Nama : Fatin Dzakiyah

Kode : FSDO003ONL002

**Final Project 1**

Membuat Payment API dengan ASP NET Core Web Api (NET 5.0) dan JWT Token serta menggunakan database MySql.

Link Heroku : <https://fatin-payment-app.herokuapp.com/>

Link Github : <https://github.com/f-dz/fatin_payment_app>

Database : CREATE DATABASE paymentdb;

Catatan : Untuk database localhost menggunakan file “(localhost) appsettings.json” yang ada di arsip project untuk konfigurasi

1. **Langkah Pembuatan Aplikasi Payment API ☑**
2. **Membuat project baru**

Buat folder untuk project baru lalu ketikan perintah pada console:

|  |
| --- |
| dotnet new webapi -n "PaymentApp" -lang "C#" -au none |

1. **Menambahkan library pada project**

NuGet.CommandLine.XPlat versi 5.11.0

Swashbuckle.AspNetCore versi 5.6.3

Pomelo.EntityFrameworkCore.Mysql versi 5.0.0-alpha.2

Microsoft.AspNetCore.Authentication.JwtBearer versi 5.0.5

Microsoft.AspNetCore.Identity.EntityFrameworkCore versi 5.0.6

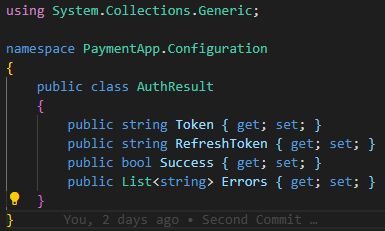
Microsoft.AspNetCore.Identity.UI versi 5.0.0

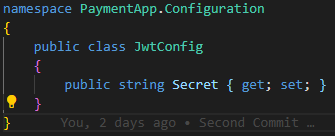
Microsoft.EntityFrameworkCore.Design versi 5.0.11

Microsoft.EntityFrameworkCore.Tools versi 5.0.11

1. **Membuat class untuk konfigurasi**

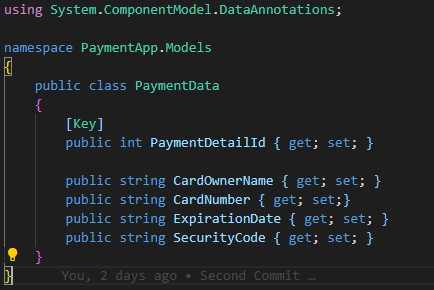
Class AuthResult berisi method untuk get set Token, RefreshToken, Success, Errors.



Class JwtConfig yang berisi method untuk get set Secret. 

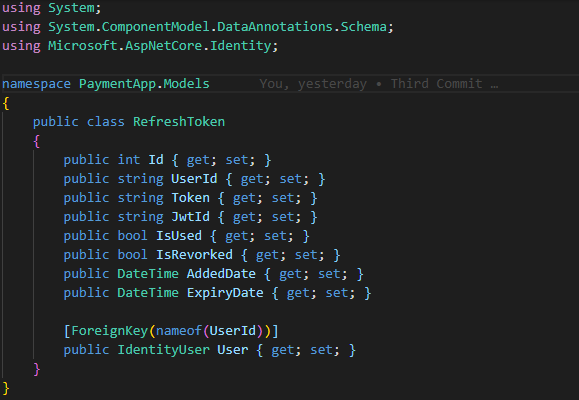
1. **Membuat class dan method untuk get set data payment**

Membuat class PaymentData yang berisi method untuk get set PaymentDetailId sebagai primary key, CardOwnerName, CardNumber, ExpirationDate, dan SecurityCode.

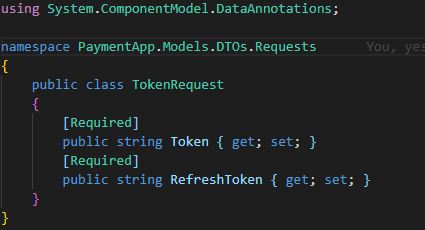


1. **Membuat class dan method untuk get set data refresh token**

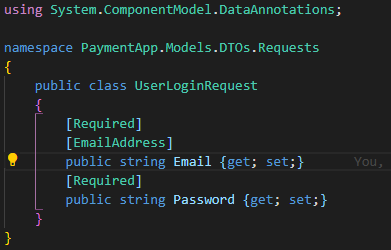
Membuat class RefreshToken yang berisi method untuk get set untuk keperluan refresh token ketika token yang digunakan untuk autorisasi sudah expired.



