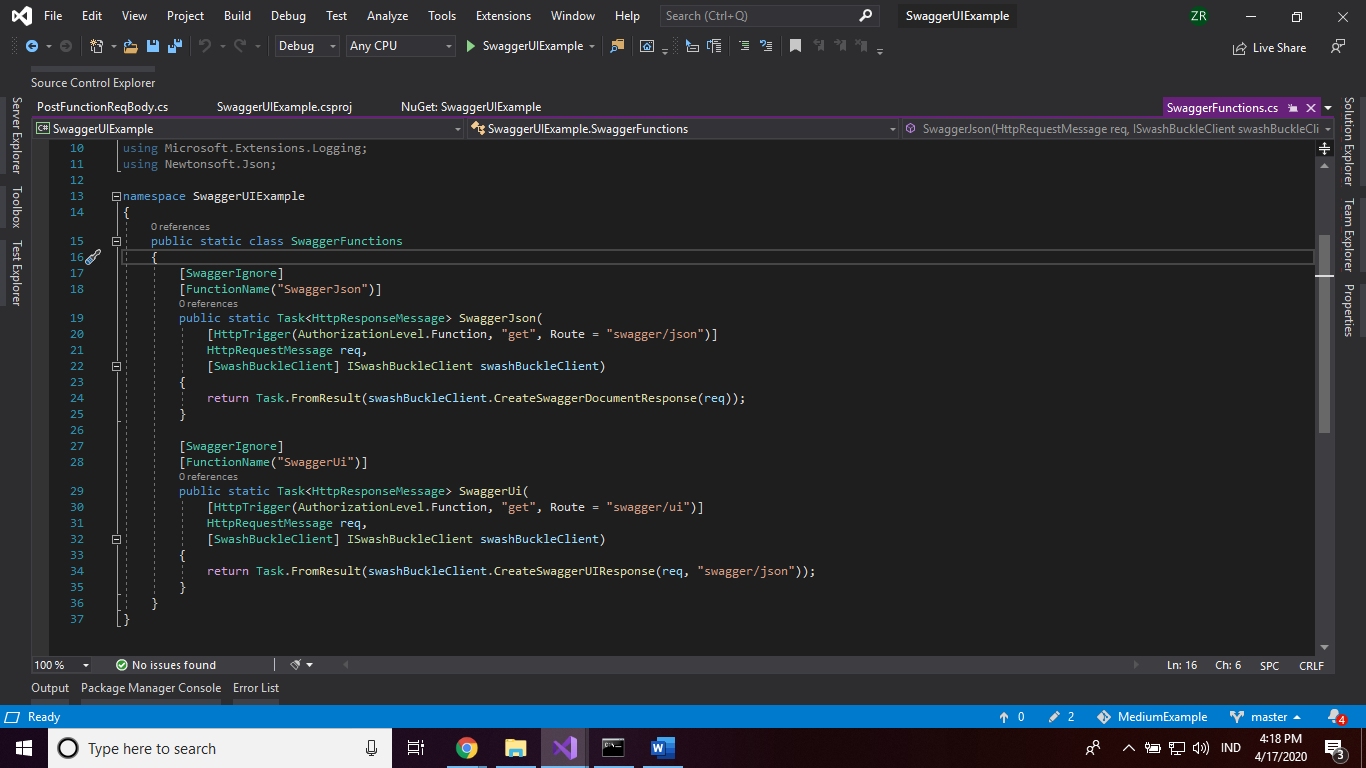


## Proses Persiapan Penggunakan Package untuk Pembuatan SwaggerUI

1. Pada proyek yang akan menggunakan Swagger UI, tambahkan sebuah file configurasi Startup.cs, seperti gambar di bawah. Fungsinya adalah untuk menambahkan package AzureExtension.Swashbuckle pada konfigurasi aplikasi sehingga anotasi yang digunakan dapat dibaca oleh package



1. Setelah itu tambahkan sebuah Azure Function baru yang bernama SwaggerFunctions.cs. Isi dari function ini digunakan untuk membuat file json dari openapi dan juga membuat visualisasi (GUI) dari openapi tersebut. Isi dari SwaggerFunction.cs adalah seperti berikut:



Keterangan:

* Function SwaggerJson berfungsi untuk membuat file openapi json. Untuk AuthorizationLevel bisa anonymous bisa function.
* Function SwaggerUI berfungsi untuk membuat visualisasi dari file openapi json yang telah dibuat oleh function SwaggerJson. Untuk AuthorizationLevel bisa anonymous bisa function. Jika pada SwaggerJson memiliki AuthorizationLevel function, maka pastikan menambahkan parameter “?code=<apiKey>” pada function call CreateSwaggeUIResponse().

