**Final Project**

Nama : Mohamad Edward Moche

Kode Peserta : FSDO003ONL006

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Soal :

Pada Final Projects ini silahkan buat REST API untuk Online Debit/Credit Card. Cukup kerjakan 1 Modul : Modul payment details dengan Method CRUD Perlu diperhatikan untuk penilaian final project mencakup : - Pembuatan REST API harus menggunakaan .Net 5 - Aplikasi-kan EF Model & DB Context Class - Sql/Sqli Migration - Create API Controller = CRUD Web Methods - Test CRUD Operations on Postman and Swagger

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

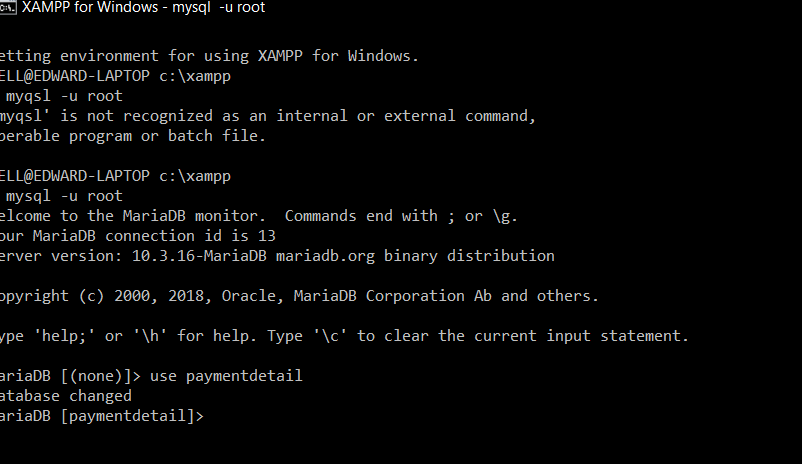
Nama database local : paymentdetail

Link:

Git = https://github.com/edwardmoche/FinalProject

Heroku = [Mohamad Edward Moche (edward-payment-api.herokuapp.com)](https://edward-payment-api.herokuapp.com/)

1. Buat database terlebih dahulu (database yang saya buat bernamaPaymentDetail)



1. Buat project

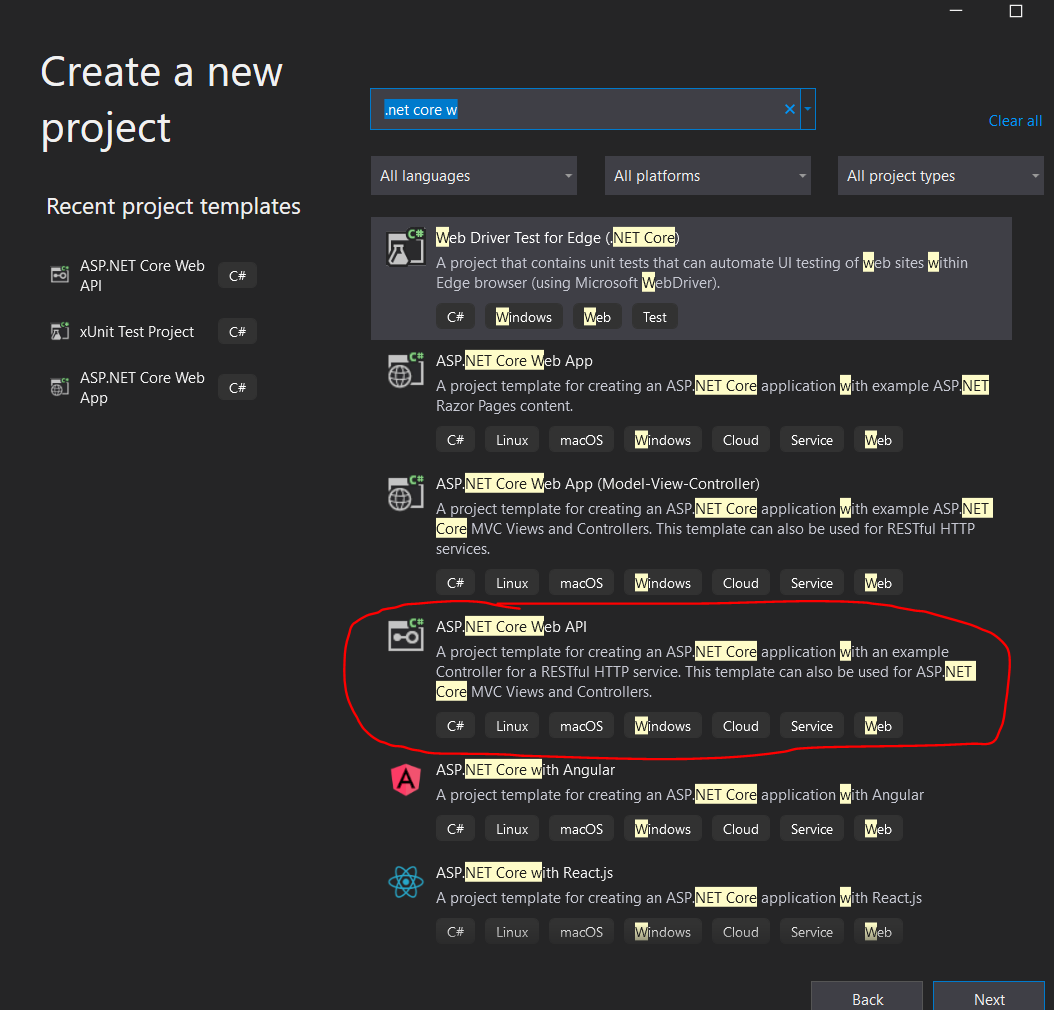
Untuk pembuatan project bisa menggunakan visual studio bisa juga menggunakan visual studio code.

Untuk visual studio code gunakan

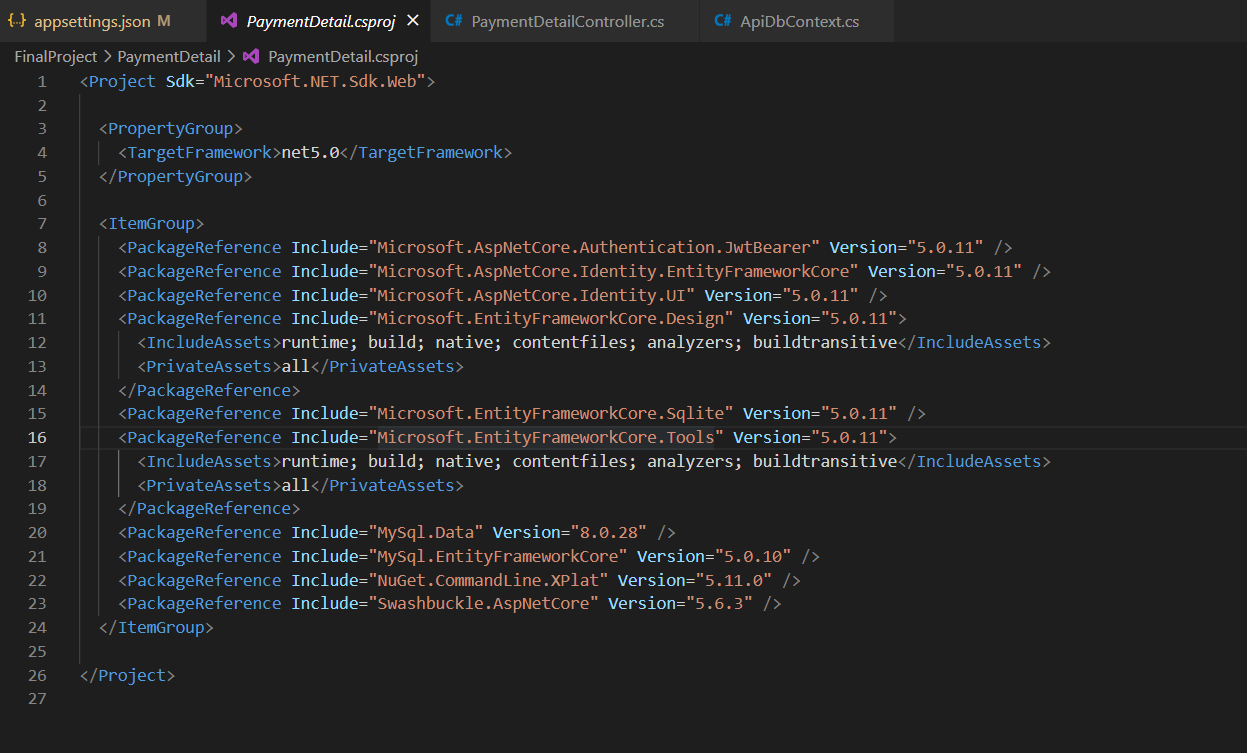
dotnet new webapi -n "(nama aplikasi)" -lang "C#" -au none

pada terminal, terlebih dahulu sesuaikan apakah kita sudah berada pada folder yang sesuai.

Untuk pembuatan di visual studio pilih create new project > asp.net core web api



1. Library yang dipakai pada aplikasi

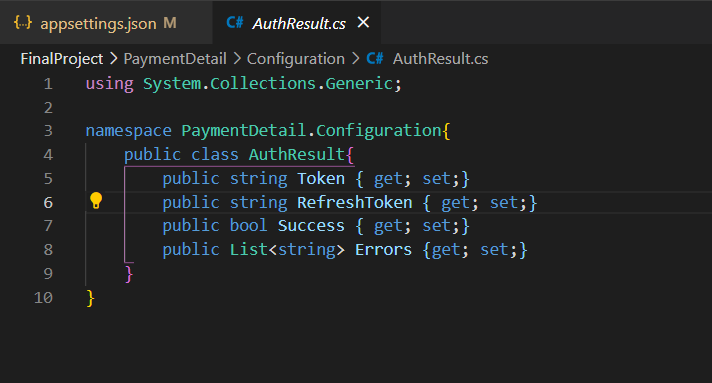


Untuk penginstalan package pada visual studio code, kita ketikan

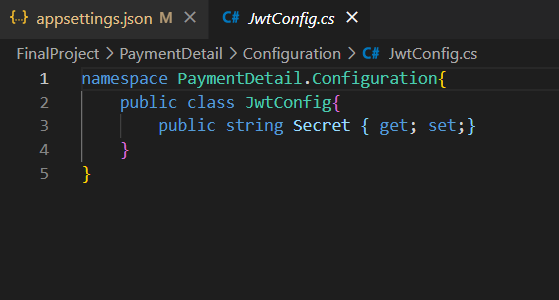
dotnet add package (nama package) --version (versi package)

\* Tambahan: untuk instalasi package disarankan pada pembuatan project untuk menggunakan visual studio karena lebih mudah dari segi penginstalan package (bisa liat ver dsb), selanjutnya untuk pengerjaan bisa kembali menggunakan visual studio code.

1. Authresult untuk pengembalian respon dari authentication dengan memberikan token untuk pengaksesan aplikasi (payment detail)



1. Berikut adalah scrip untuk configurasi dari Jwt (Jwtconfig)



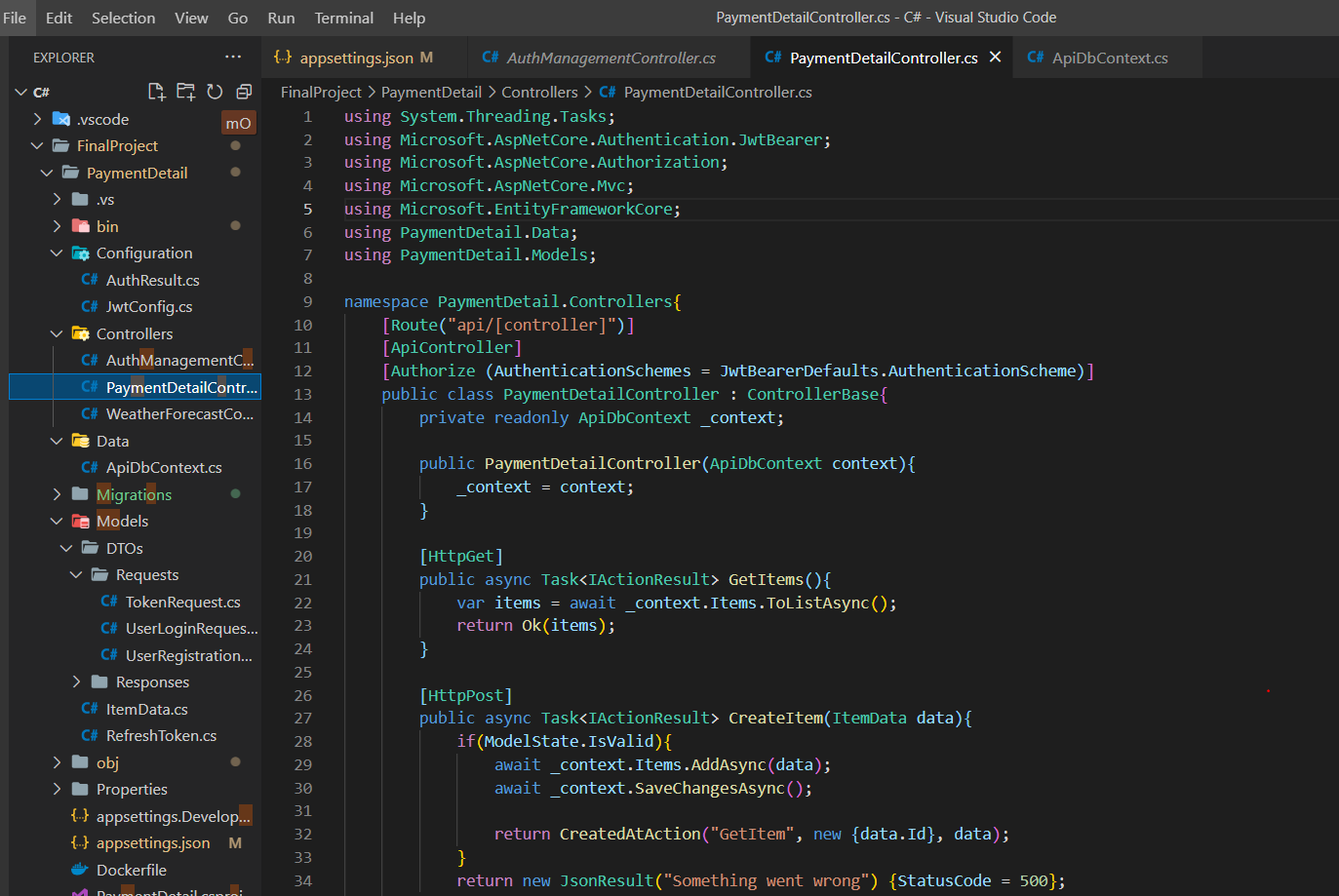
1. Authmanagecontroller

Authmanager berfungsi untuk menjalankan fitur registrasi, login, dan authentication pada aplikasi yang kita buat. Point yang perlu di perhatikan disini ada pada bagian expires, yang dimana disini kita dapat memberikan hak akses kepada orang lain dengan cara memberikan 1 token selama waktu tertentu untuk dapat mengakses fitur dari aplikasi. Disini diatur agar token berganti setiap 30 menit sekali. Apabila waktu telah habis, kita tidak dapat mengakses fitur fitur yang diberikan, oleh karena itu user akan diminta untuk login kembali agar mendapatkan akses token yang baru.