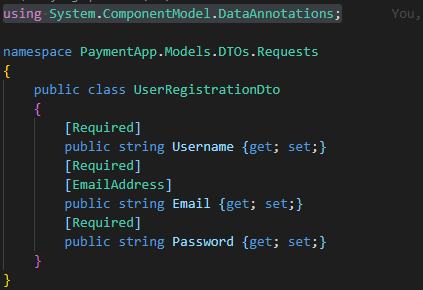
1. **Membuat class untuk melakukan request**

Membuat class untuk menangani permintaan refresh token

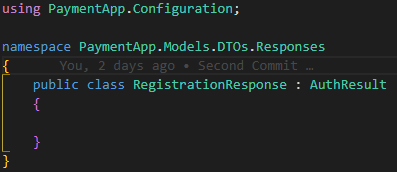
Membuat class untuk menangani permintaan login



Membuat class untuk menangani permintaan register



1. **Membuat class untuk menerima response**



1. **Membuat controller untuk CRUD autentikasi (AuthManagementController)**

Membuat method-method untuk CRUD auntentikasi. Berikut nama-nama method yang dibuat, yaitu Register, Login, RefreshToken, GenearteJwtToken, VerifyAndGenerateToken, RandomString, dan UnixTimeStampToDateTime :















Menambahkan JWT dengan token time 30 menit dengan set waktu expired selama 30 menit pada fungsi GenerateJwtToken di AuthManagementController :



1. **Membuat controller untuk CRUD data payment (PaymentController)**

Membuat method untuk mendapatkan data payment, membuat detail payment baru, mendapatkan payment dengan id tertentu, mengedit data payment, dan menghapus data payment :





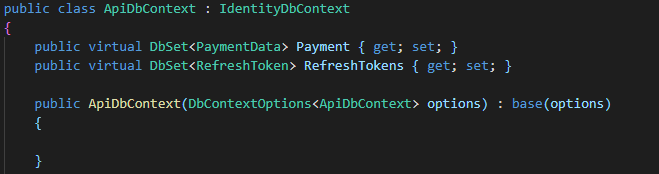






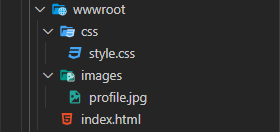
1. **Membuat class database yang berisi method untuk get set dari class payment dan refresh token yang sudah dibuat**

Membuat class ApiDbContext yang berisi class virtual dari Payment dan RefreshToken dan menuruni atribut dan method dari class IdentityUser :



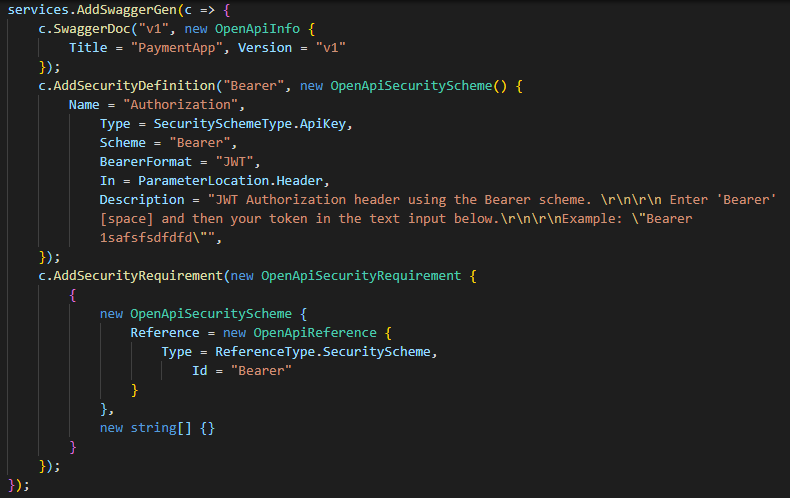
1. **Membuat landing page**

Pada project PaymentApp, buat folder “wwwroot” yang berisi file index.html sebagai landing page serta file css dan image yang diperlukan.