Contoh: “.CreateSwaggerUIResponse(req, “swagger/json?code=<apiKey>”)

Untuk mendapatkan api key dari function SwaggerJson maka anda perlu mempublish terlebih dahulu, lalu ambil api keynya, kemudian edit function “.CreateSwaggerUIResponse” lalu lakukan publish ulang.

## Membuat Dokumentasi Request Body dan Response dari API

Setelah kita melakukan persiapan diatas, sebenarnya dokumentasi sederhana dari API sudah terbuat. Namun, kita masih perlu menambahkan beberapa dokumentasi yang diperlukan yaitu Request Body dan Response dari API.

Untuk menambahkan contoh bentuk object dari Response API kita dapat menambahkan anotasi

[ProducesResponseType((int)HttpStatusCode.<ResponseType>, Type = typeof(<ResponseObject>))]

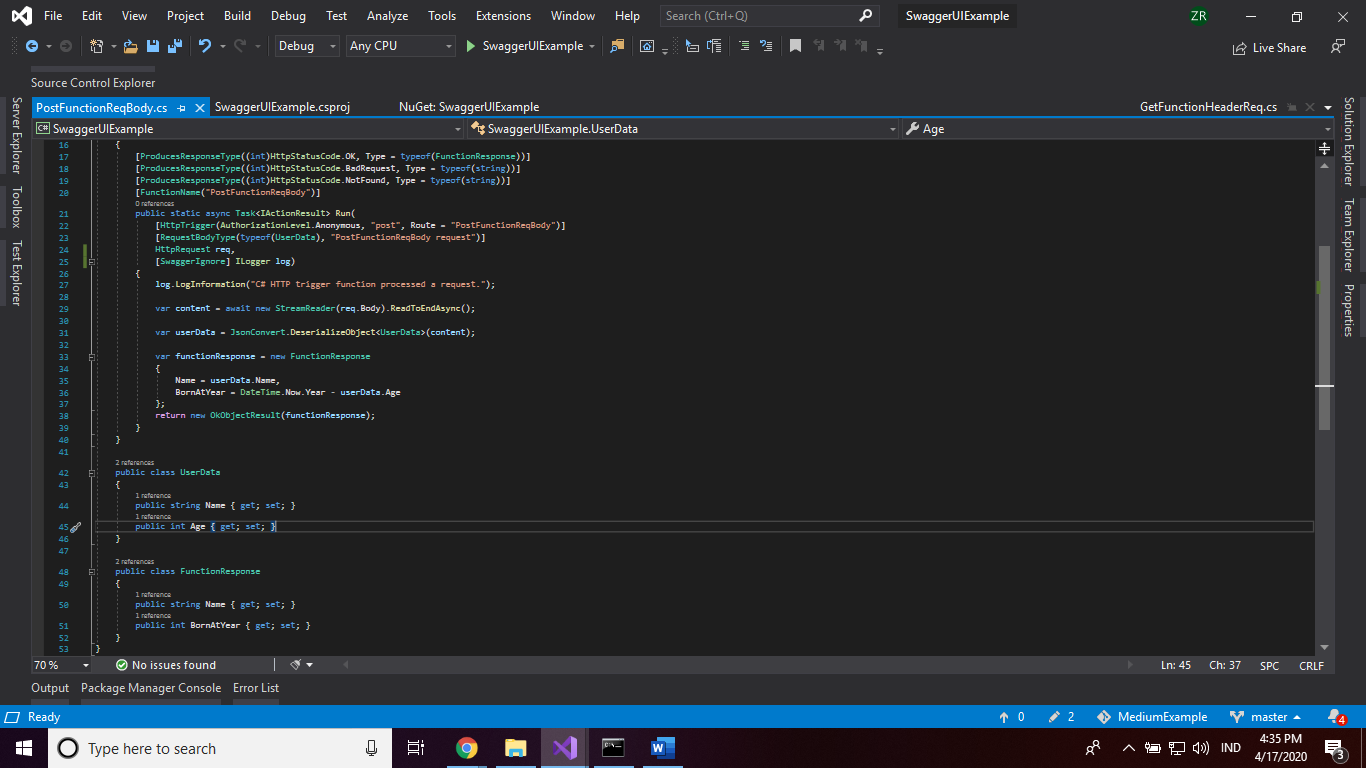
pada fungsi yang diperlukan.