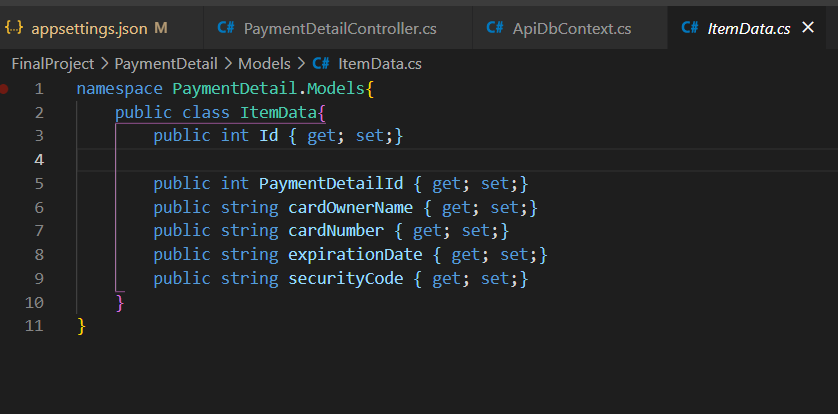
1. PaymentDetailController

Controller ini berfungsi dalam mengatur crud di dalam aplikasi payment detail

s

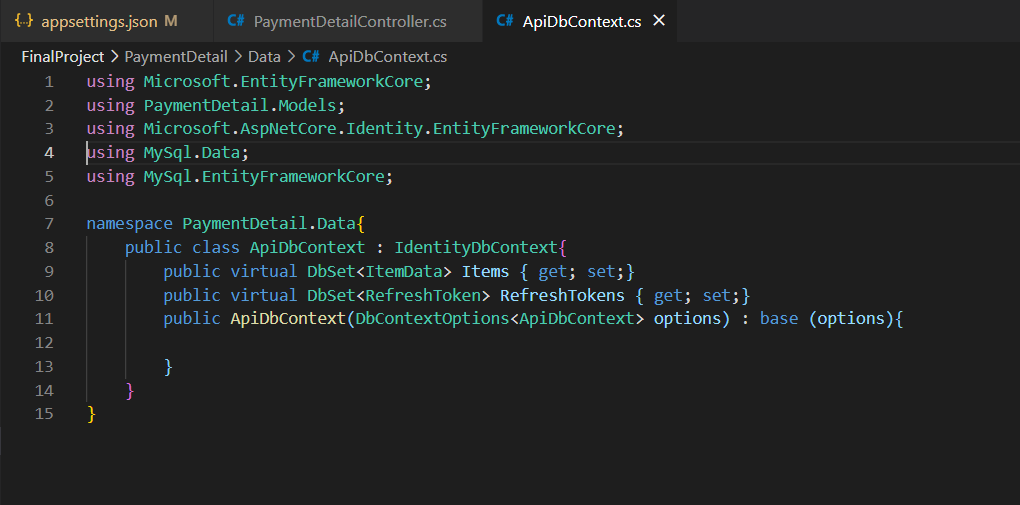
1. Itemdata

Script ini berfungsi untuk mengatur table item data



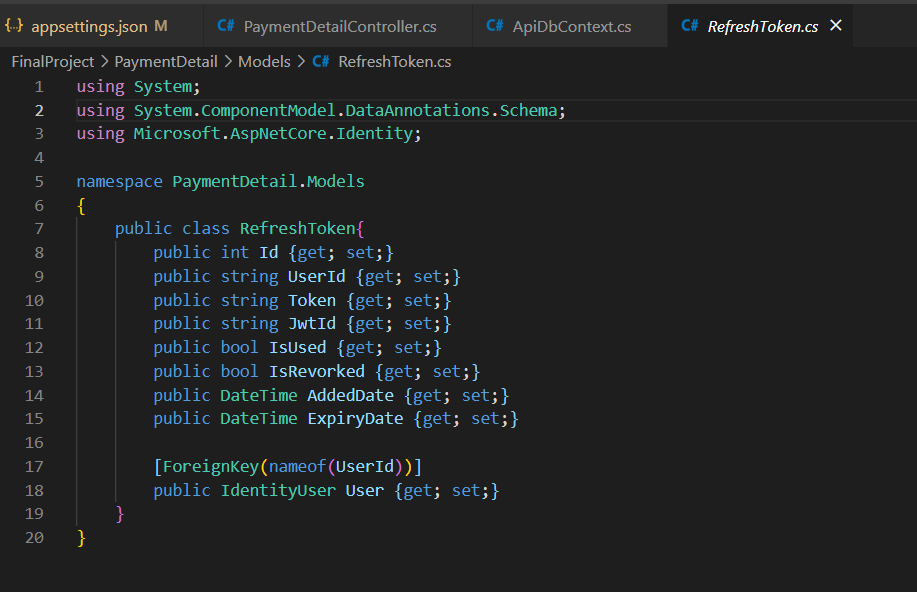
1. Apidbcontext

Script ini berfungsi untuk menjalankan table table yang telah kita buat di dalam models (item data, refresh token)



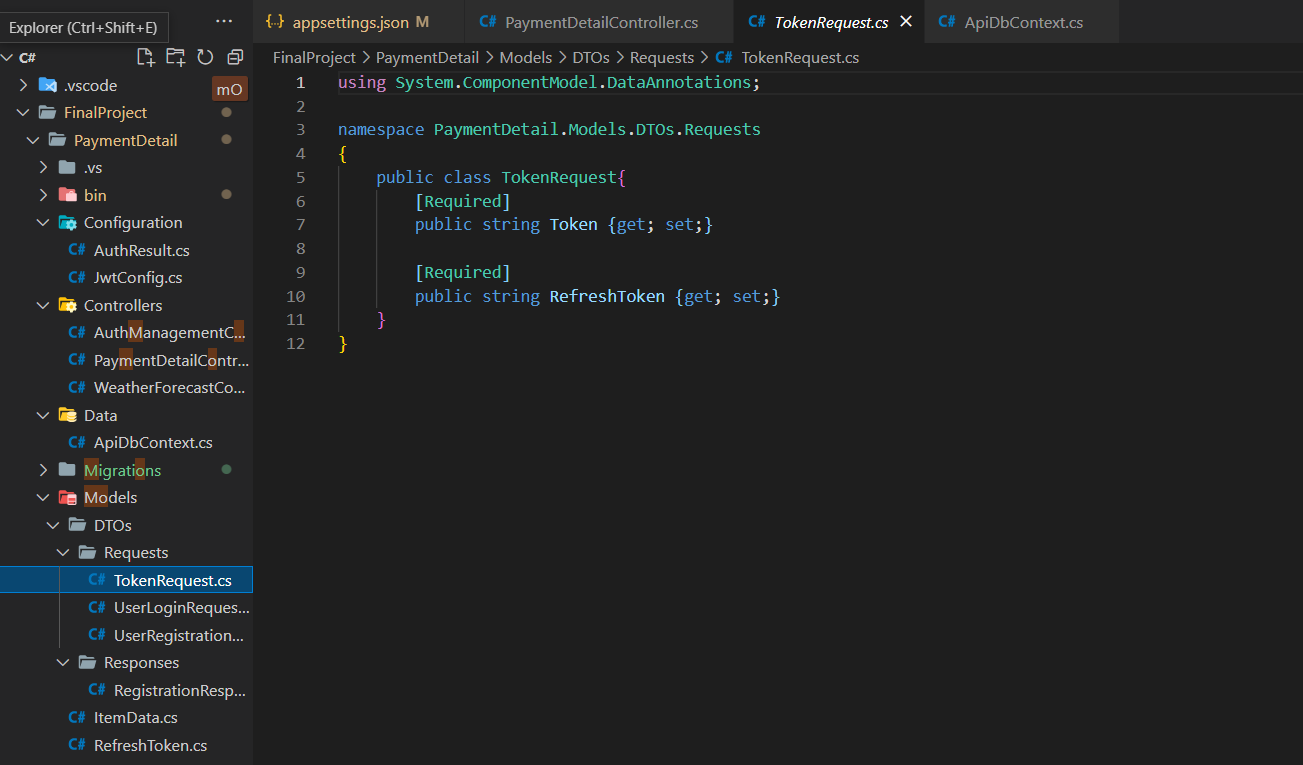
1. Refresh Token

Script ini berfungsi untuk mengatur table item data



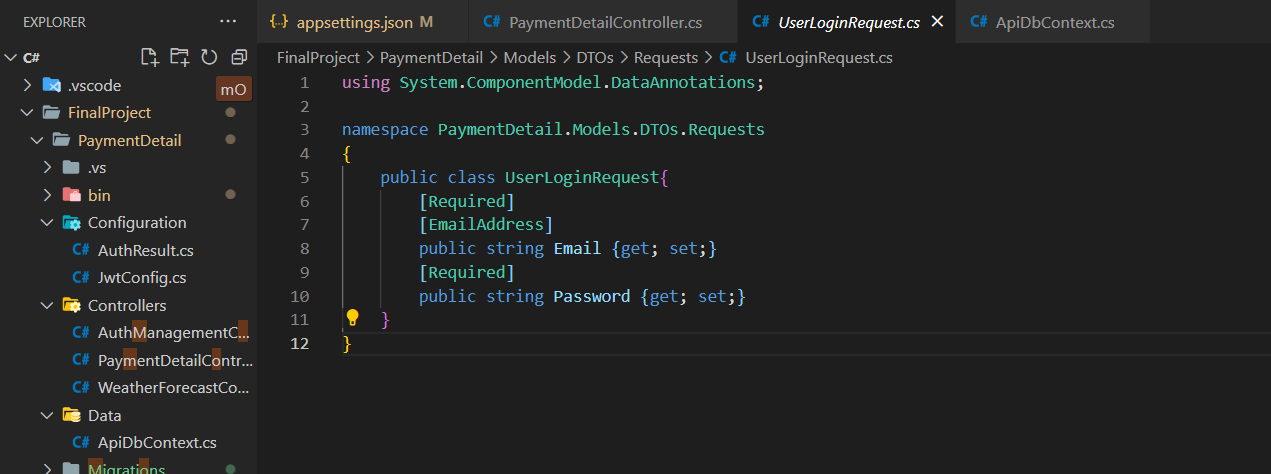
1. Token request

Script ini yang nantinya akan digunakan untuk pengecekan apakah token yang diberikan sudah kadaluarsa/tidak



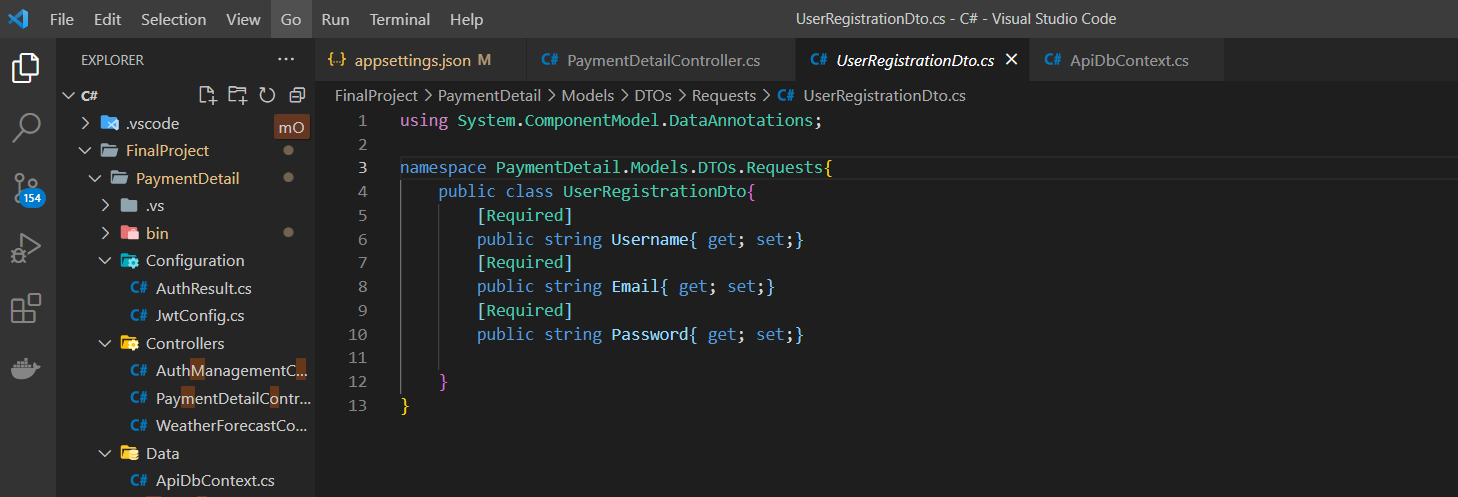
1. Userlogin

Script ini untuk melakukan login



1. User regist

Script ini untuk menambahkan user baru



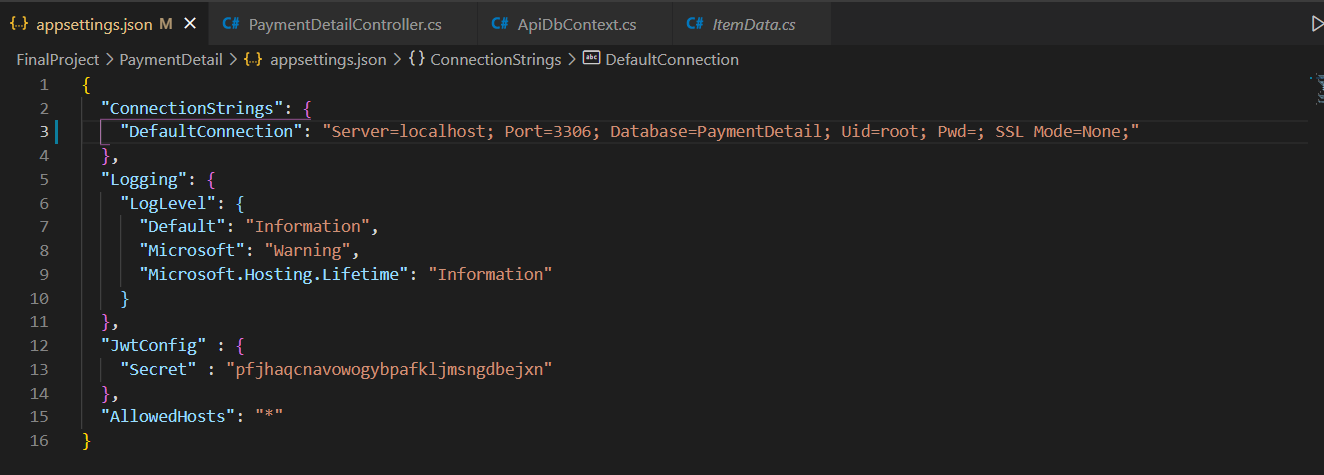
1. Response

Script untuk memangil authresult



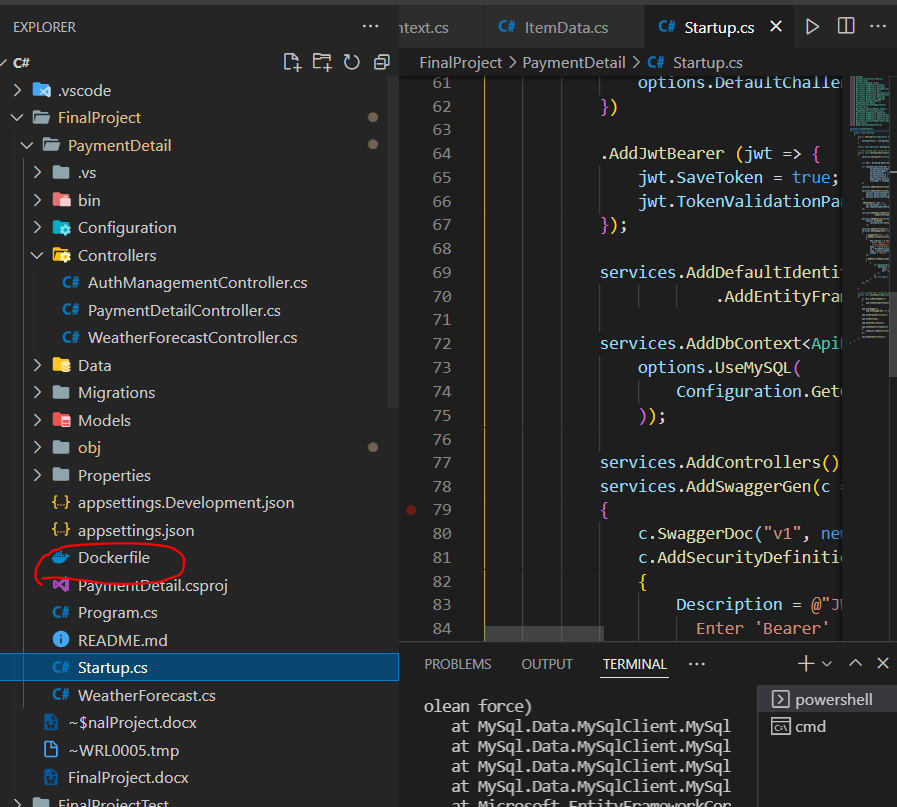
1. App setting (koneksi db)

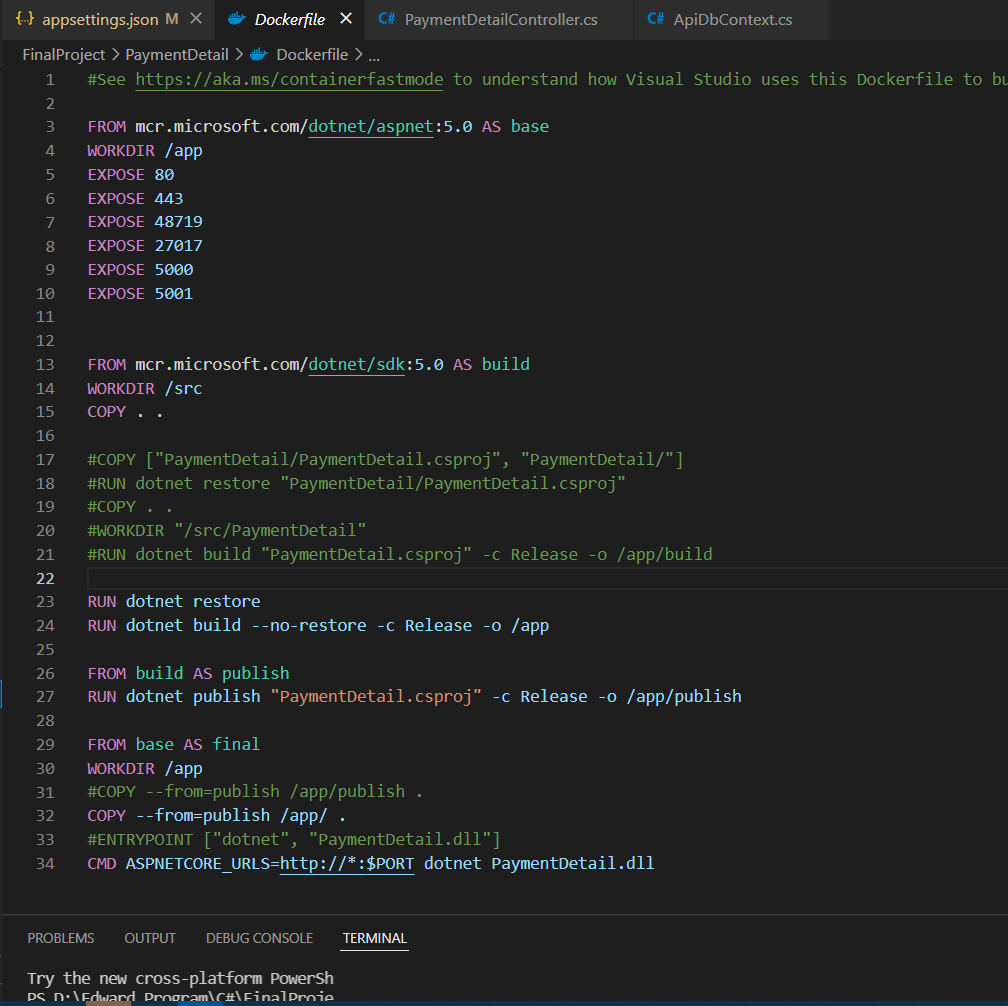
Script ini berfungsi untuk menghubungkan aplikasi kita dengan database yang kita buat



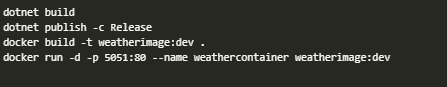
1. Docker

Docker untuk menyatukan berbagai file software dan pendukungnya dalam sebuah wadah (container) agar memudahkan proses pengembangan software.