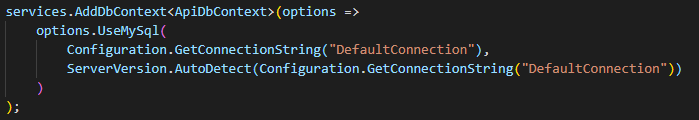


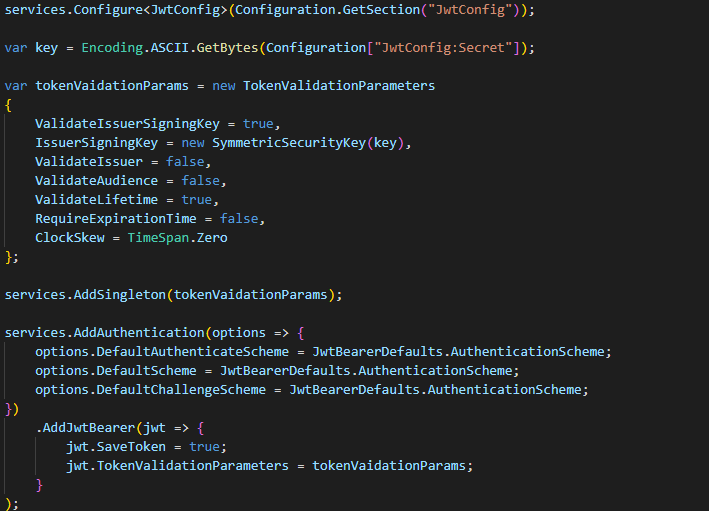
1. **Mengatur service dan middleware yang digunakan pada file Startup.cs**

Menambahkan service AddSwagger dengan authorisasi :



Menambahkan services untuk konfigurasi penggunaan database MySql dan menambahkan koneksi yang sudah diatur pada file appsetings.json :



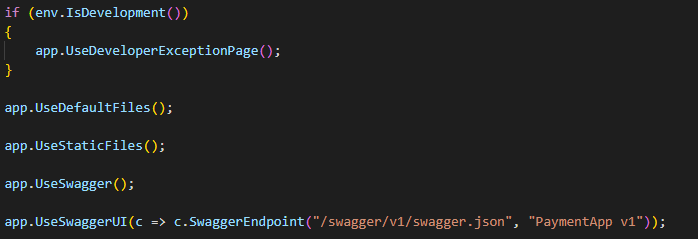
Menambahkan services untuk konfigurasi Jason Web Token : 

Menambahkan services konfigurasi IdentityUser ketika mengakses ApiDbContext :

Menambahkan middleware untuk mengakses default file yang dibaca ketika aplikasi dijalankan. Dengan menambahkan middleware ini, maka file pertama yang dibaca adalah file index.html pada folder wwwroot :



Merubah middleware swagger yang digunakan sebagai front-end aplikasi yang semula di dalam fase development, menjadi di luar fase development. Artinya ketika aplikasi di deploy maka pengguna dapat mengakses halaman swagger tersebut.



Menambahkan middleware untuk melakukan autentikasi :



1. **Mengatur konfigurasi pada file appsettings .json**

Menambahkan pengaturan mengenasi server, port, database, username, dan password yang digunakan :



Menambahkan pengaturan menganai secret key yang digunakan :



1. **Membuat file docker**

Membuat docker file dengan nama “Dockerfile”