Untuk membuat dokumentasi dari Request Body yang diperlukan oleh API, kita butuh menambahkan sebuah anotasi pada bagian parameter di Azure Function. Kita harus menambahkan

[RequestBodyType(typeof(<RequestBodyObj>), <Description>)]

sebagai anotasi pada HttpRequest.

Hasil code yang sudah menambahkan dokumentasi Response API dan Request Body untuk API dapat dilihat dibawah ini:



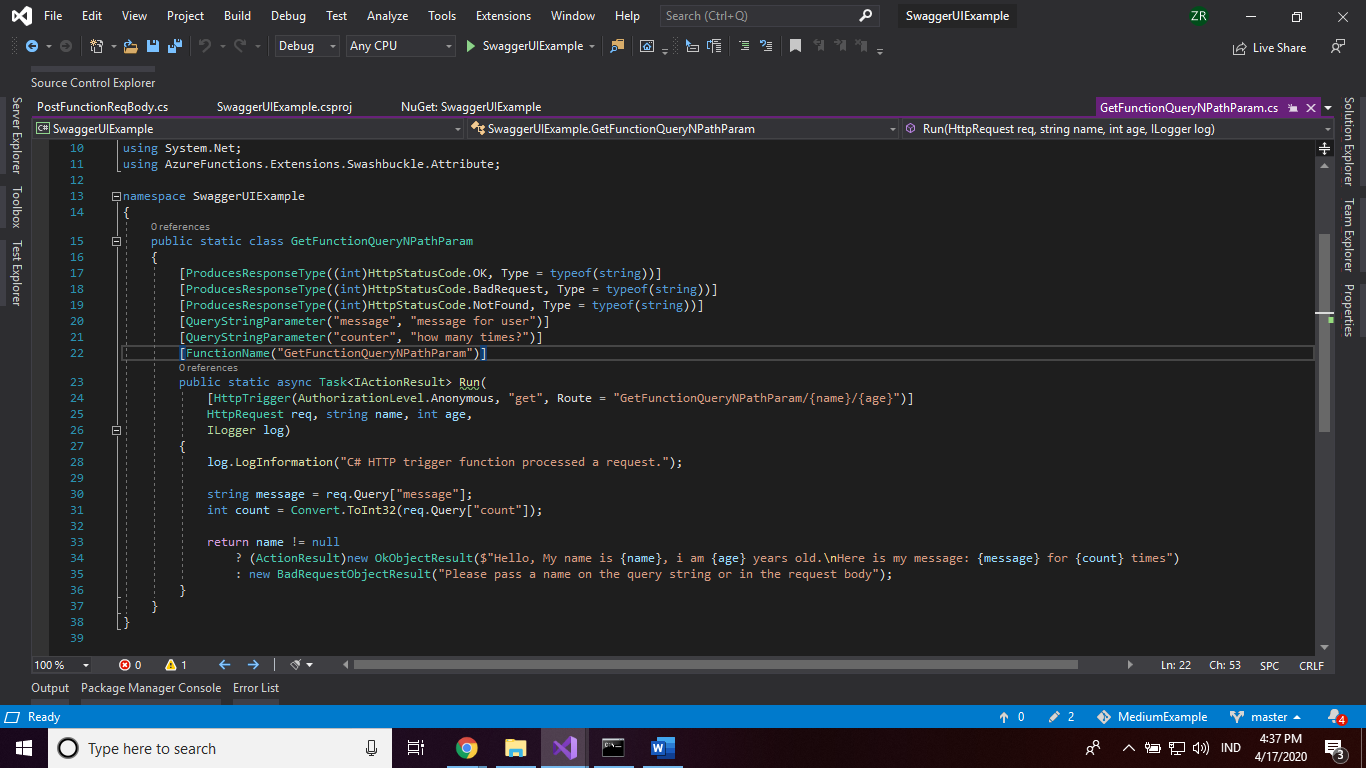
Pada source code diatas, kita menambahkan dokumentasi Request Body dimana API tersebut harus menerima sebuah Request Body yang isi datanya seperti pada object UserData. Sedangkan API tersebut akan memberikan Response berupa object FunctionResponse jika status code OK (200) dan memberikan Response berupa string apabila status code BadRequest (400) atau NotFound (404).