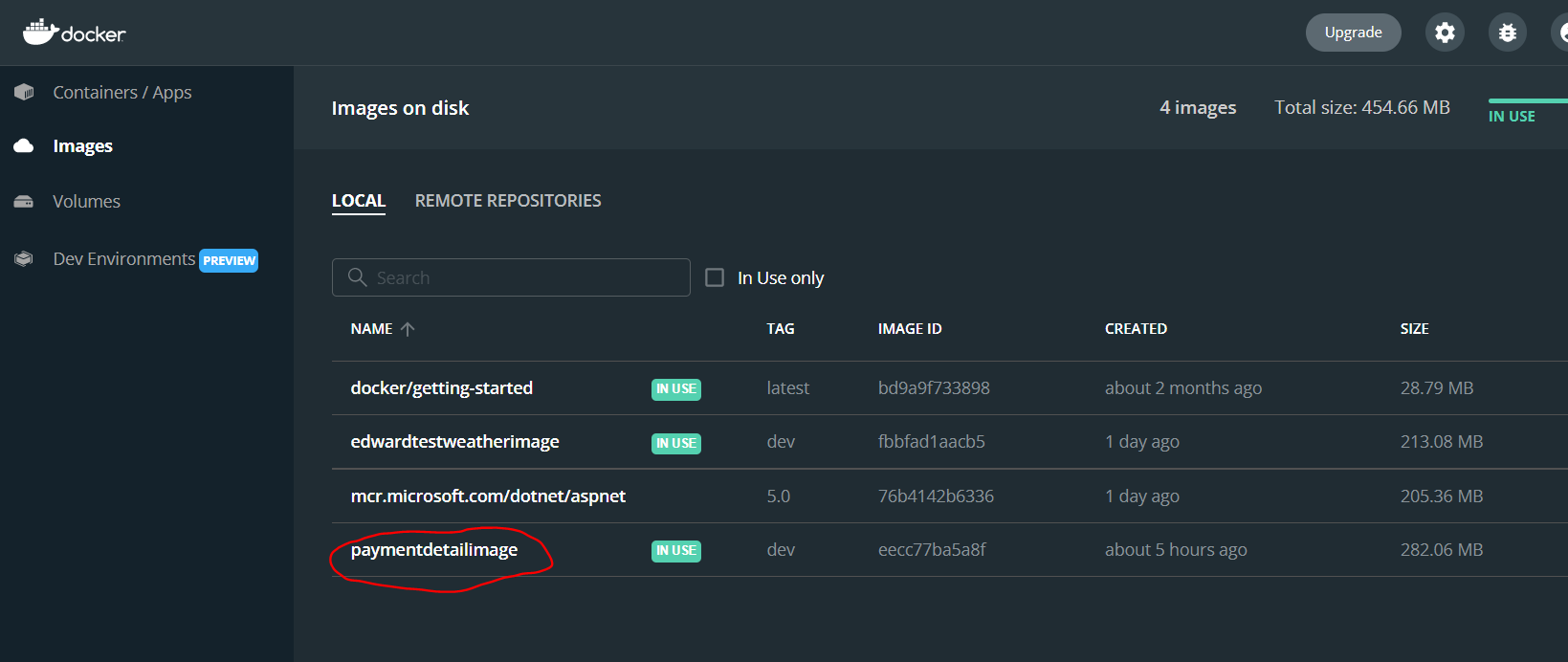


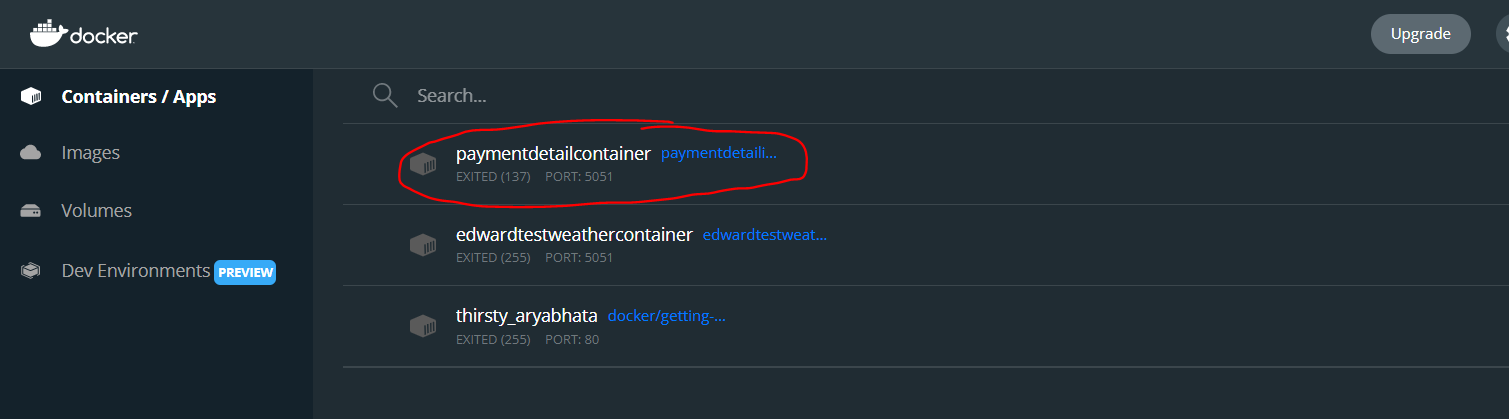


1. Build docker



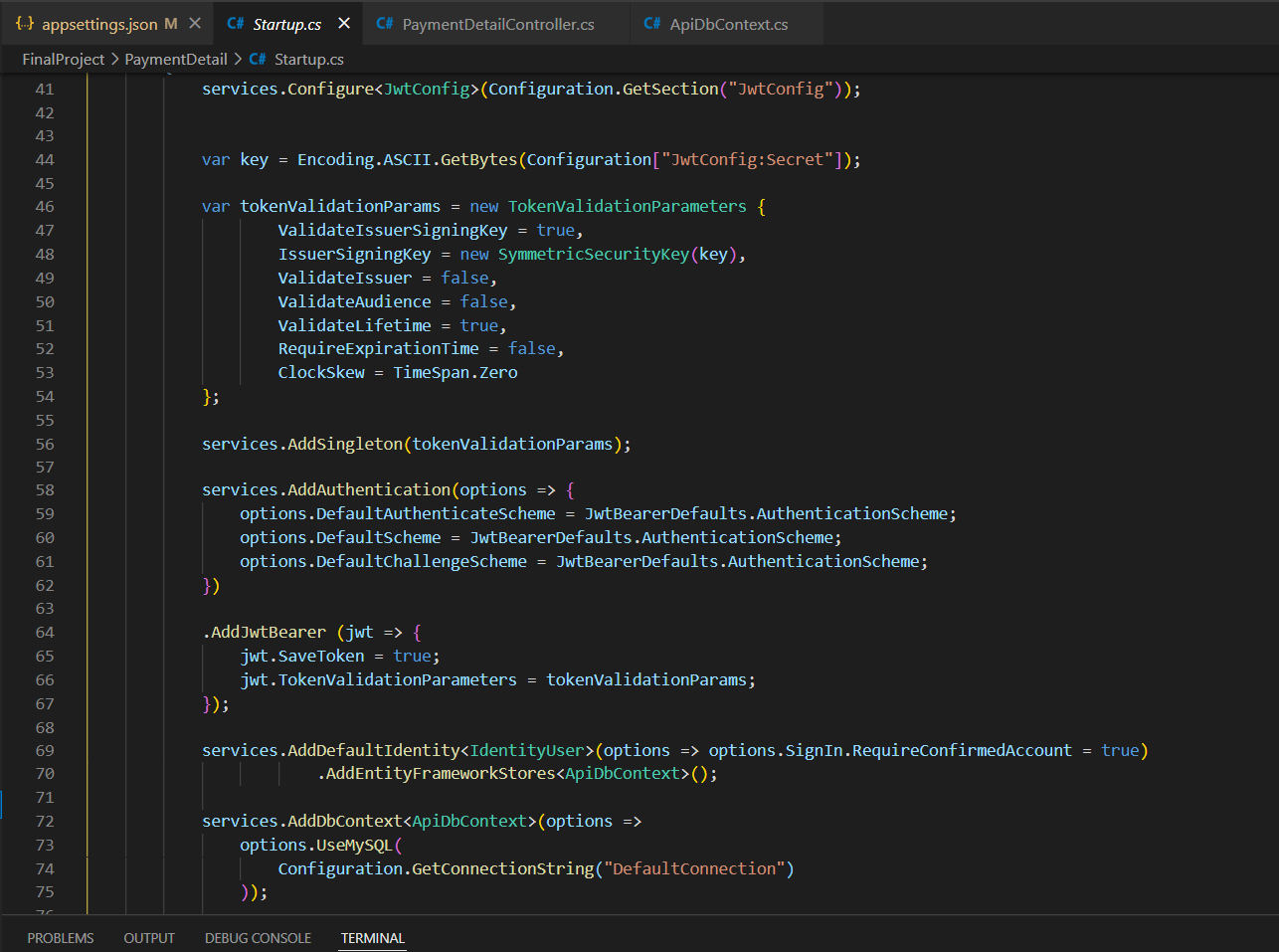
Tambahan : untuk weather pada gambar diatas diganti dengan nama project kalian, dengan penamaan lowercase



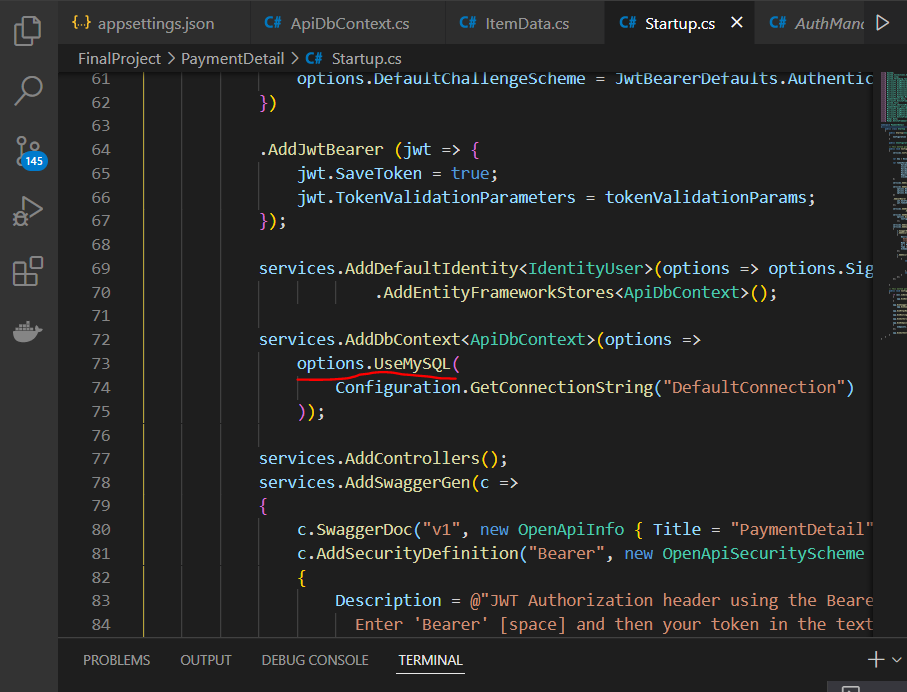


1. Startup

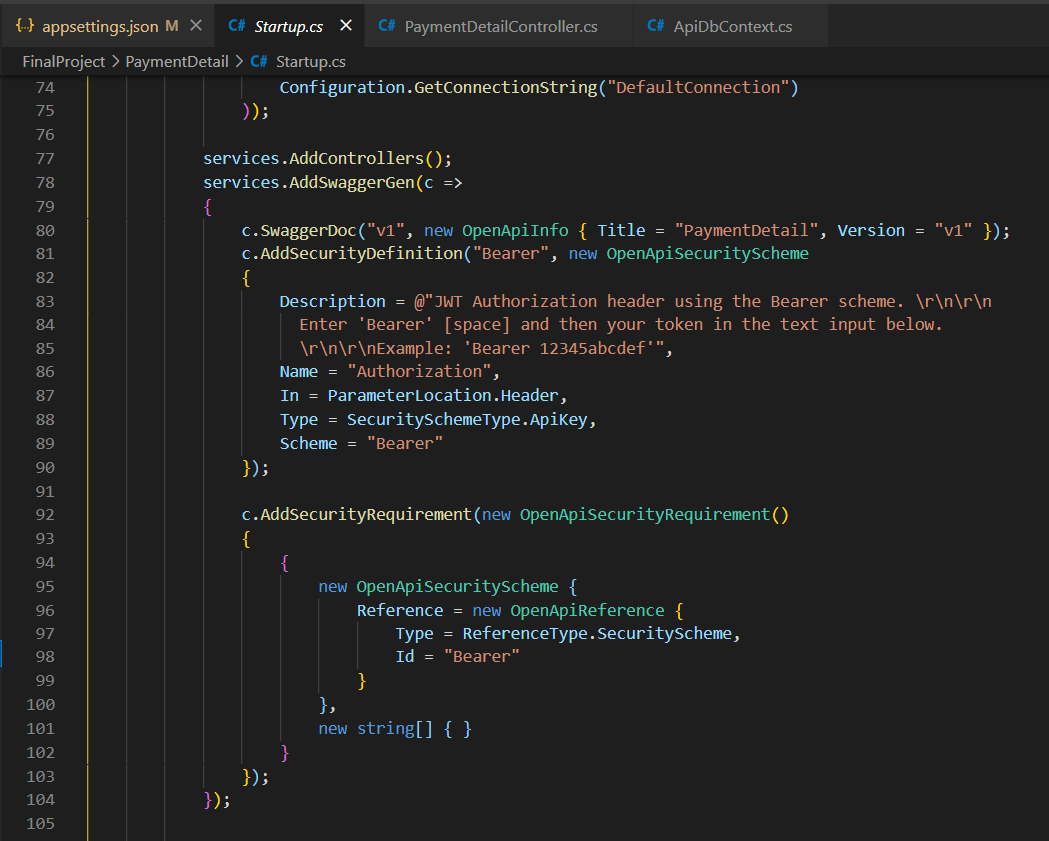
Inisiasi jwt



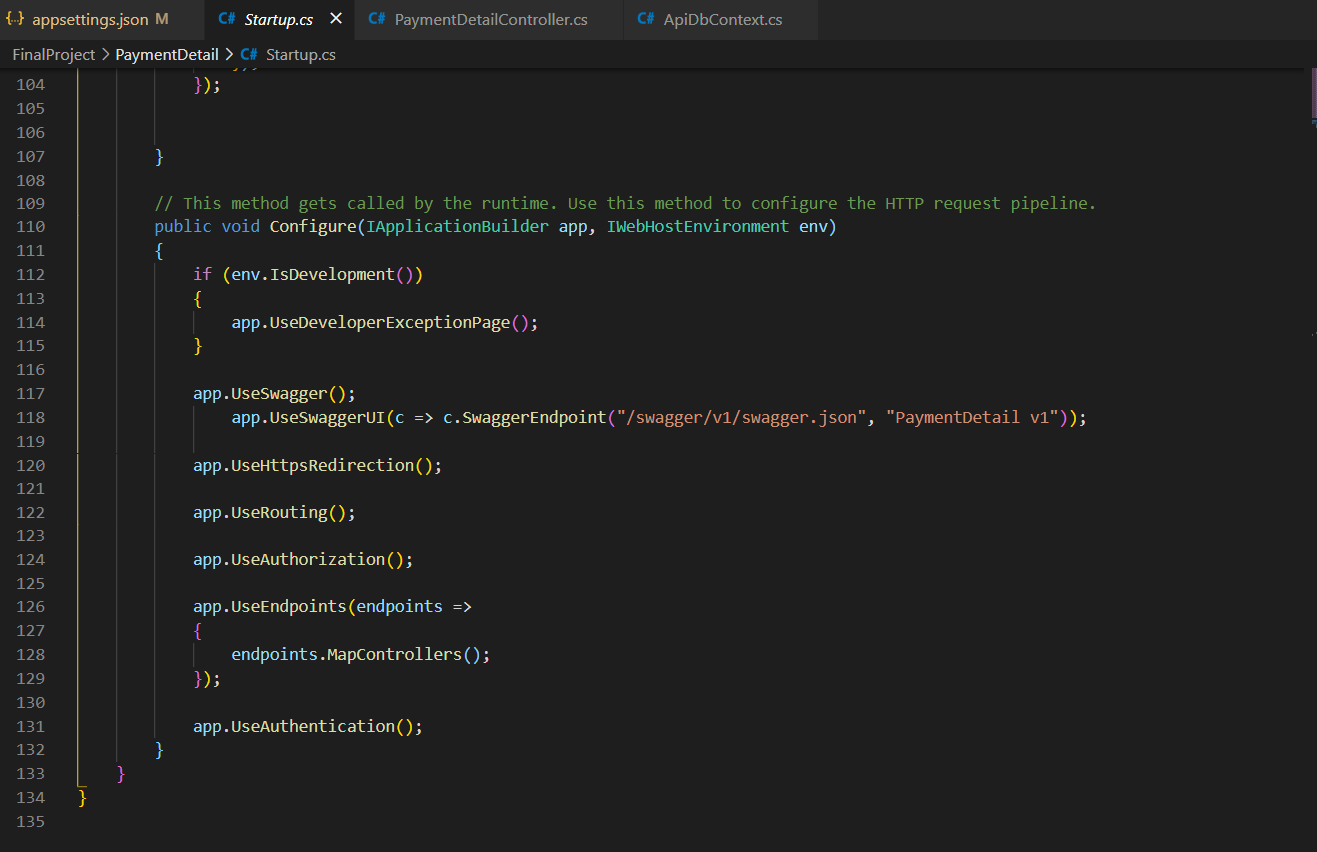
Penggunaan mysql sebagai database



Penambahan fungsi button authorize untuk swagger

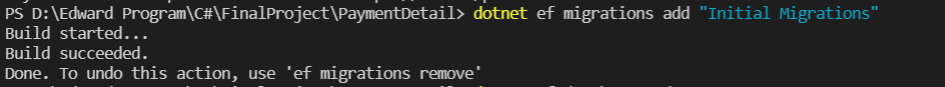


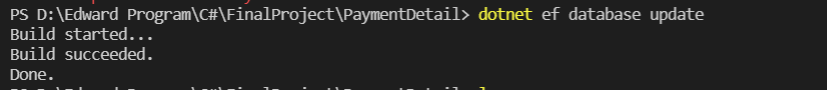
Keluarkan swagger dari env.isdevelopment agar dapat digunakan setelah di publish