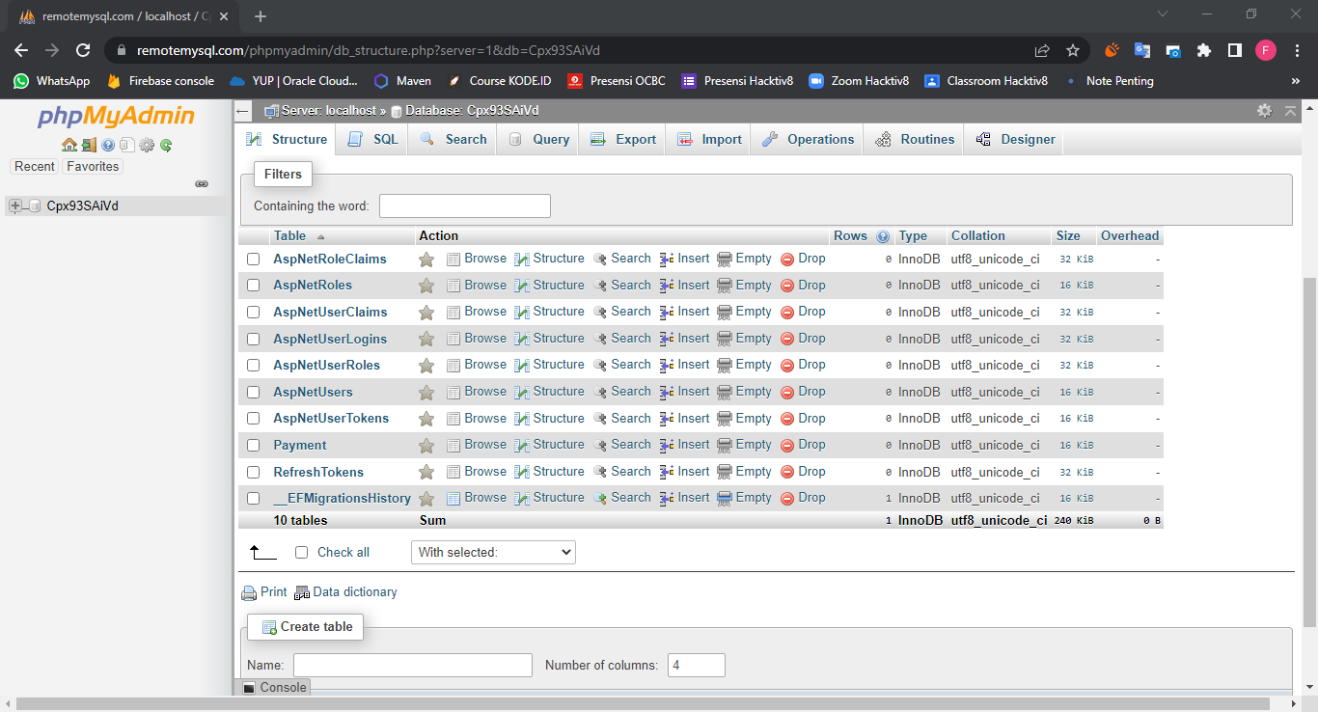
|  |
| --- |
| FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base  WORKDIR /app  EXPOSE 80  EXPOSE 443  EXPOSE 48719  EXPOSE 27017  EXPOSE 5000  EXPOSE 5001  FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build  WORKDIR /src  COPY . .  #COPY ["PaymentApp/PaymentApp.csproj", "PaymentApp/"]  #RUN dotnet restore "PaymentApp/PaymentApp.csproj"  #COPY . .  #WORKDIR "/src/PaymentApp"  #RUN dotnet build "PaymentApp.csproj" -c Release -o /app/build  RUN dotnet restore  RUN dotnet build --no-restore -c Release -o /app  FROM build AS publish  RUN dotnet publish "PaymentApp.csproj" -c Release -o /app/publish  FROM base AS final  WORKDIR /app  #COPY --from=publish /app/publish .  COPY --from=publish /app/ .  #ENTRYPOINT ["dotnet", "PaymentApp.dll"]  CMD ASPNETCORE\_URLS=http://\*:$PORT dotnet PaymentApp.dll |

1. **Migrasi database MySql ke server remotemysql.com ☑**

Perintah yang diketikan pada console untuk migrasi database :

|  |
| --- |
| dotnet ef migrations add "Initial Migration"  dotnet ef database update |