## Membuat Dokumentasi Query Parameter dan Path Parameter pada Method GET

Untuk Path Parameter, Swashbuckle sudah secara otomatis membuat dokumentasi untuk kita, namun tidak untuk Query Parameter. Untuk menambahkannya, kita perlu menambahkan anotasi

[QueryStringParameter(<queryAttribute>, <queryAttrDesc>)]

pada fungsi yang membutuhkannya. Contoh source code dapat dilihat dibawah ini:



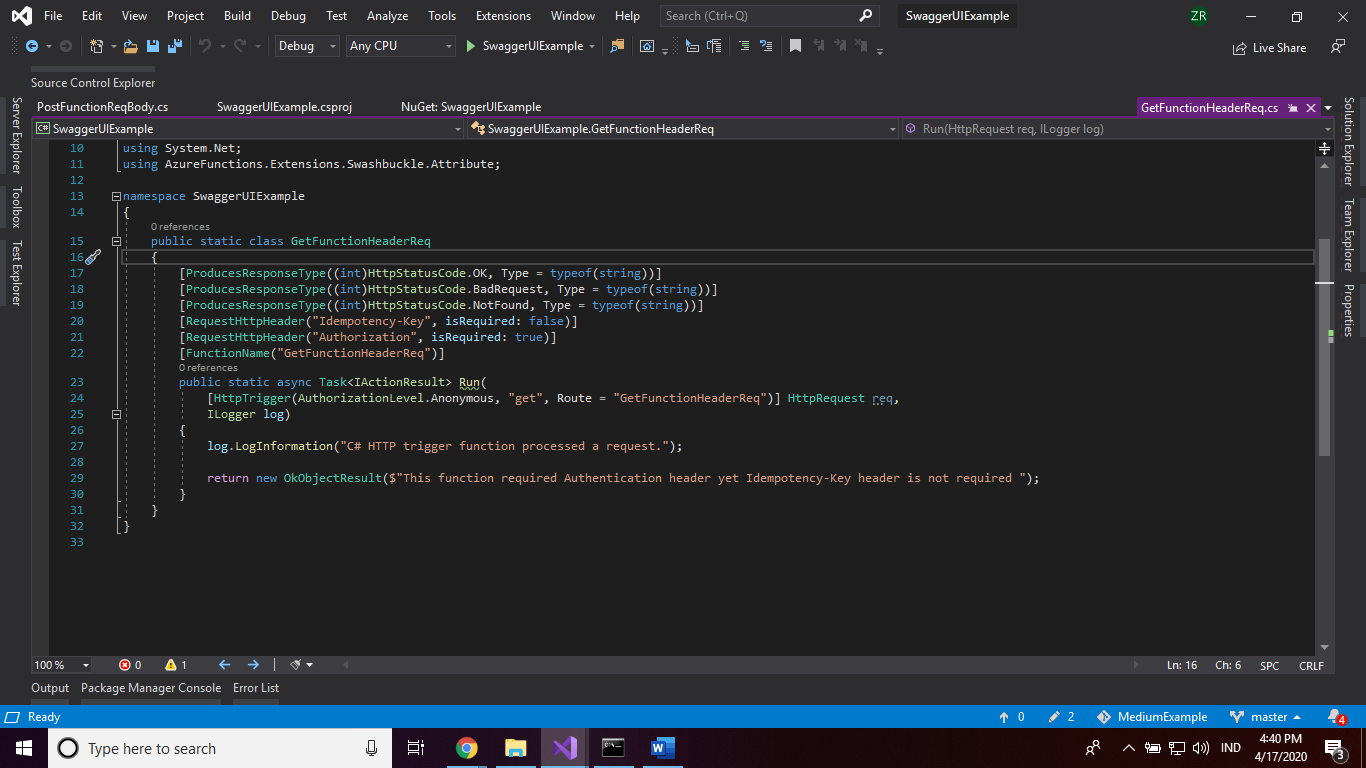
Pada source code diatas, terlihat bahwa API tersebut memiliki dua Path Parameter yaitu name dan age serta dua Query Parameter yaitu message dan count.