1. dotnet ef (migrasi db)

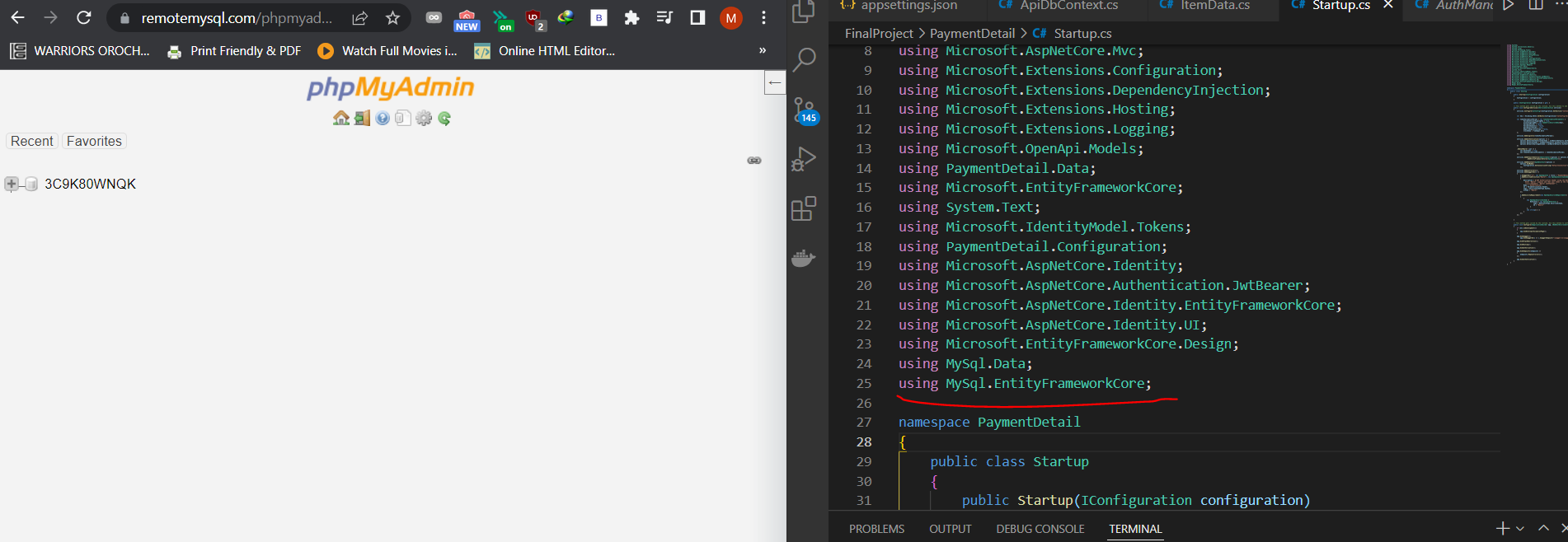
command ini berfungsi untuk me migrasikan table ke dalam database



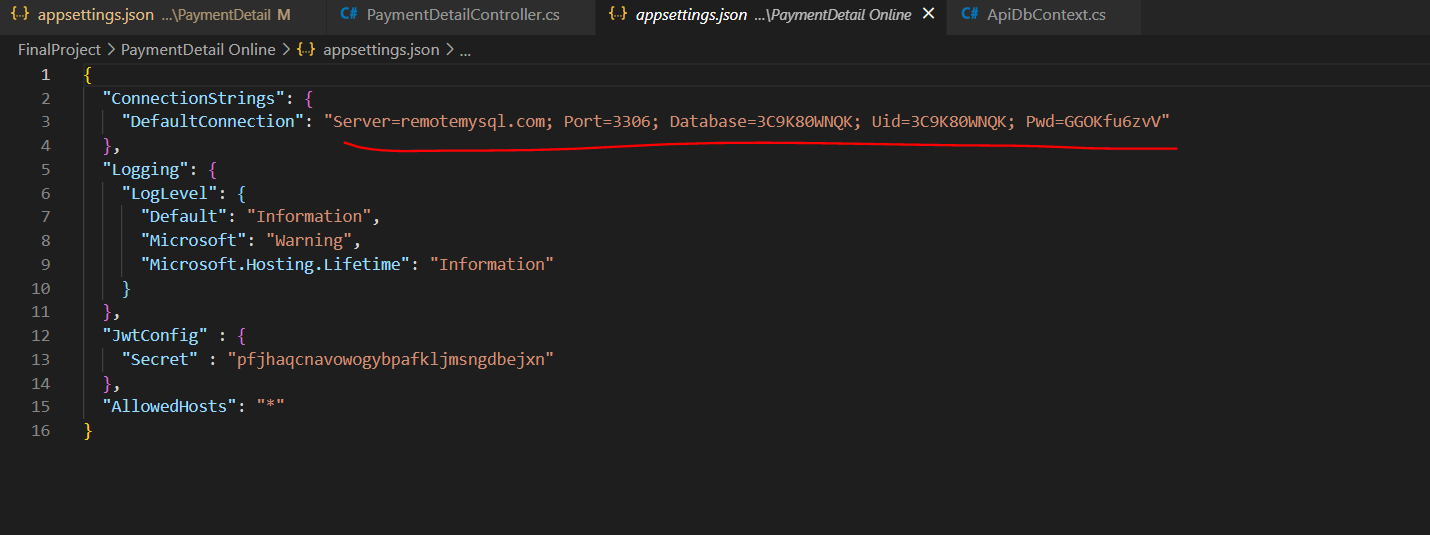


1. penggunaan database online

untuk project kali ini saya menggunakan remotemysql.com, cara penggunaannya yaitu pertamam kita mendaftar/register terlebih dahulu



1. Jika sudah login dan mendapatkan database, user, dan password. Masukkan data tersebut ke appsetting



1. push ke github branch main

Cara push ke repository git baru

Git init

Echo “# (nama repo)’ >> README.md

Git add .

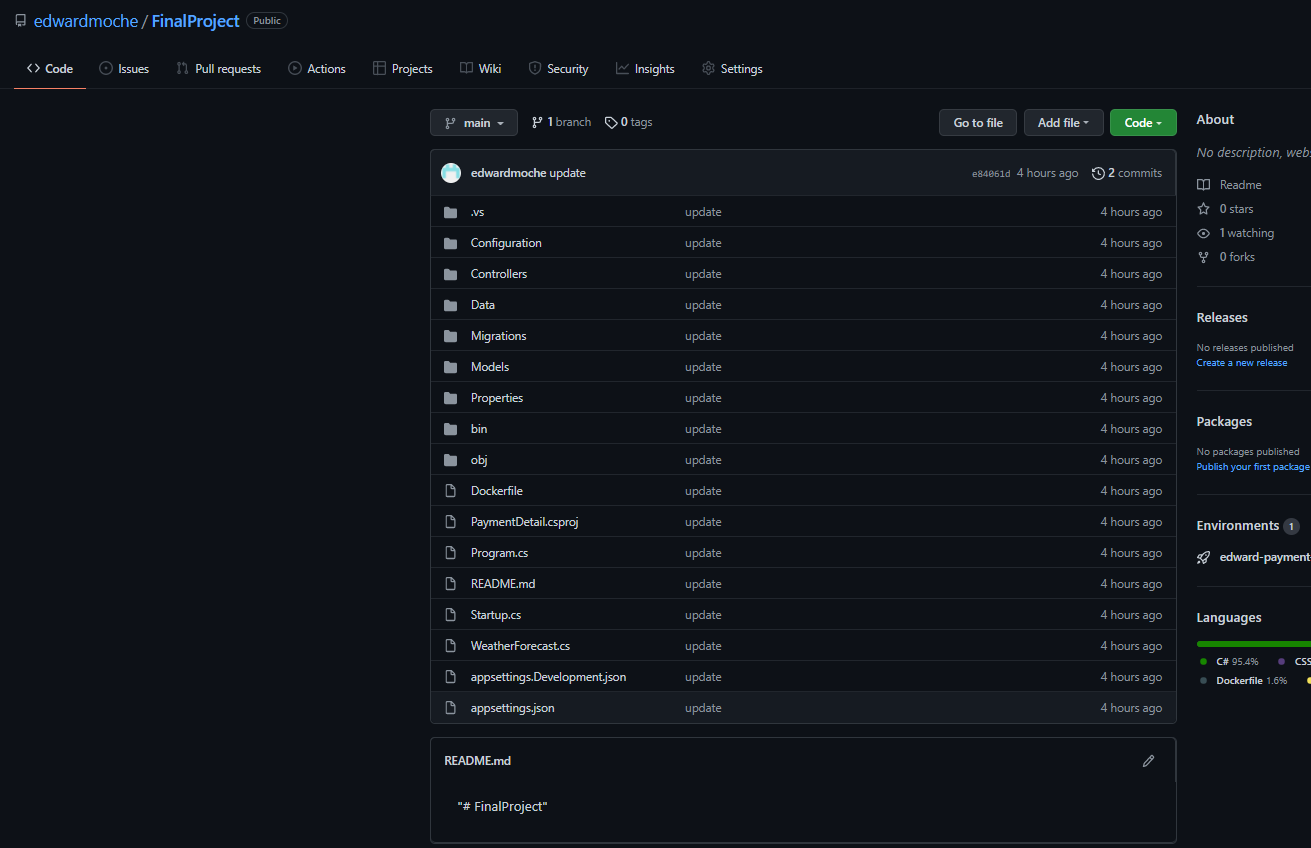
Git commit –m “(komentar yang ingin diberikan)”

Git branch –M main

Git remote add origin (link repo.git)

Git push –u origin main

Jika sudah maka akan tampil didalam repository git yang kita buat

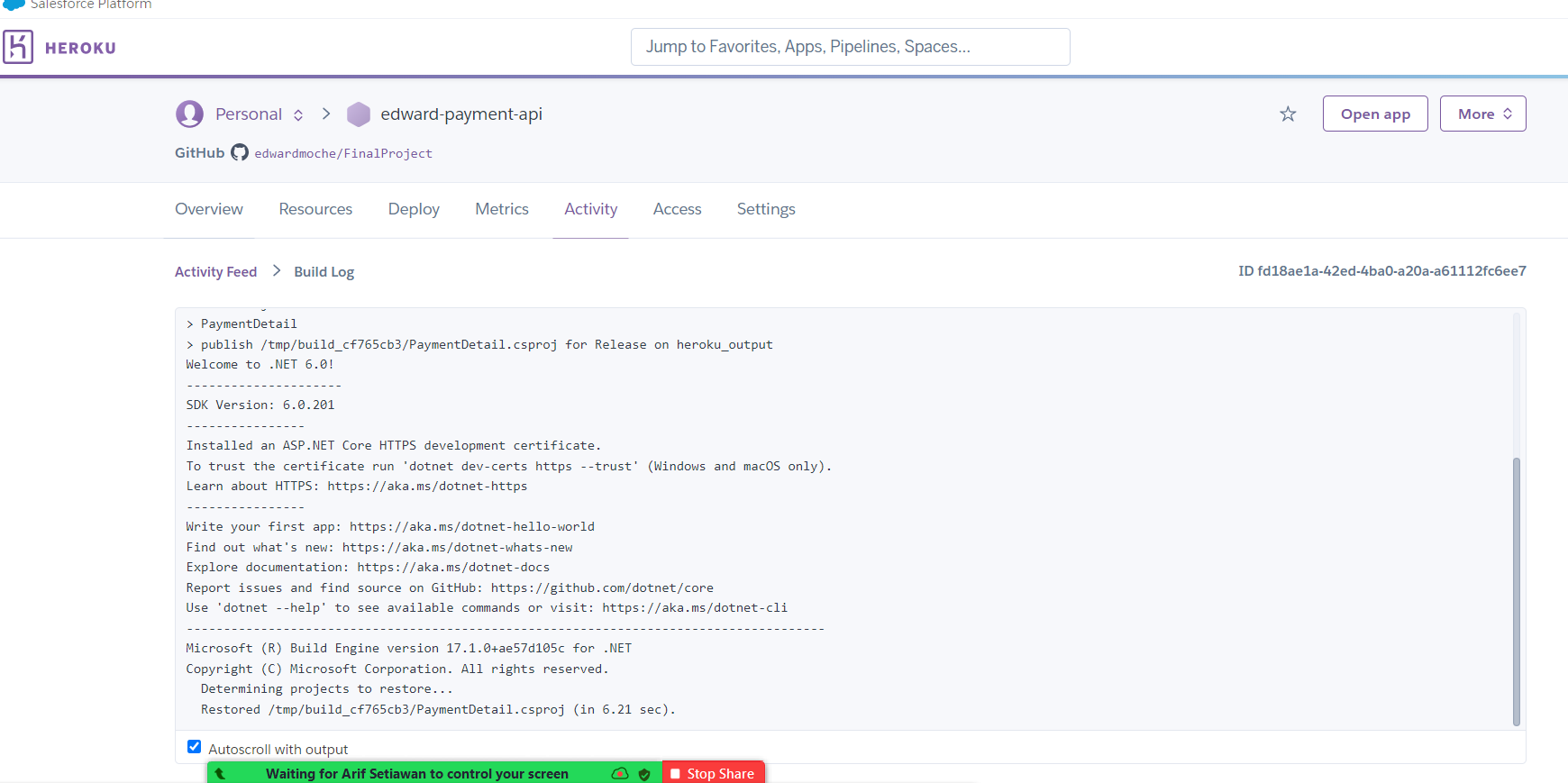


1. push ke heroku (syaratnya harus ada docker dan sudah di push ke git)

buat terlebih dahulu aplikasi (Edward-payment-api)

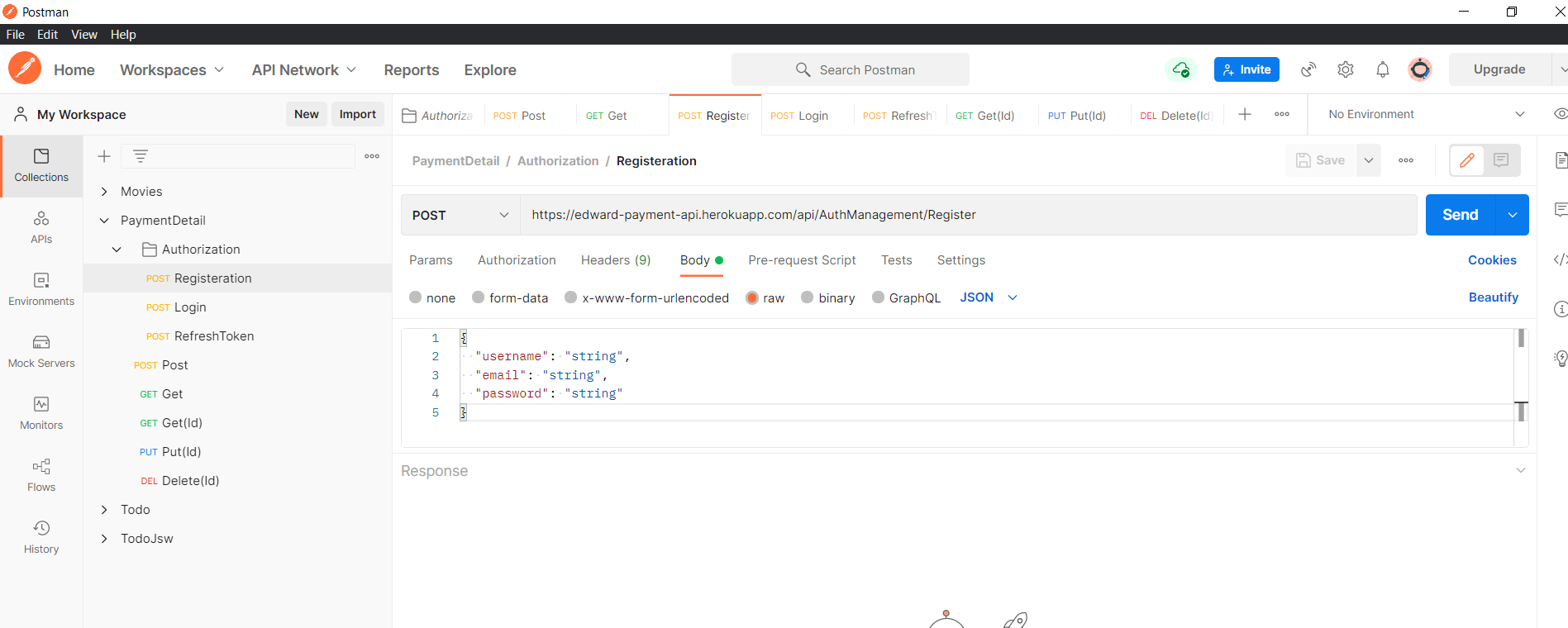
jika sudah masuk ke setting untuk menambahkan build yang dimana dapat didapatkan dari git jincod

kemudian kembali ke deploy, pilih git, lalu sambungkan dengan repo yang sudah kita buat sebelumnya