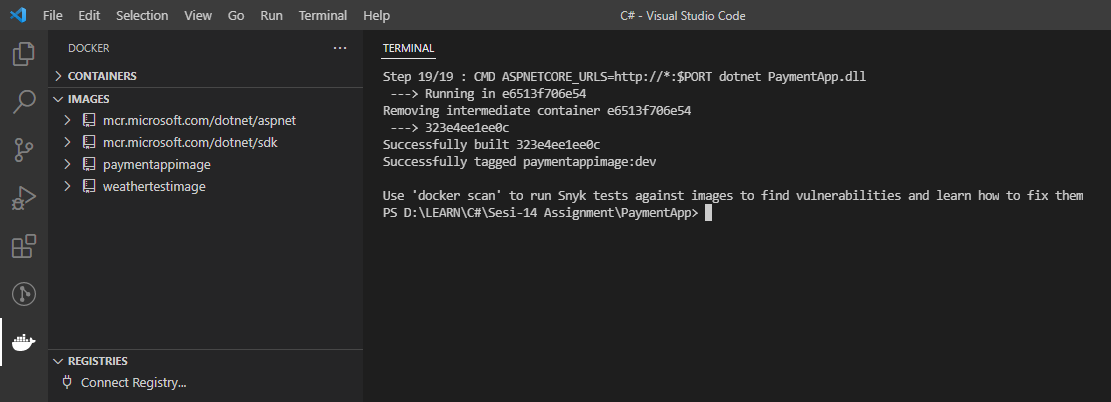
Database yang digunakan adalah database MySql dengan menggunakan server <https://remotemysql.com>. Tampilan database setelah dilakukan migrasi.



1. **Build Docker ☑**

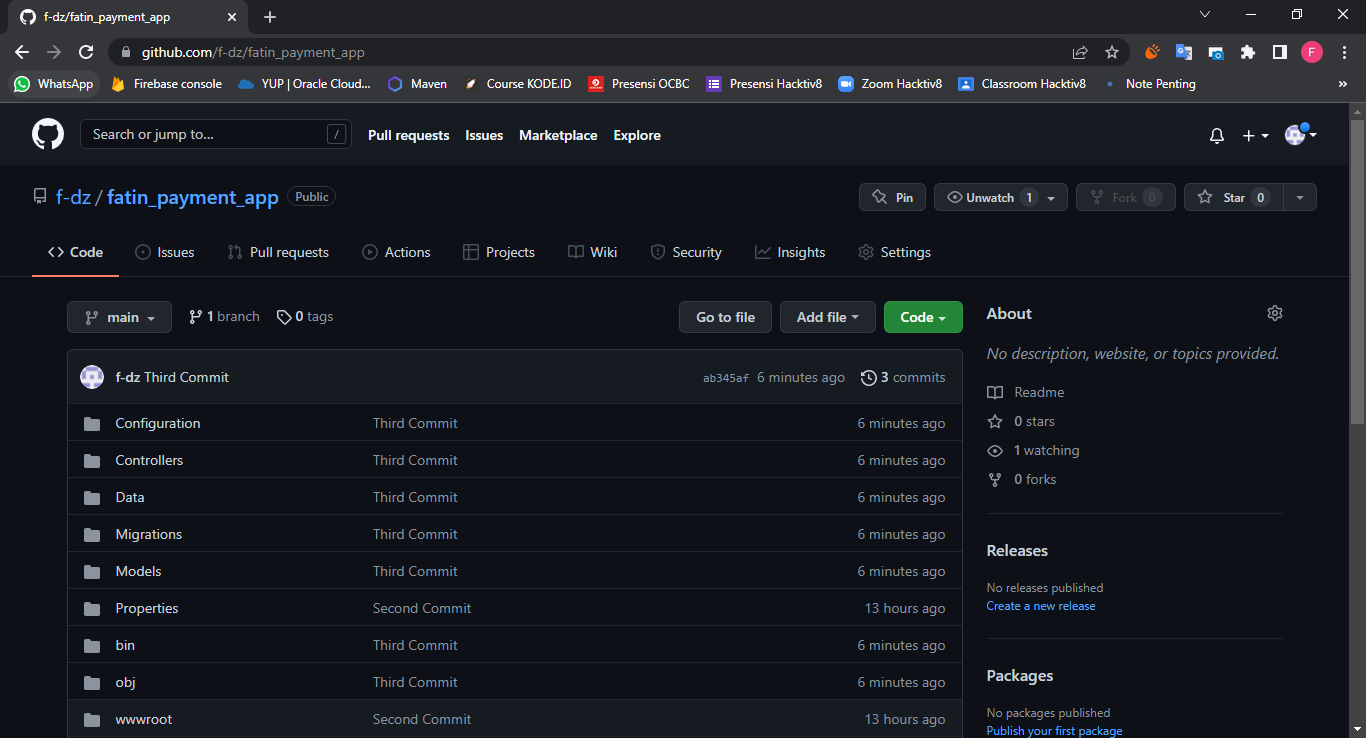
|  |
| --- |
| dotnet build  dotnet publish -c Release  docker build -t paymentappimage:dev . |