## Membuat Dokumentasi Header yang Diperlukan pada Sebuah API

Untuk beberapa API terkadang memerlukan header tertentu. Seperti pada saat kita menggunakan JWT token sebagai media otentikasi dan otorisasi untuk seorang user. Untuk membuat dokumentasi dari header pada sebuah API, kita dapat menambahkan anotasi

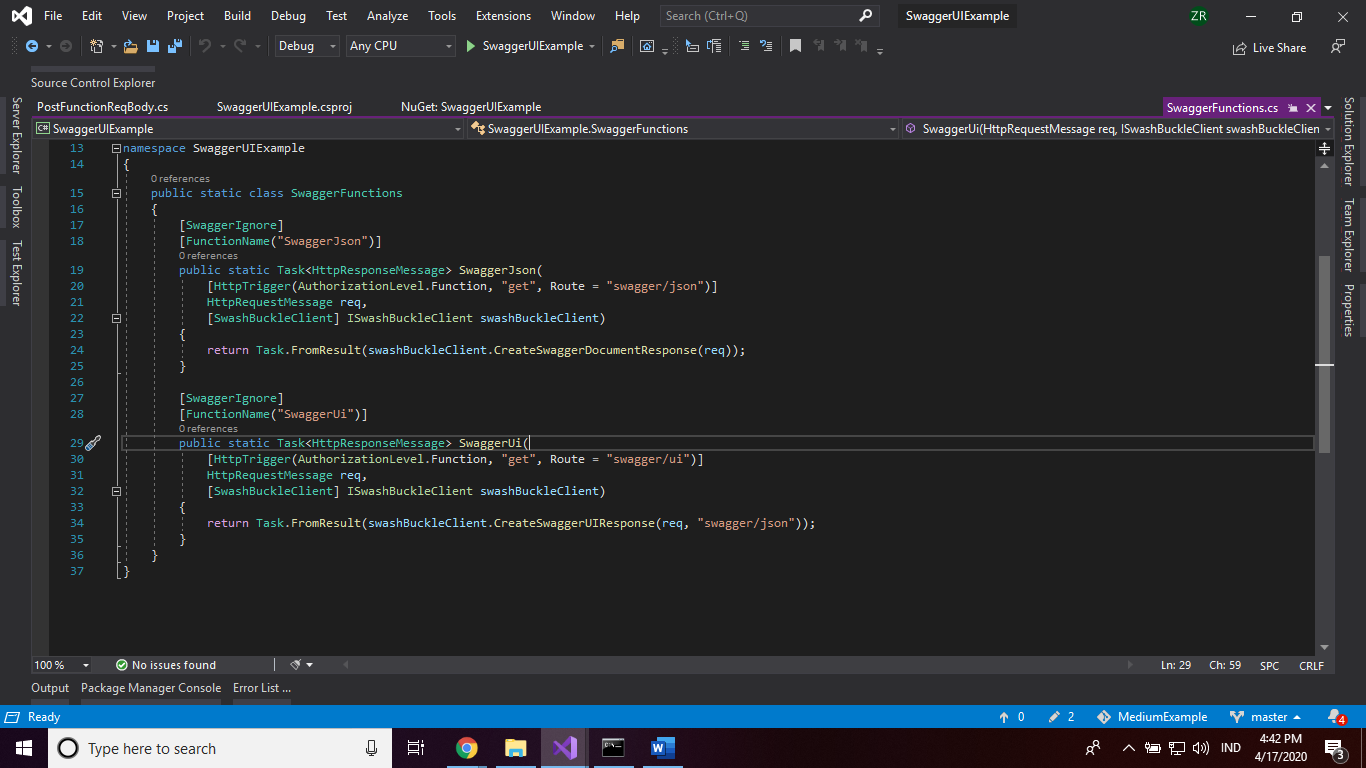
[RequestHttpHeader(<headerAttrName>, isRequired: <boolean>)]

pada fungsi yang membutuhkannya. Berikut merupakan contoh source code yang mengimplementasikannya:

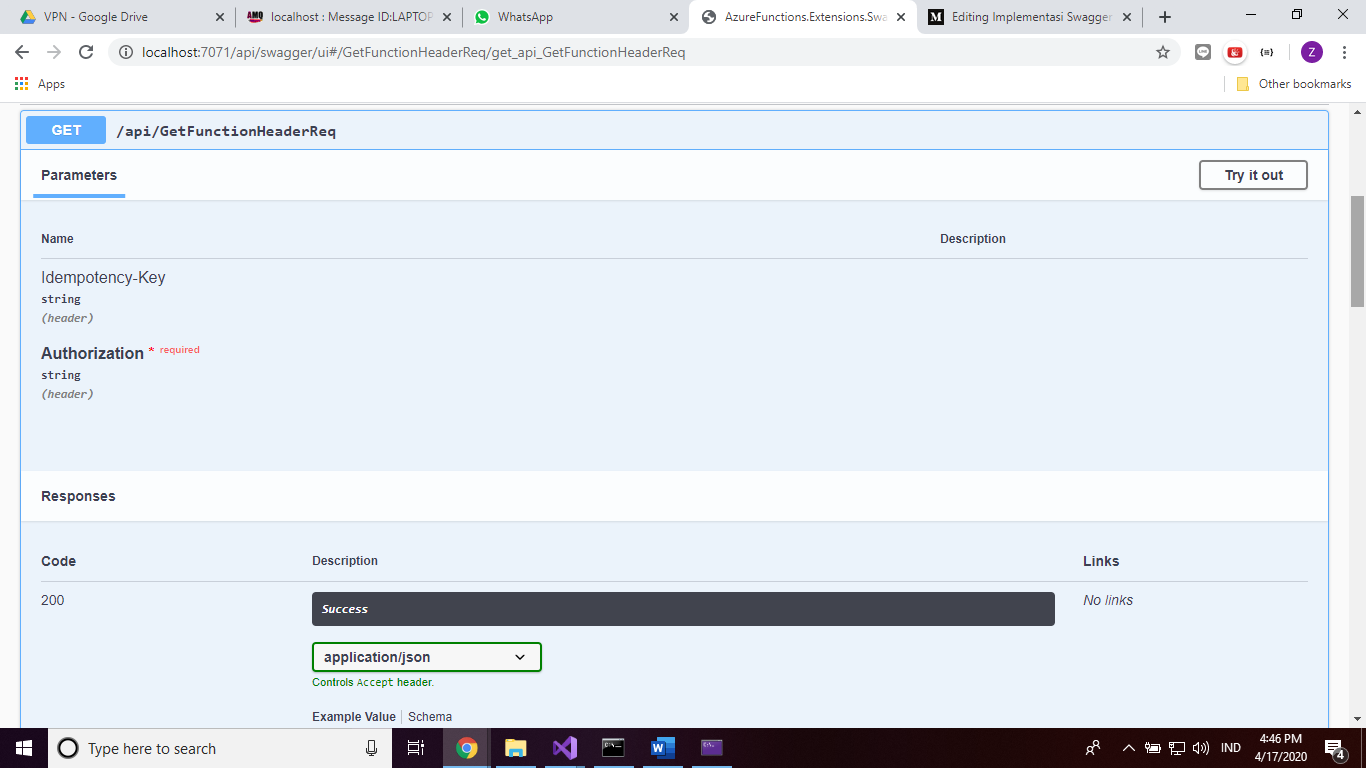


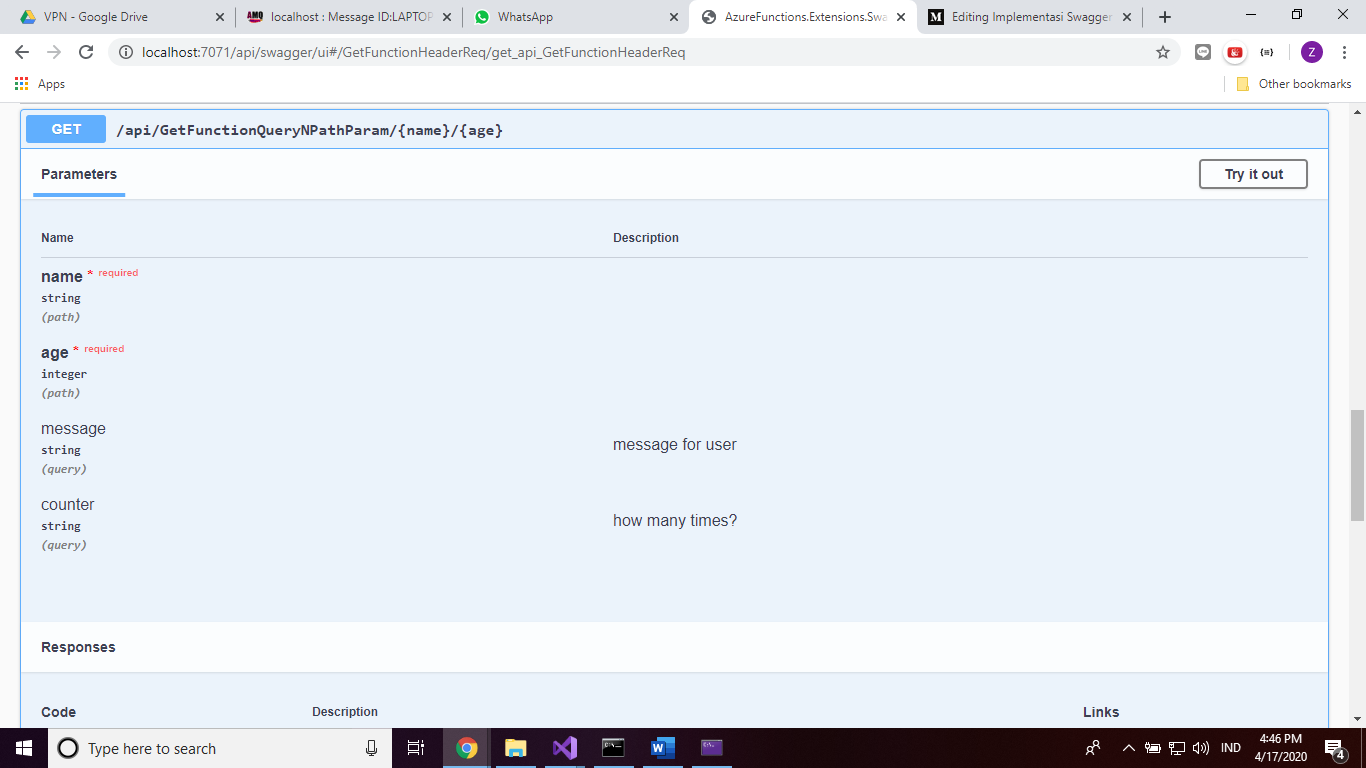
## Meng-exclude API dari Swagger UI

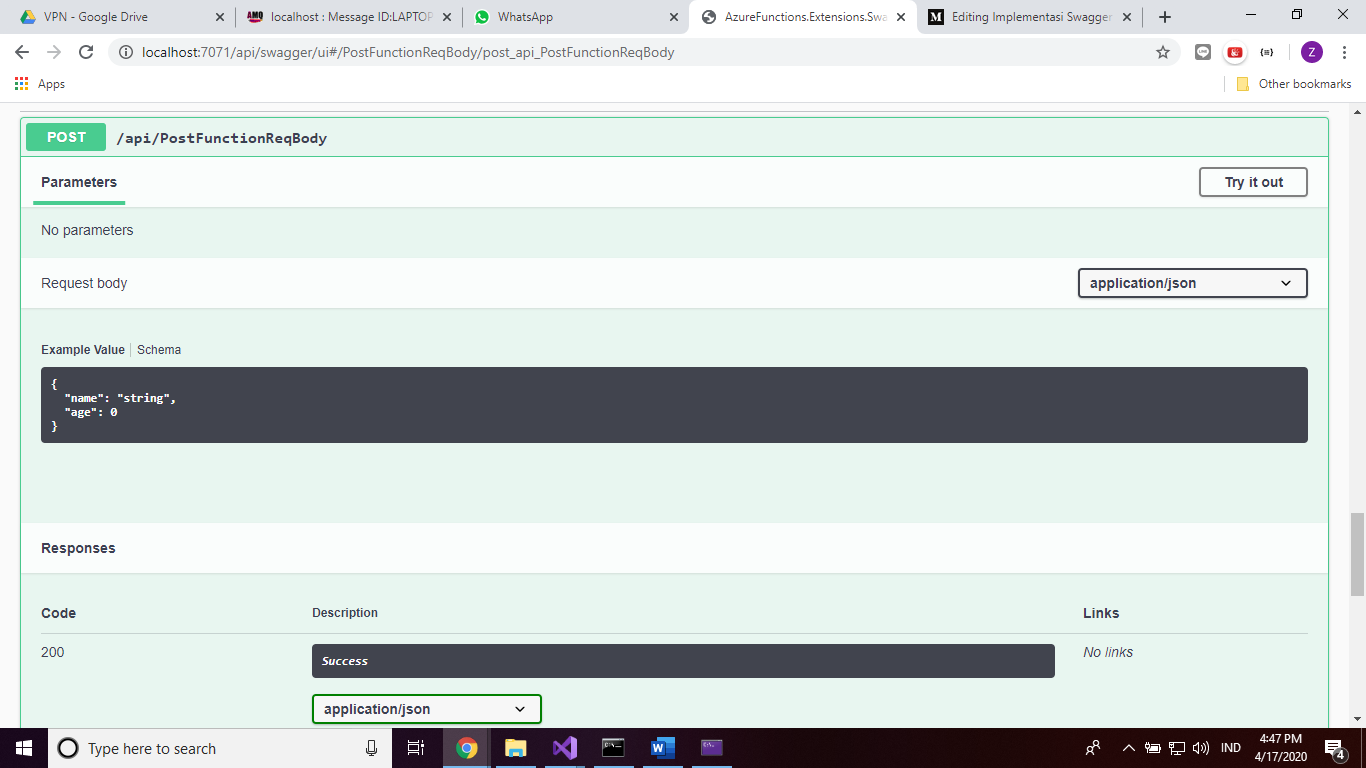
Jika terdapat API atau parameter yang tidak ingin ditampilkan kita dapat menambahkan anotasi [SwaggerIgnore] pada API yang tidak kita inginkan untuk tampil di Swagger UI. Seperti pada contoh berikut:

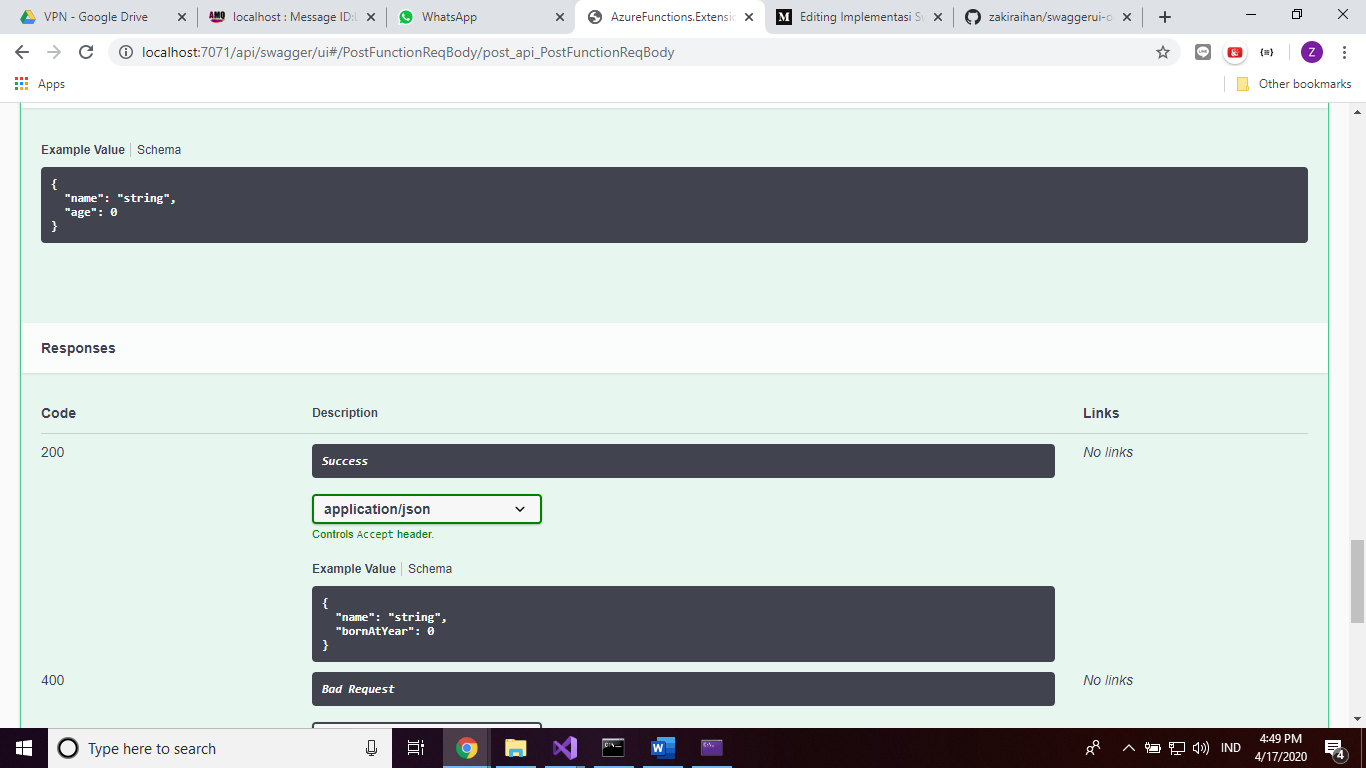


## Contoh hasil pembuatan Swagger UI









## Penutup

Project yang dijadikan contoh dapat diakses pada: <https://github.com/zakiraihan/swaggerui-on-azure-function>