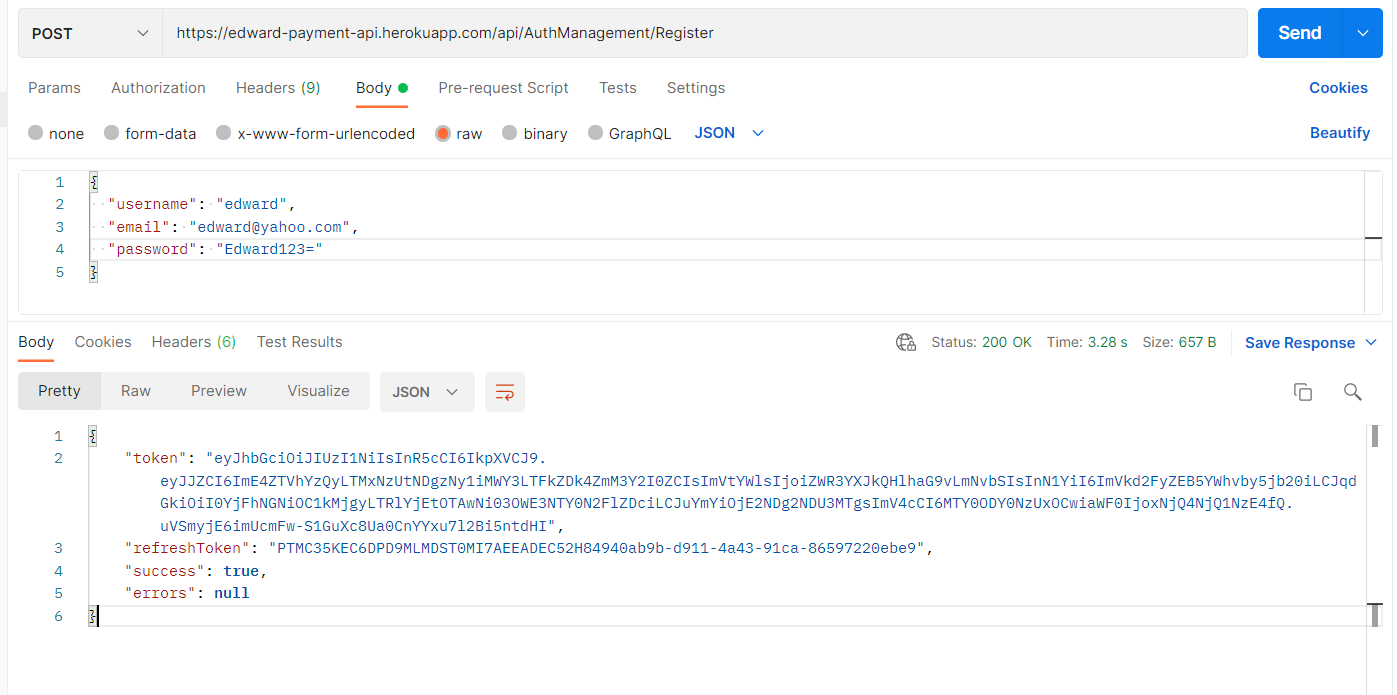


1. Penggunaan postman
   * + Register

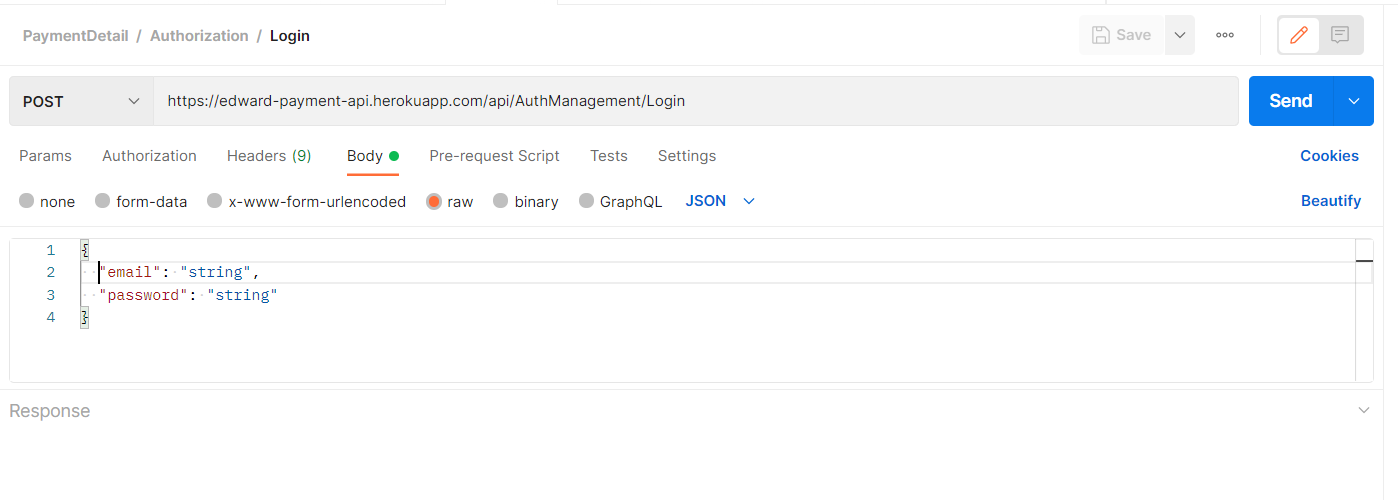
Sebelum, masukan link api yang kita buat, lalu gunakan fitur yang diinginkan (post), lalu buka body> raw > ubah text menjadi json > masukan script



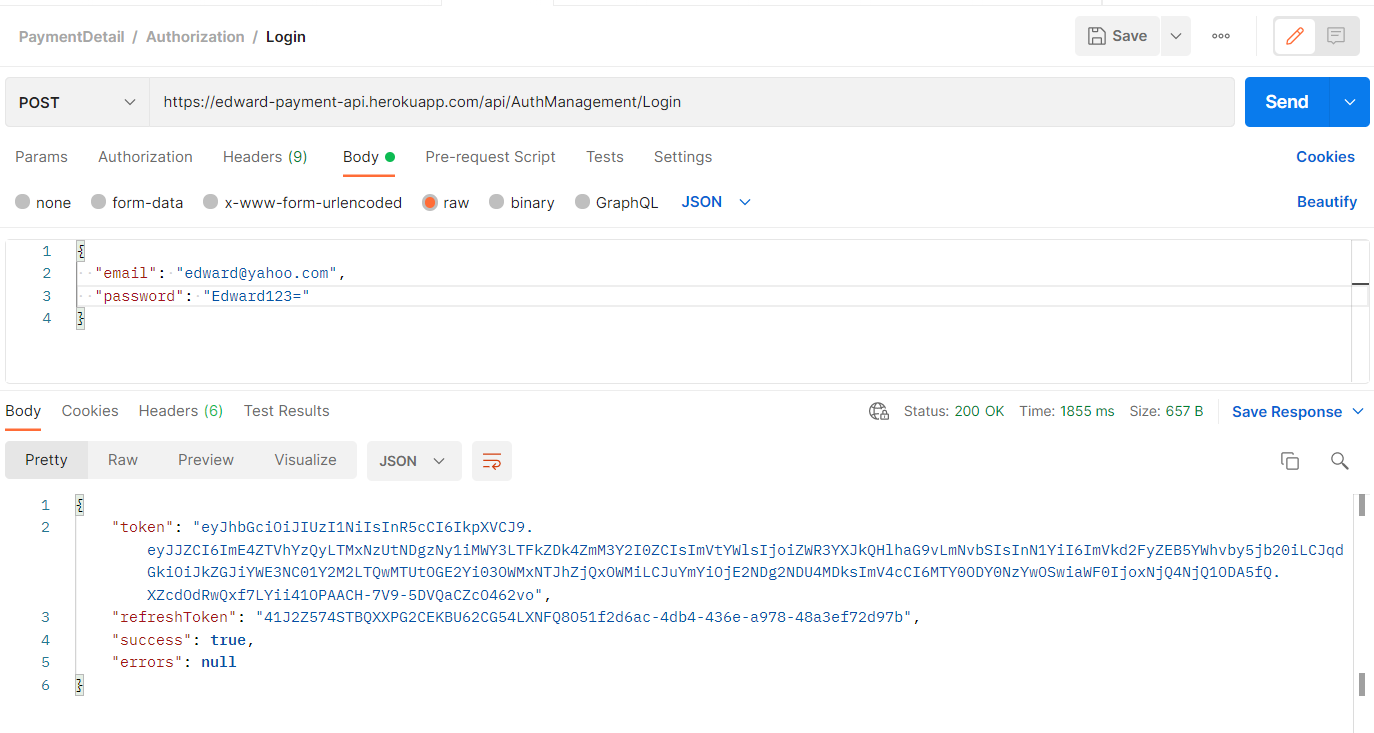
Sesudah kita masukan data, kita tekan send untuk menjalankan program



* + - Login

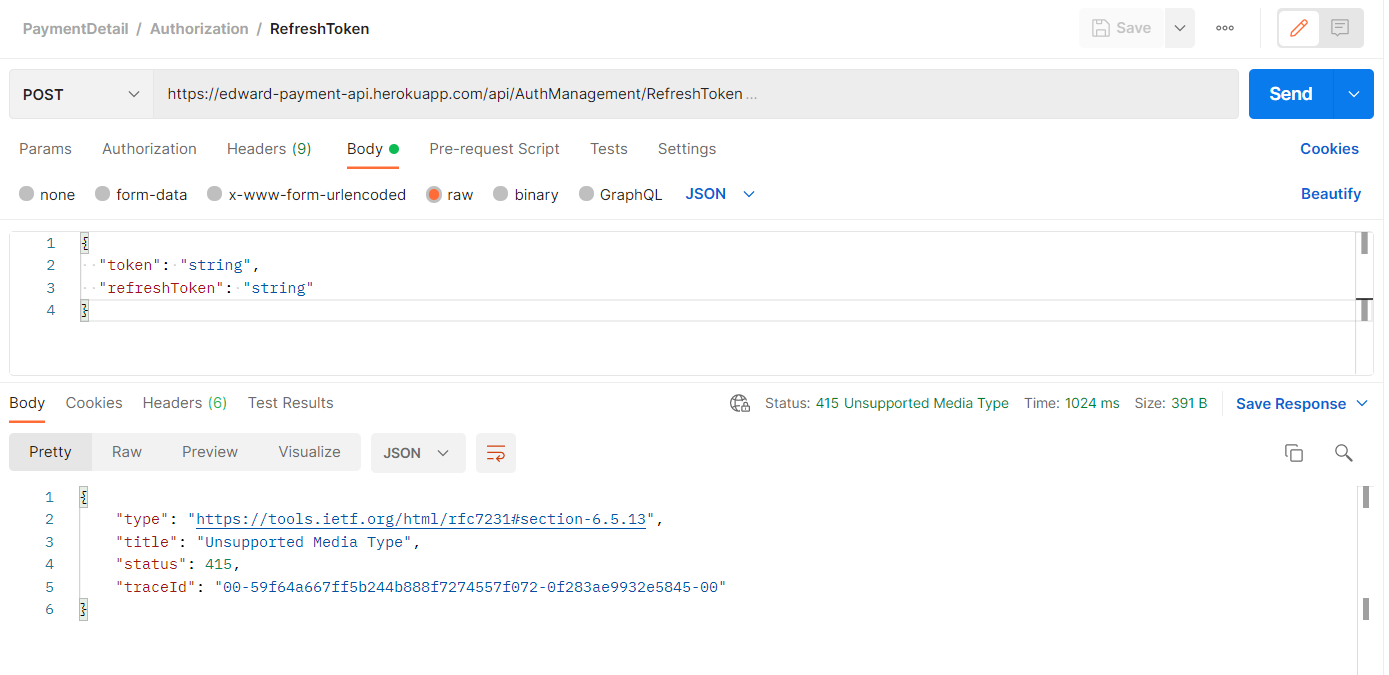
Sebelum, masukan link api yang kita buat, lalu gunakan fitur yang diinginkan (post), lalu buka body> raw > ubah text menjadi json > masukan script

Sesudah kita masukan data, kita tekan send untuk menjalankan program

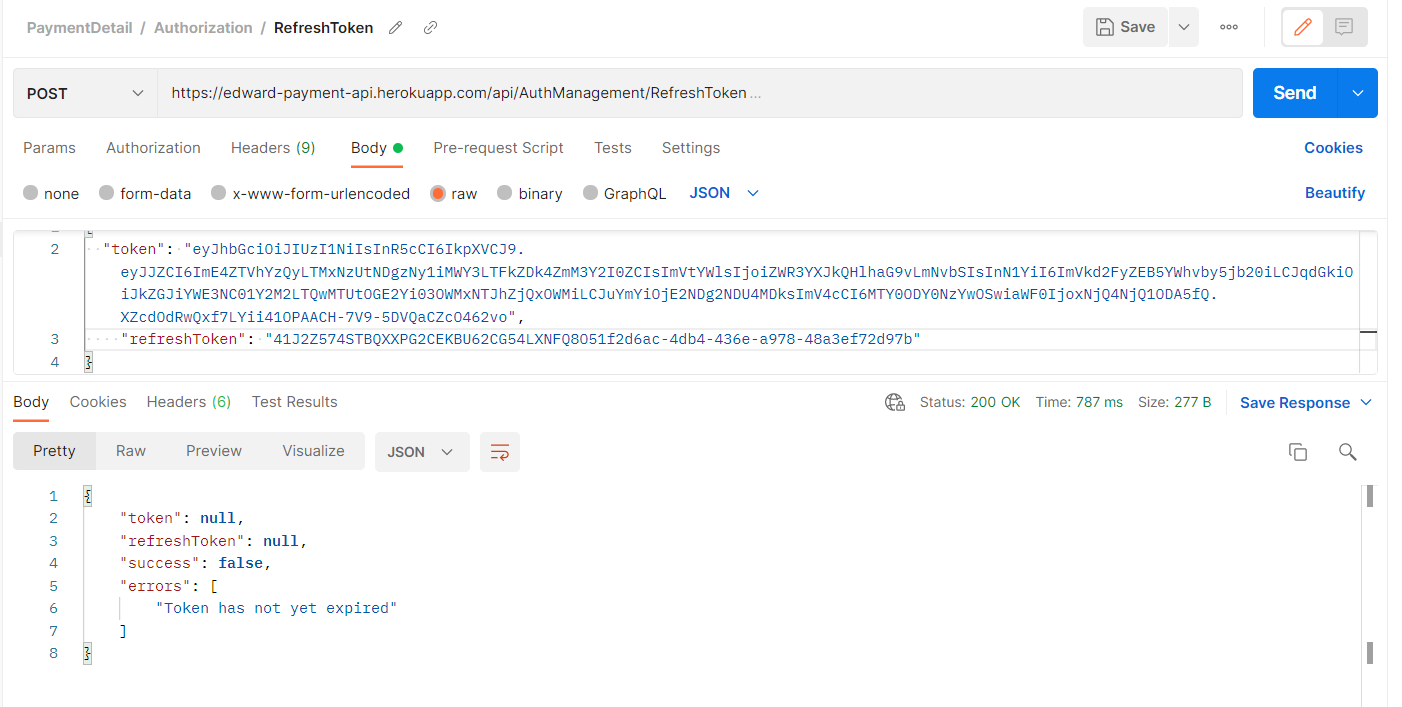


* + - refresh token

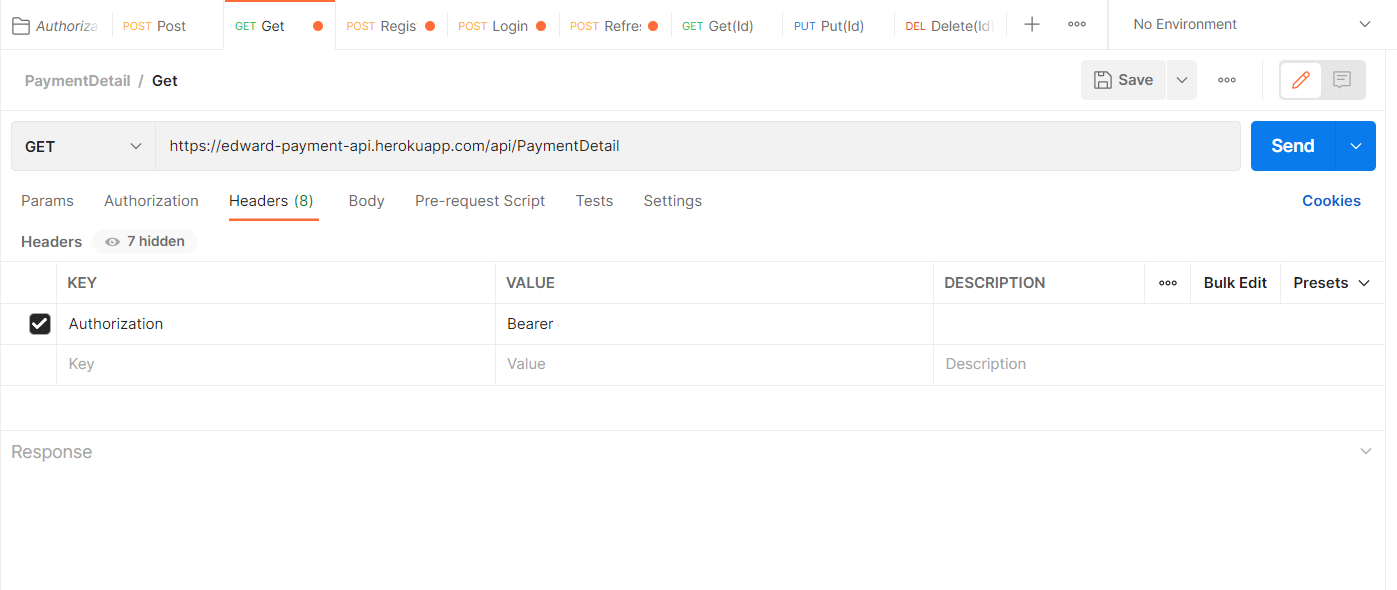
Sebelum, masukan link api yang kita buat, lalu gunakan fitur yang diinginkan (post), lalu buka body> raw > ubah text menjadi json > masukan script “token” dan “refresh” token dari login/register