Setelah proses build selesai, pada Docker terbentuk dokumen image baru dengan nama paymentappimage.

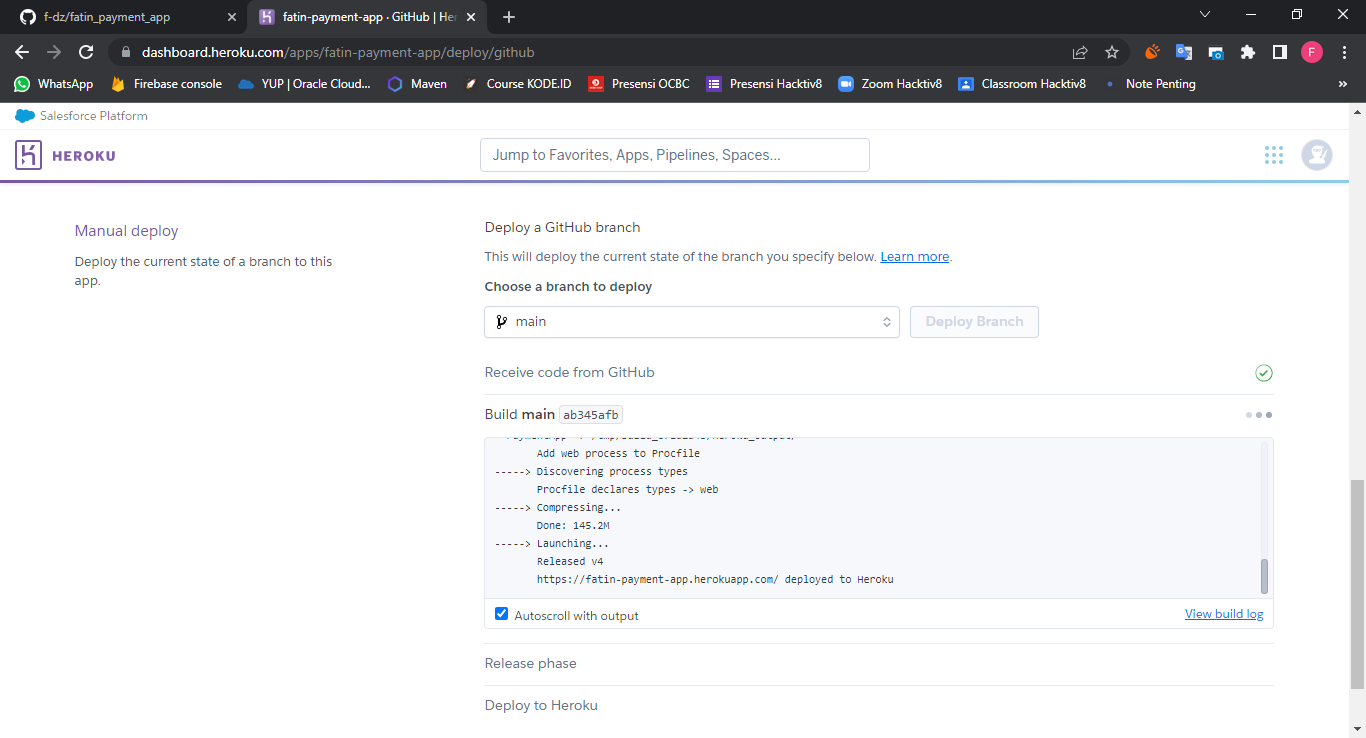


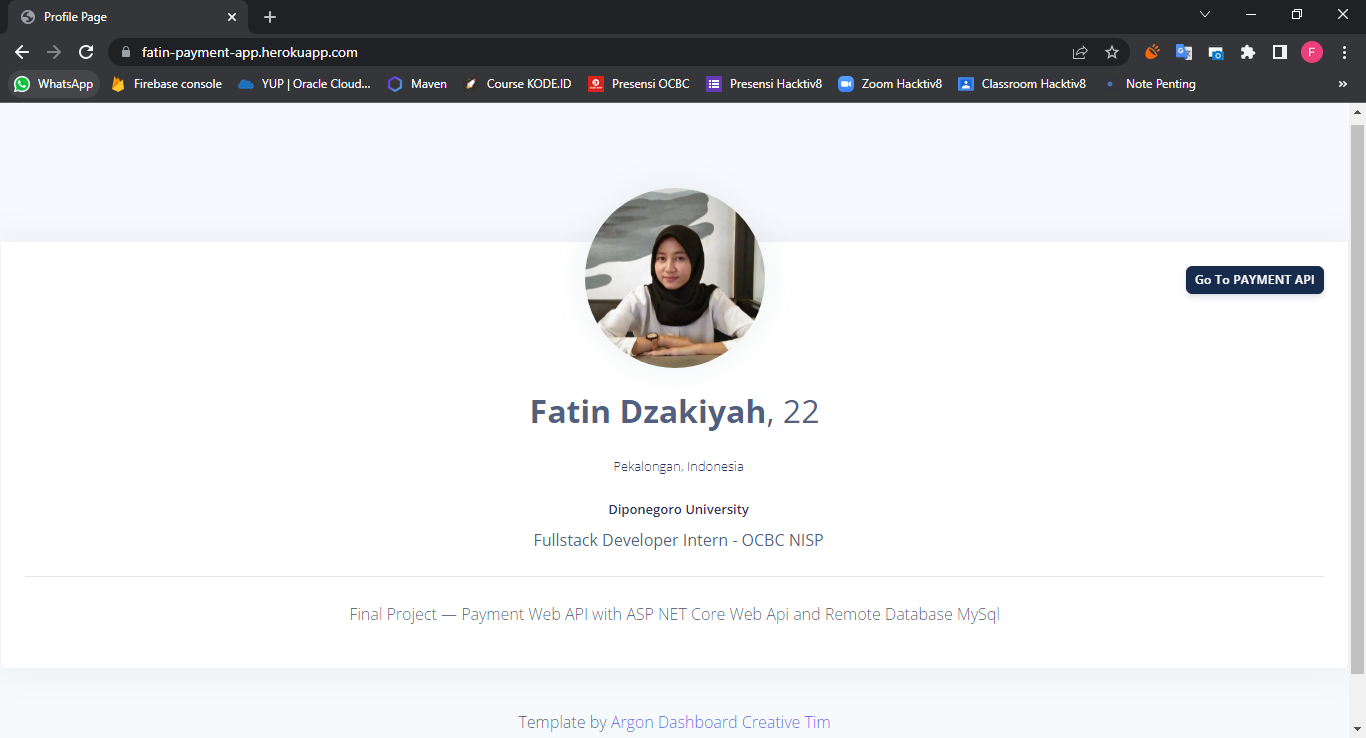
1. **Push di GitHub branch main sebagai repository baru ☑**
2. Buat repositori baru pada GitHub dengan nama fatin\_payment\_app yang di set ke Public dan kosong tanpa READ.ME maupun git ignore
3. Buka folder pada file manager tempat project PaymentApp berada
4. Lalu push project ke GitHub menggunakan Git dengan perintah sebagai berikut