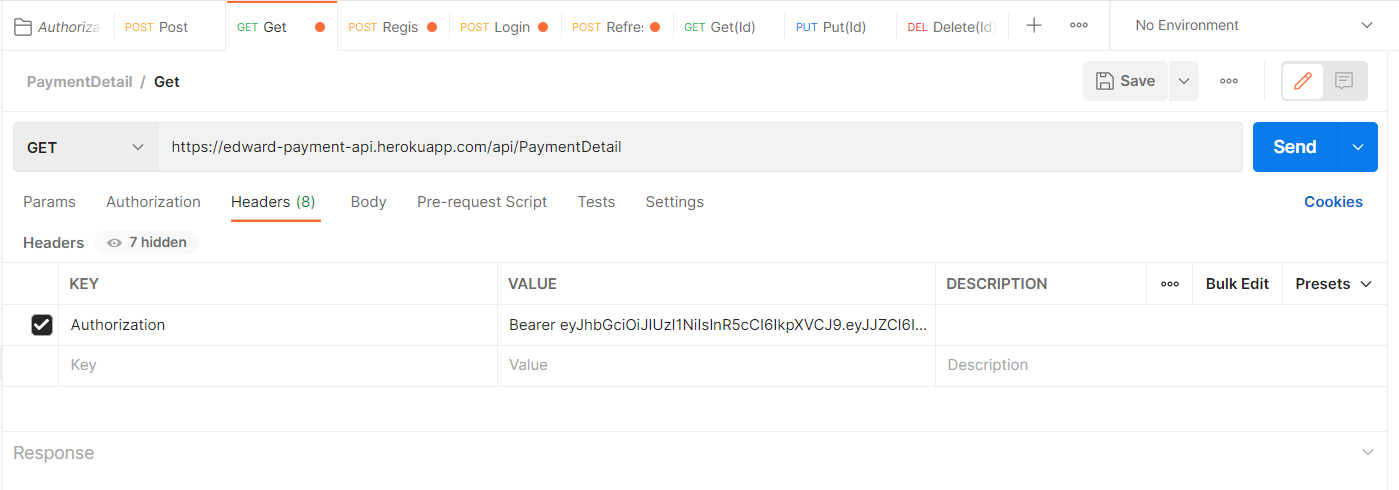


Sesudah kita masukan data, kita tekan send untuk menjalankan program maka akan tampil status dari token kita



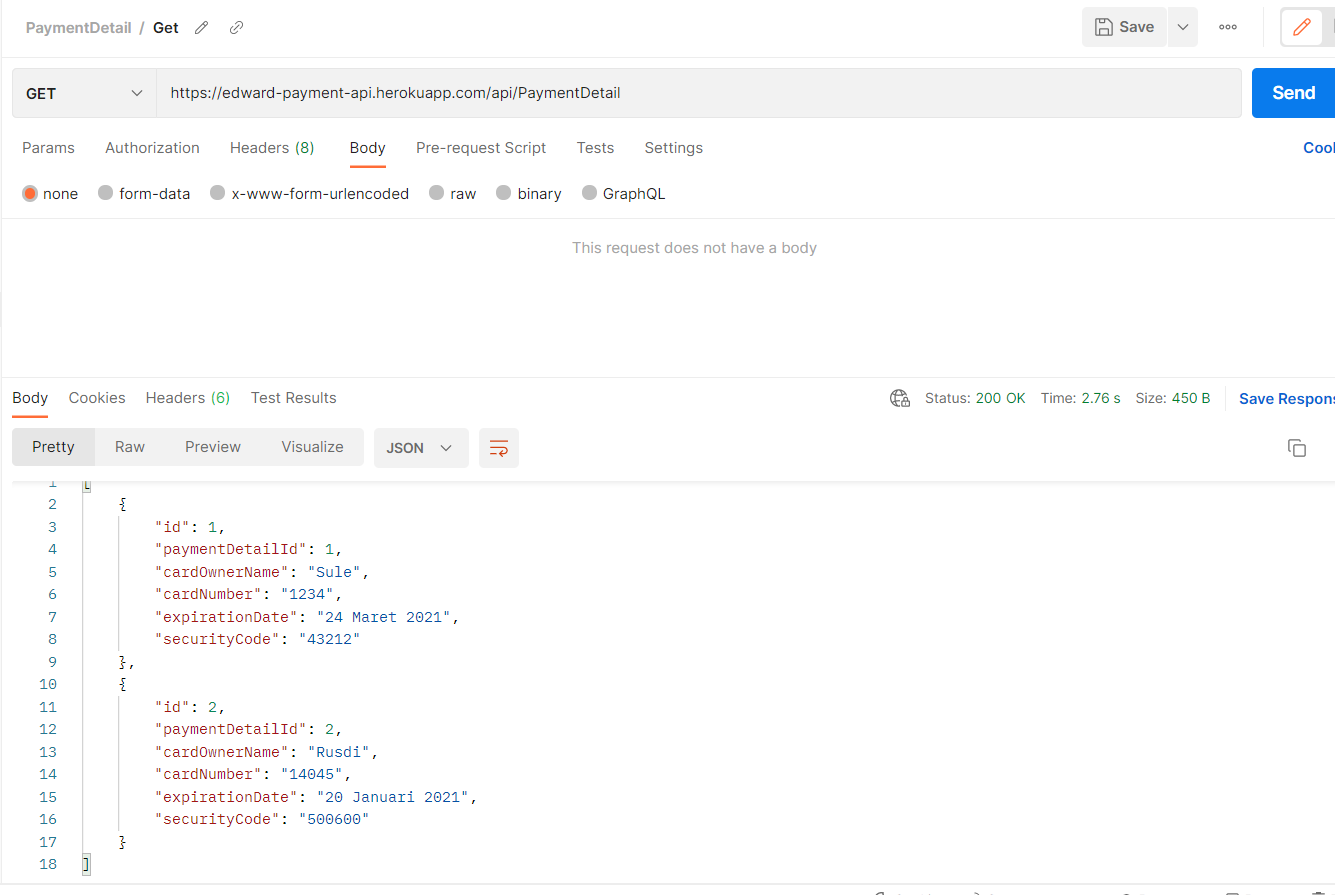
Untuk fitur kedepannya (get, post, put, get(id), dan delete) sebelum menjalankan terlebih dahulu tambahkan authorization pada headers tiap fungsi lalu masukan bearer (token) agar aplikasi dapat di akses





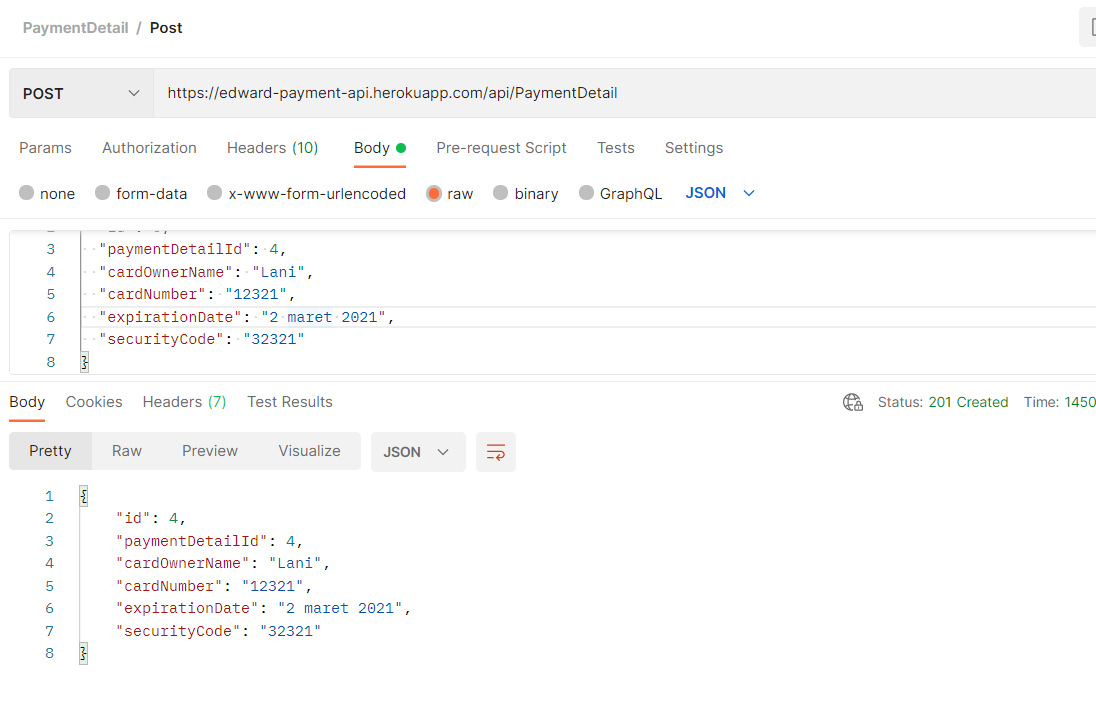
* + - get

langsung saja di run untuk melihat data



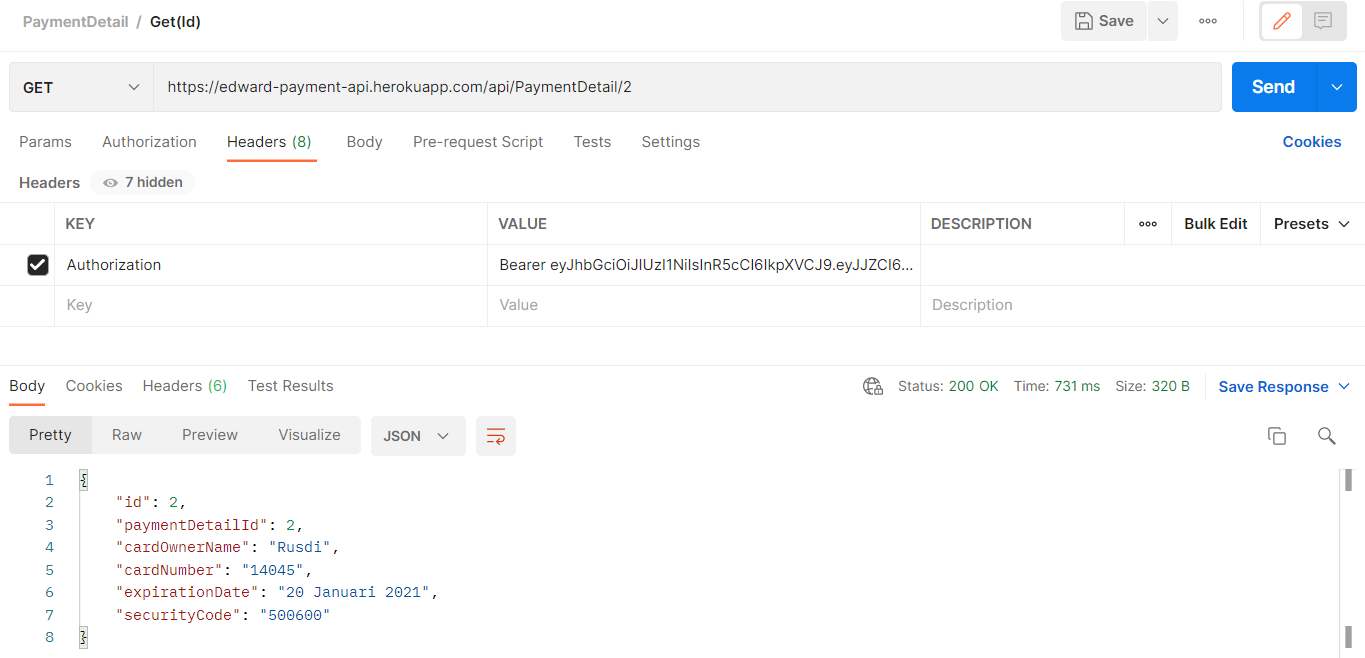
* + - post

Masukan parameter data yang ingin di buat lalu send

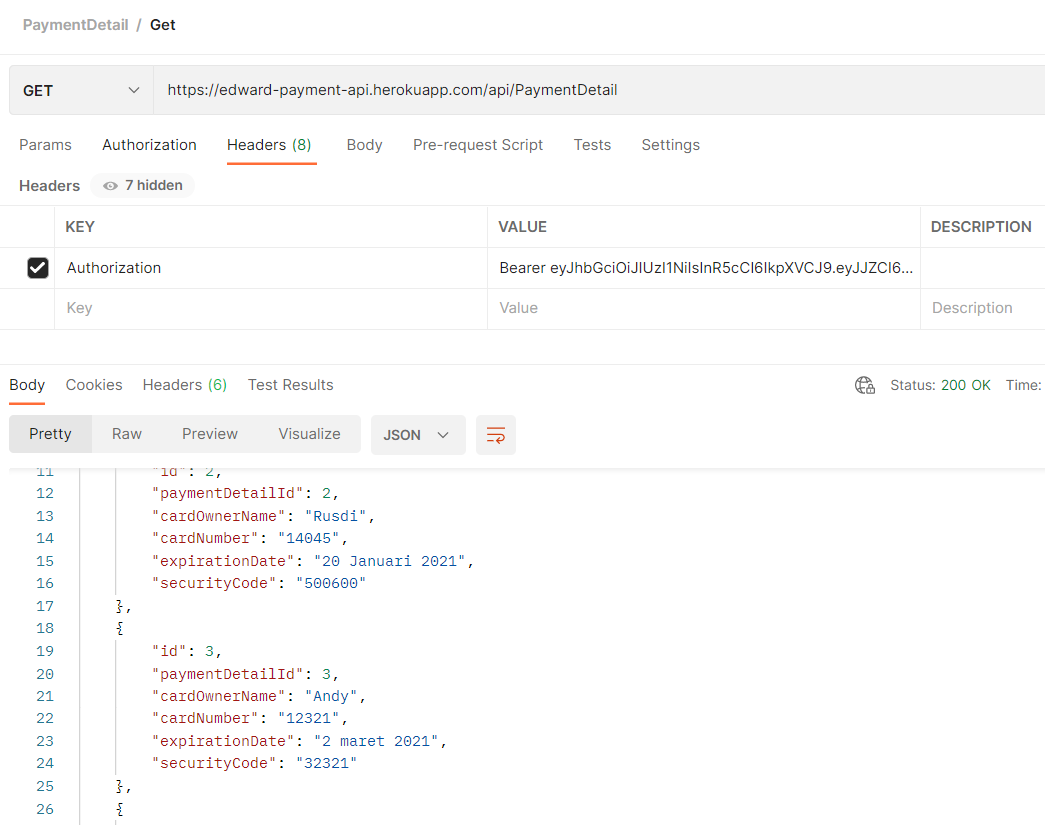


* + - get (Id)

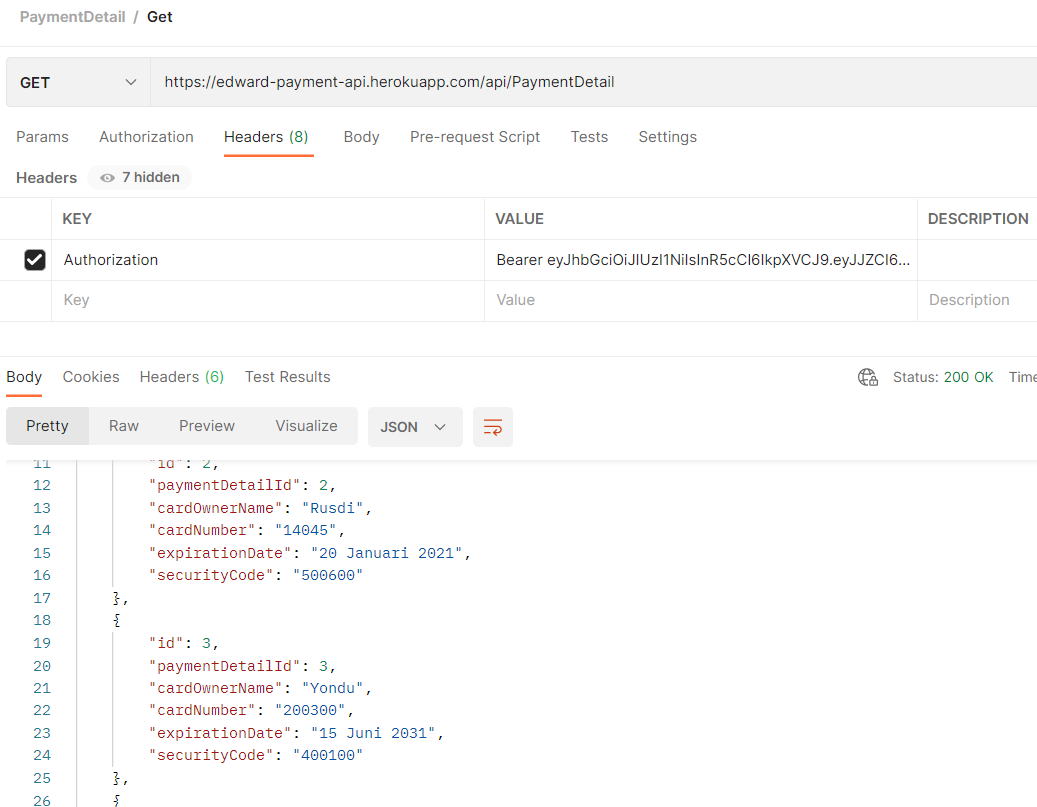
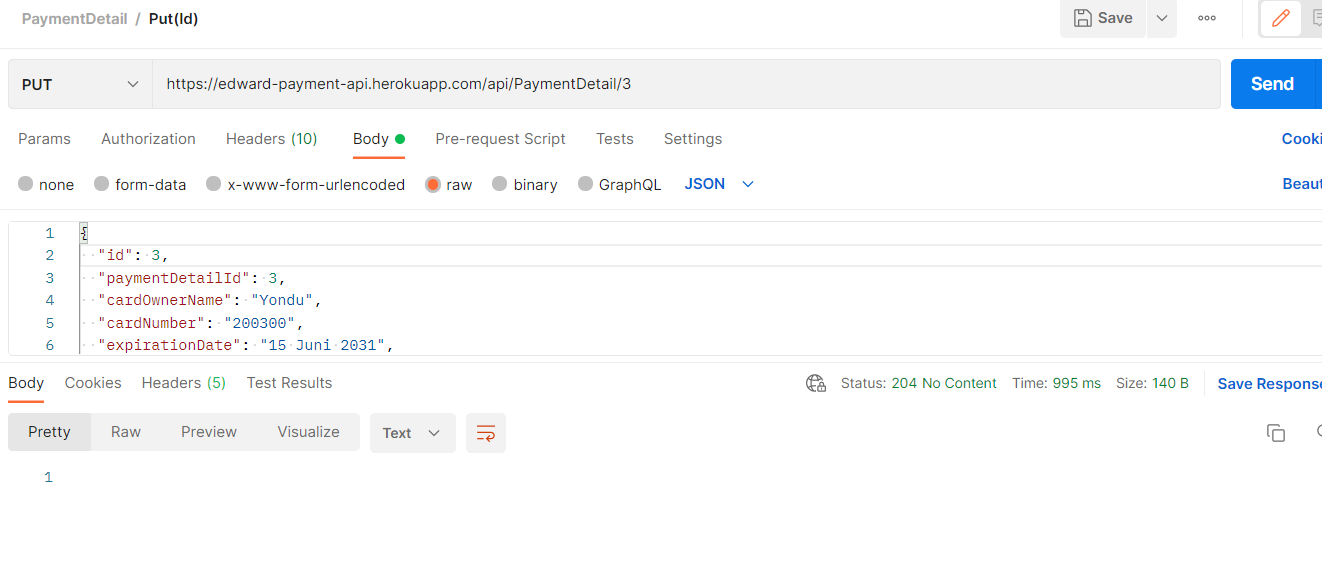
tambahkan nomer id pada link untuk menampilkan data yang spesifik lalu run



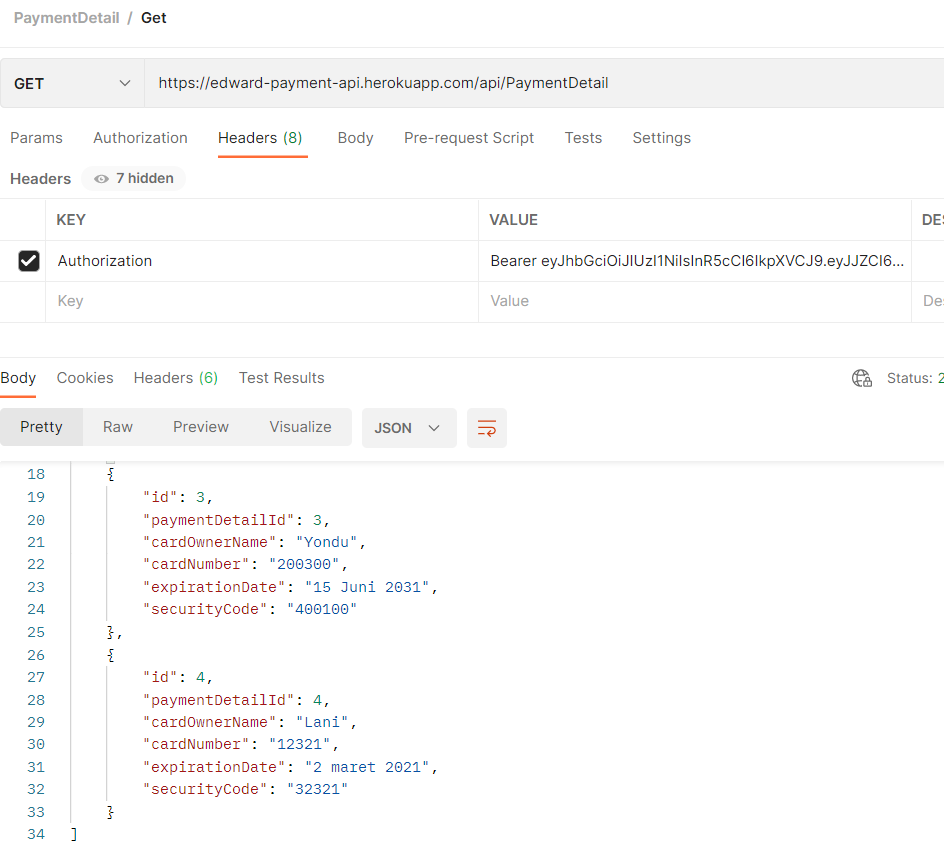
* + - put(Id)



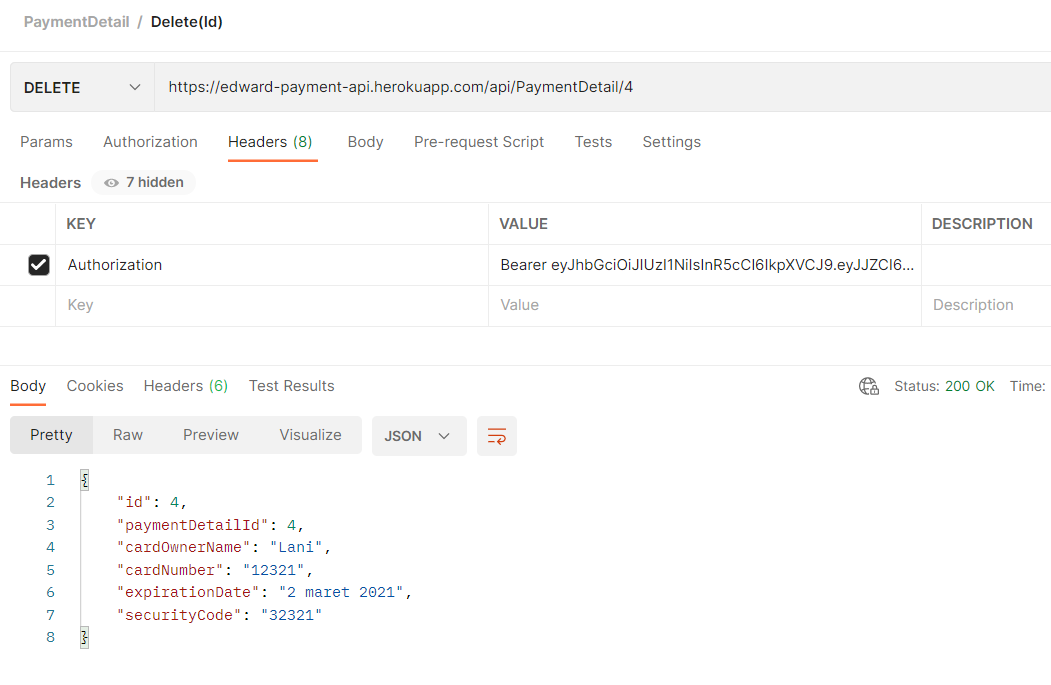
untuk put hampir sama seperti get(id) namun kita gunakan parameter seperti untuk push perbedaanya hanya di ubah pada bagian yang ingin diganti

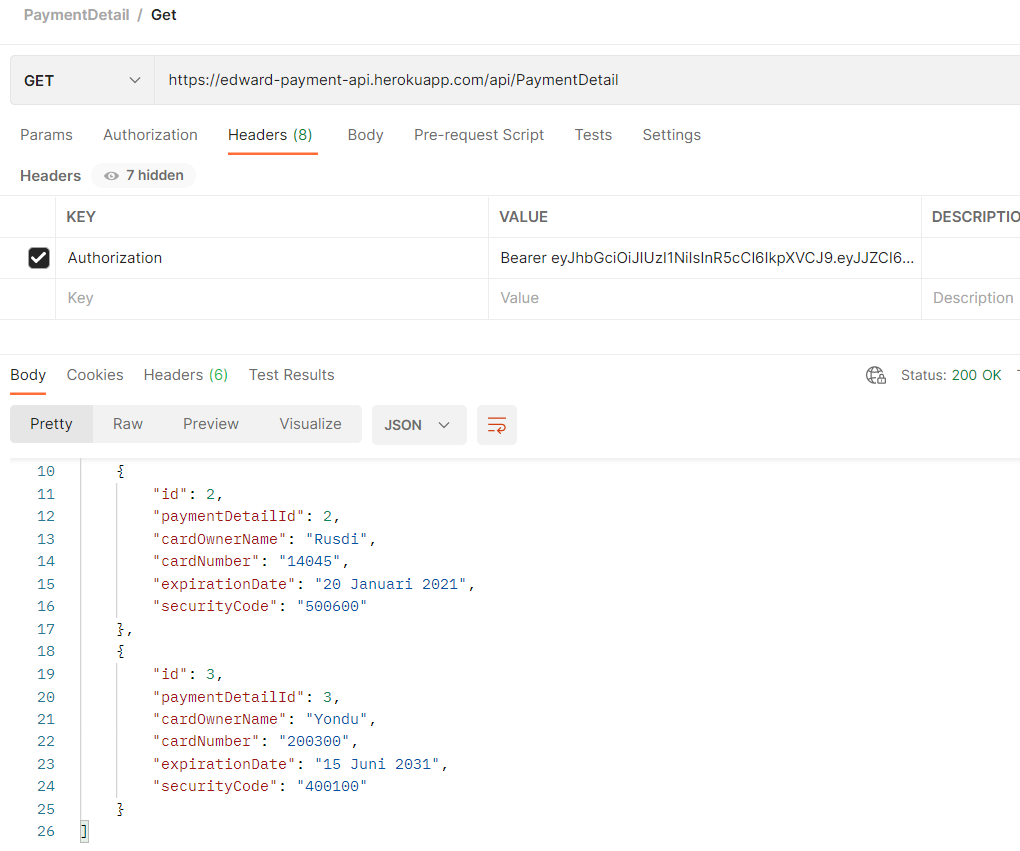


* + - delete



cara penggunaan sama seperit get(id), hanya perlu menambahkan id yang ingin di apus pada link

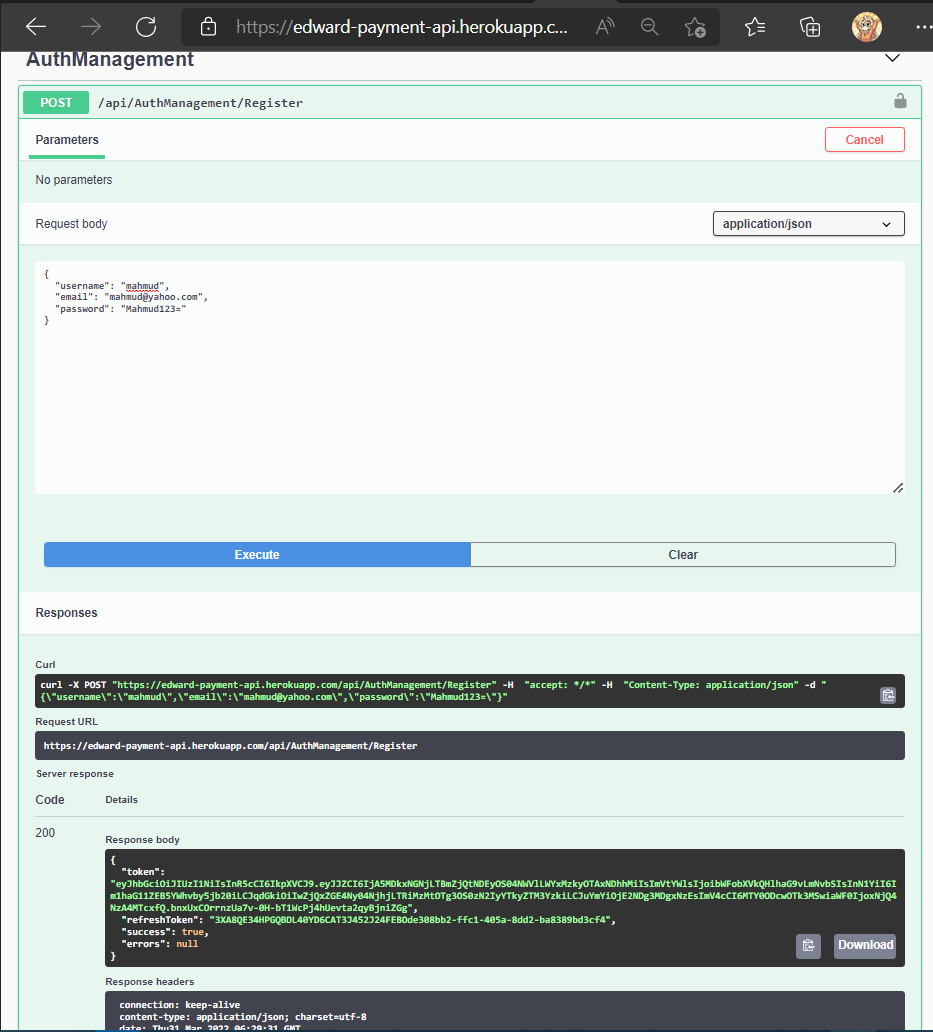




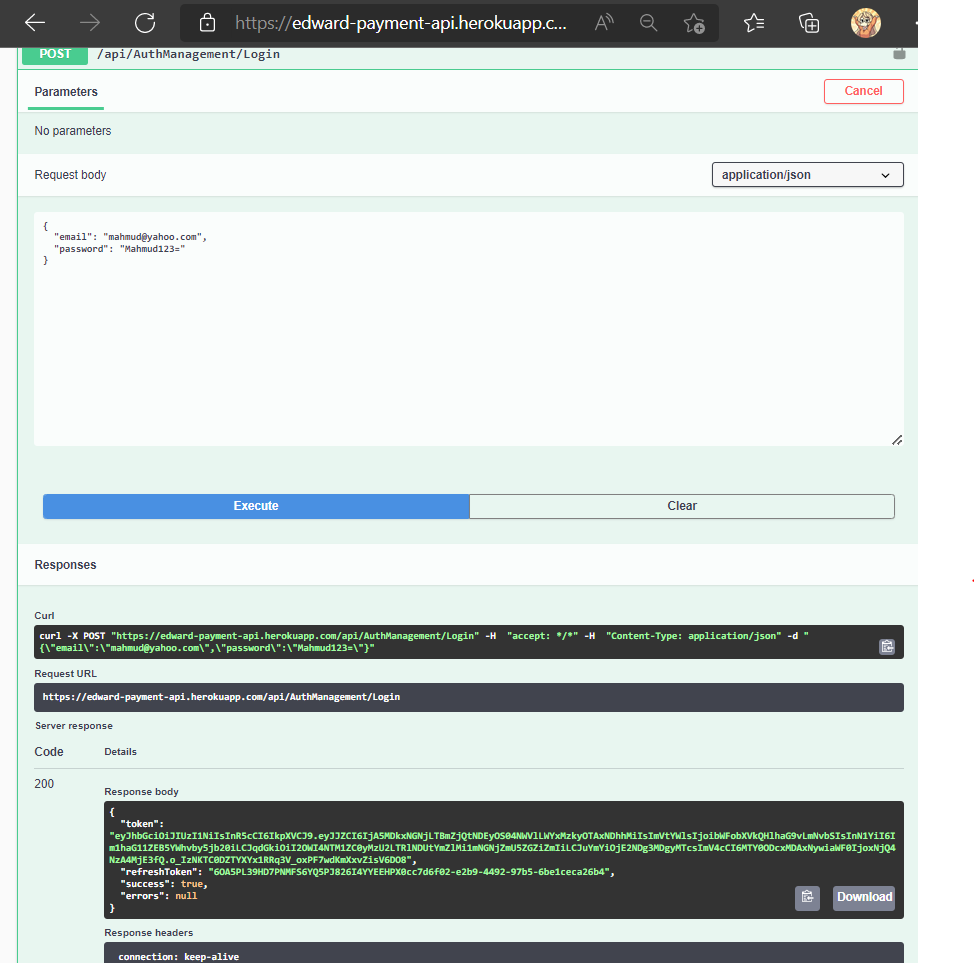
1. Penggunaan Swagger

Untuk swagger hampir sama persis seperti pada postman, perbedaanya untuk authorisasi hanya dilakukan sekali, yaitu pada button “Authorize”

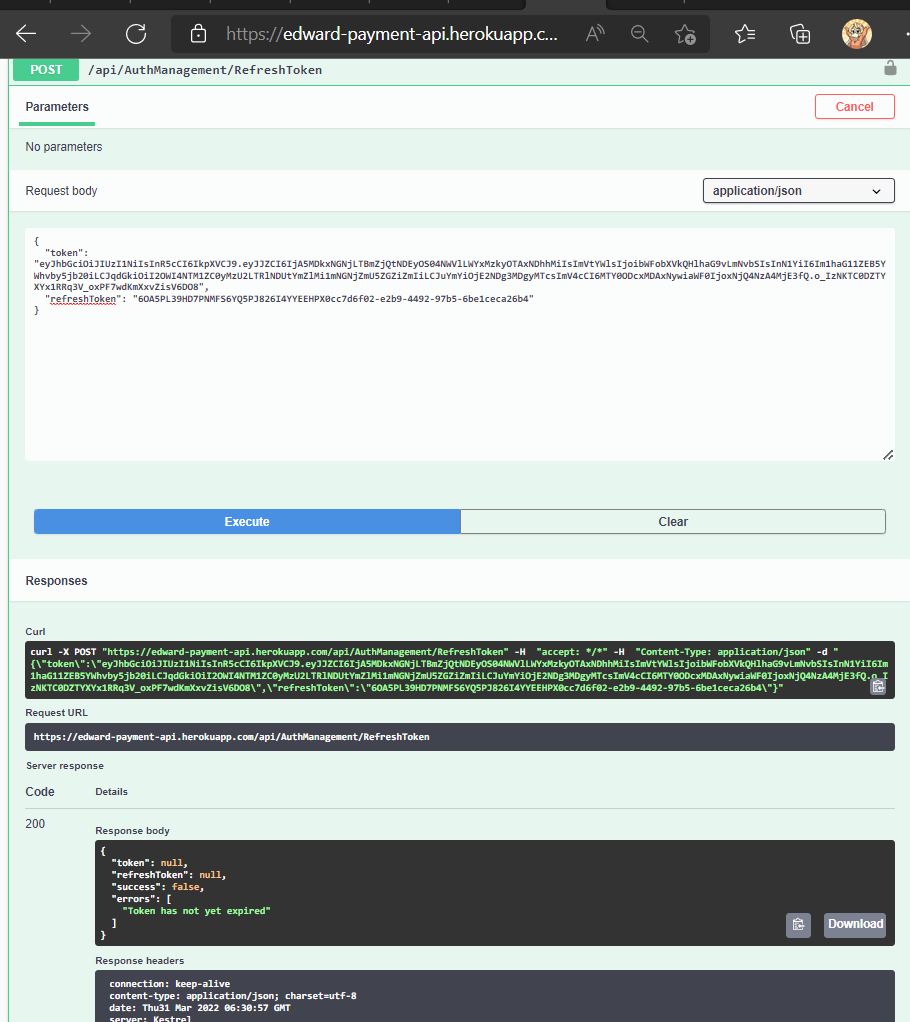
* Register



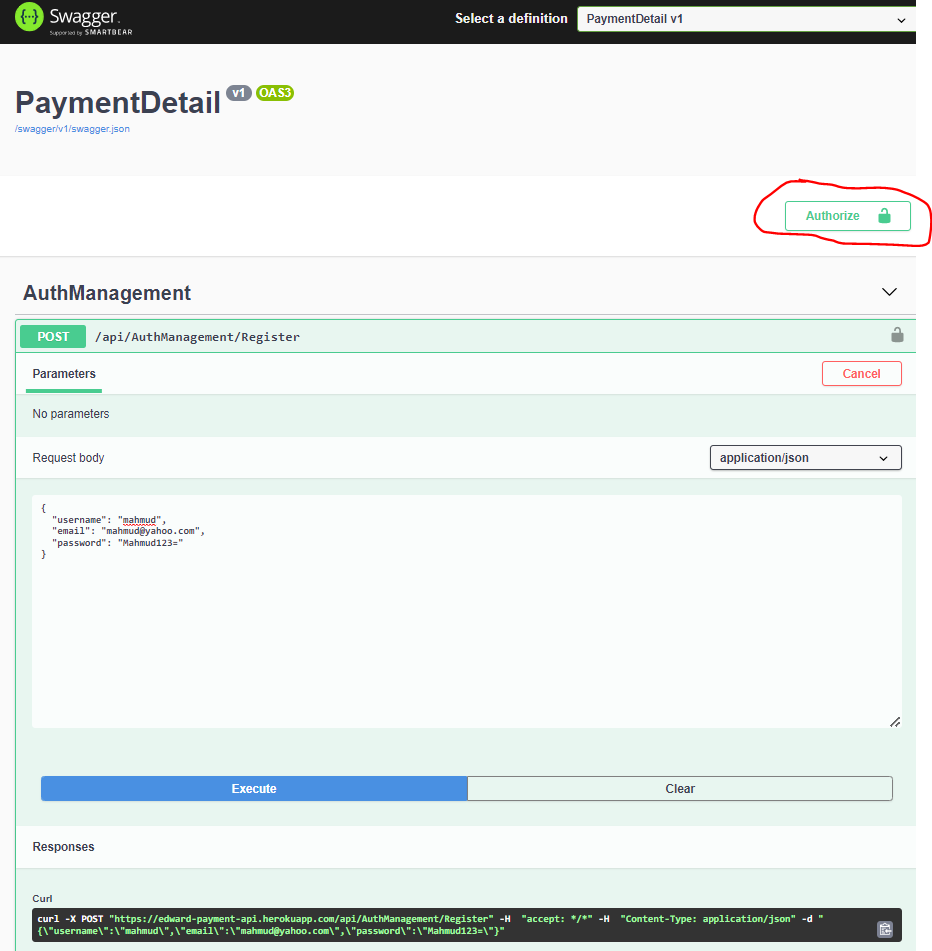
* Login

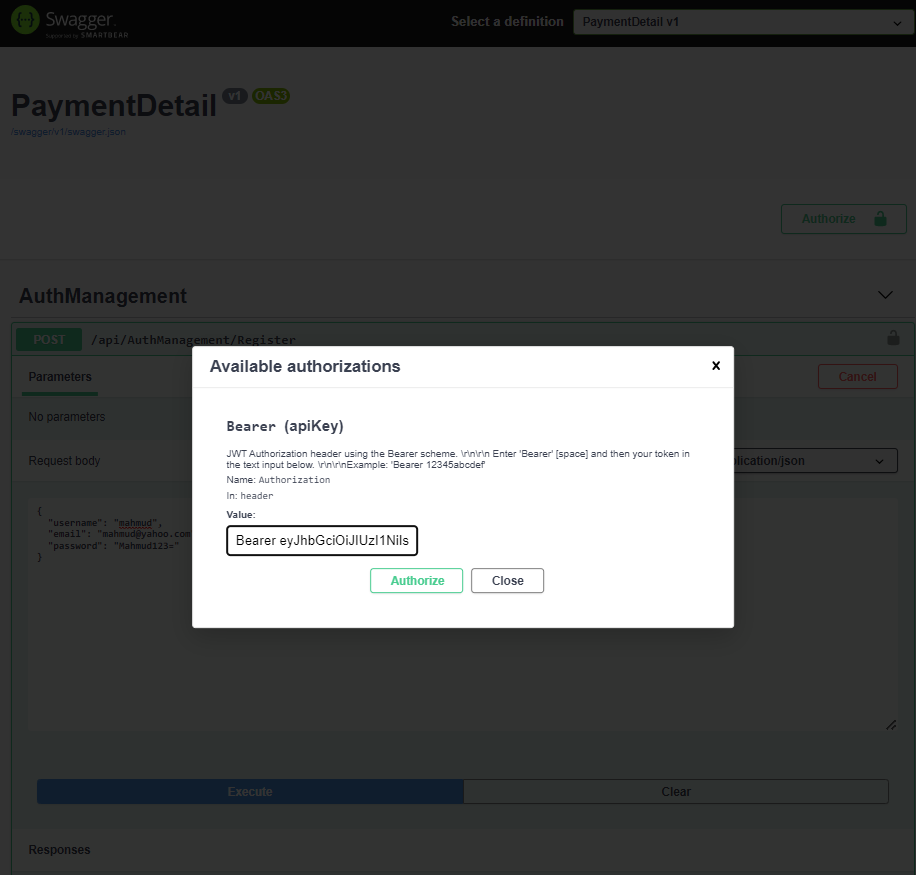


* Refresh token

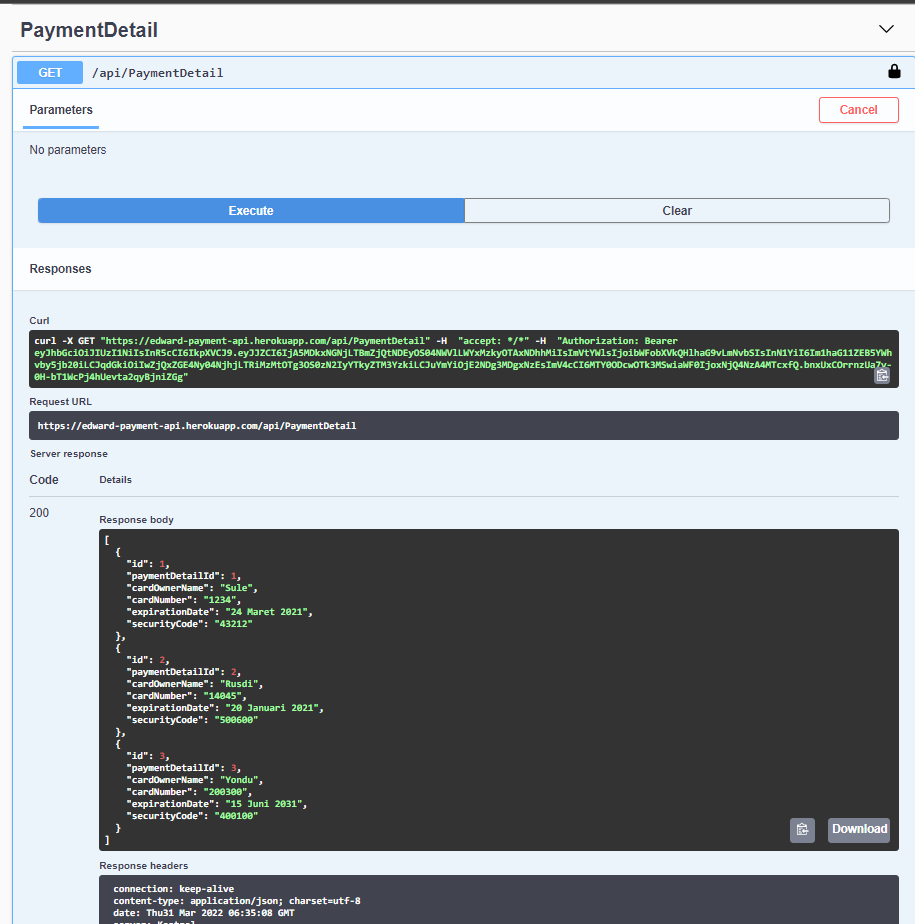


* Authorize

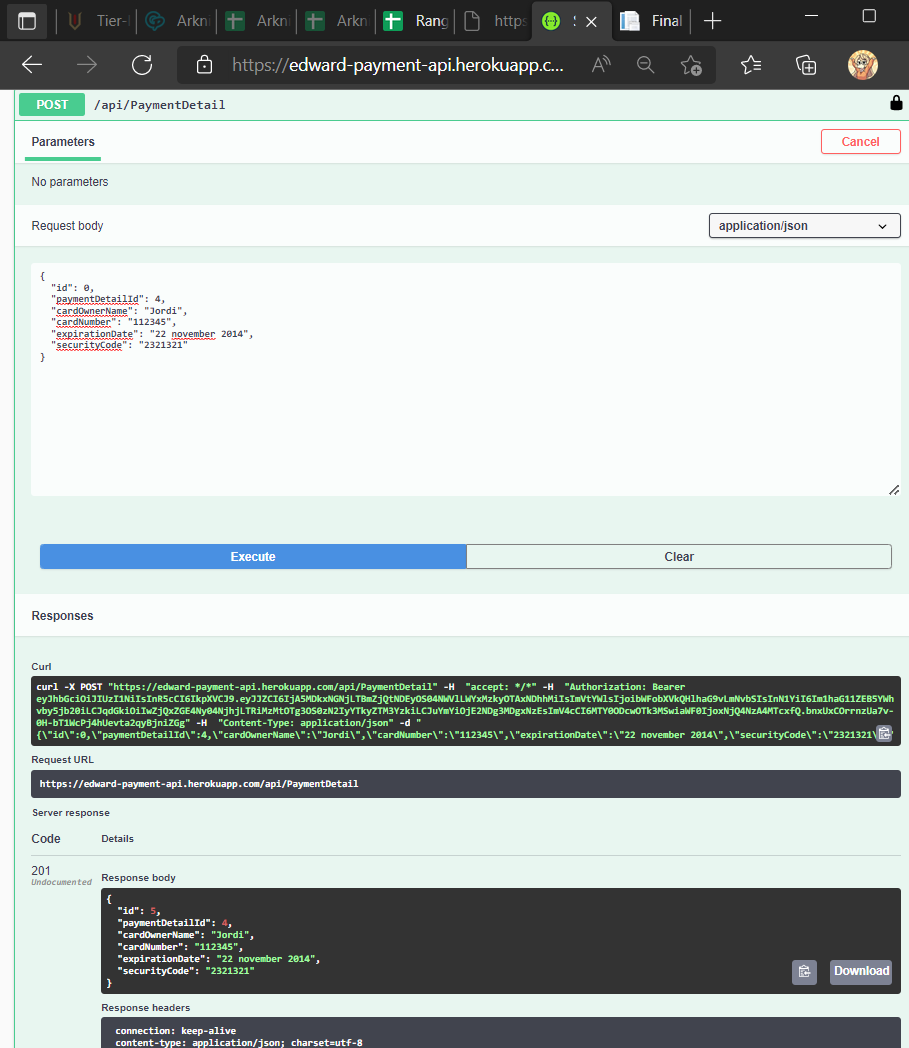




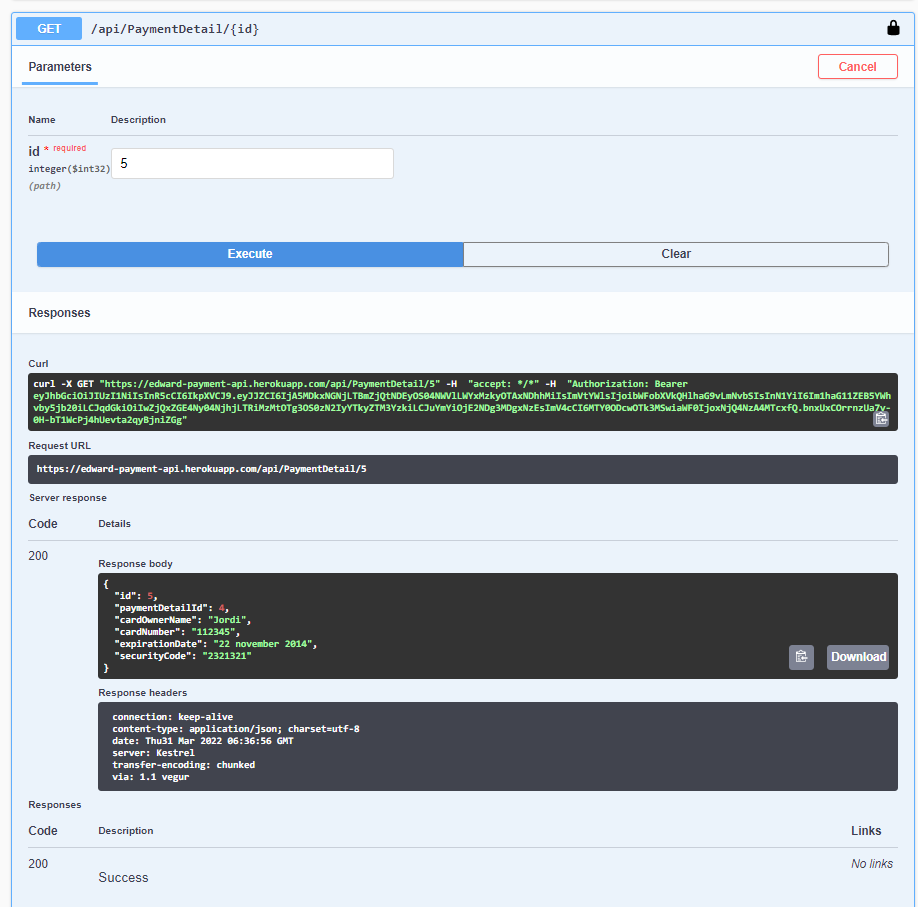
* Get



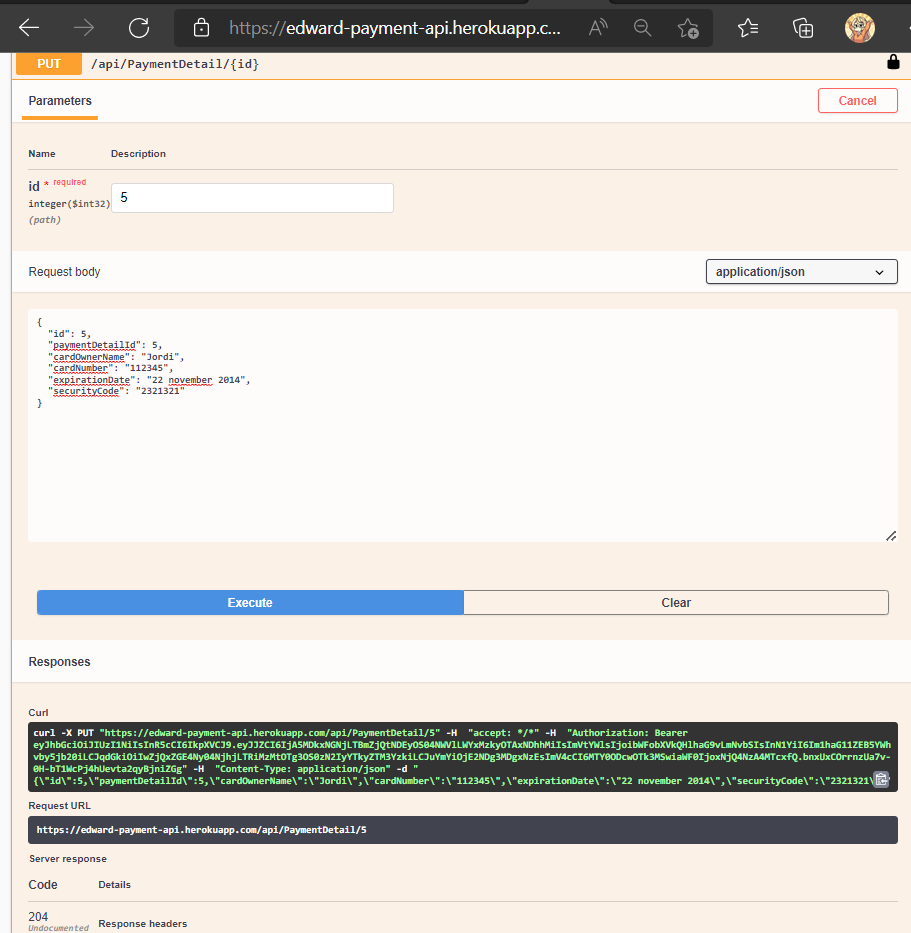
* Post



* Get(id)



* Put



* Delete