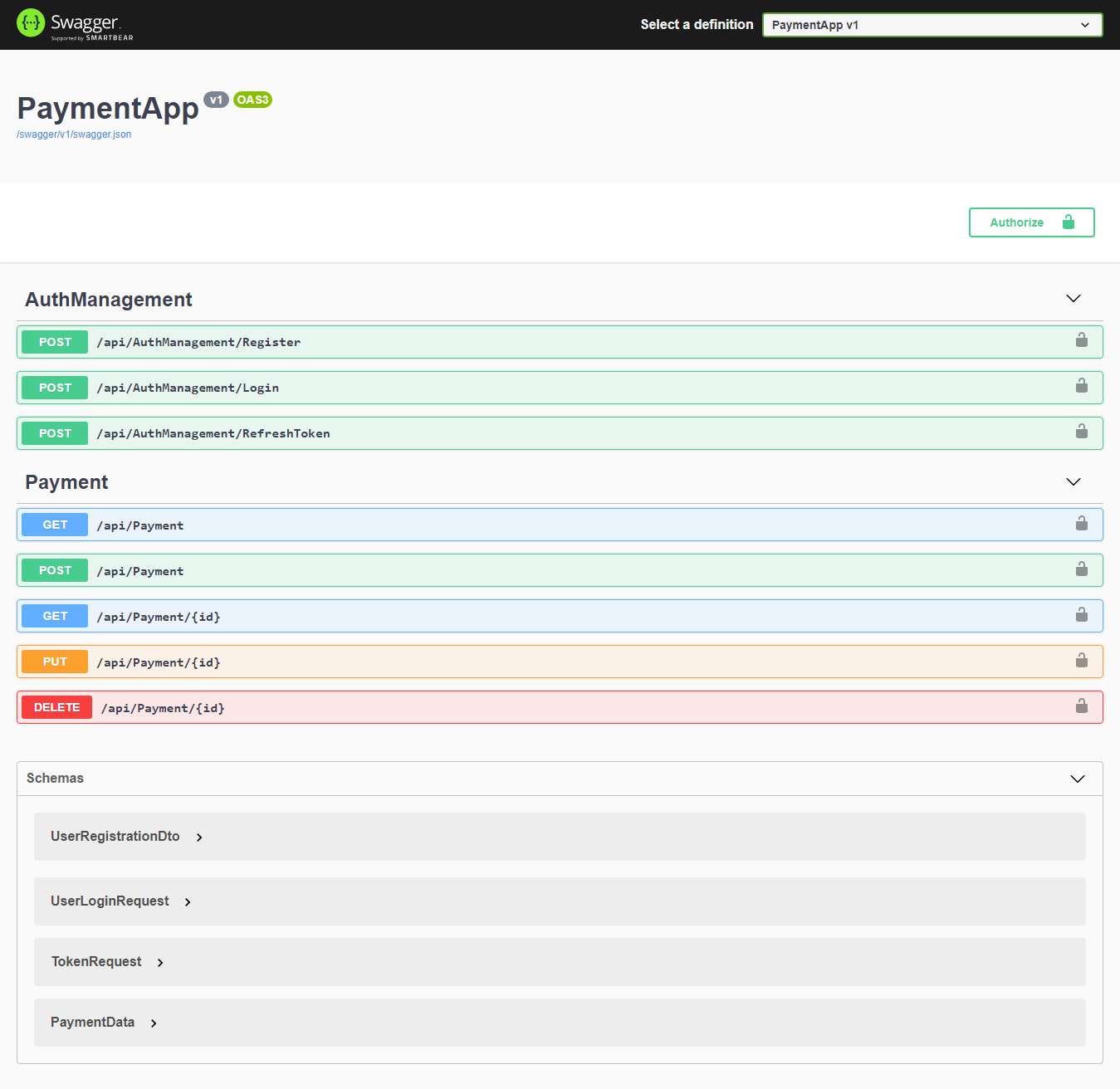
|  |
| --- |
| git init  echo “# fatin\_payment\_app” >> README.md  git add README.md  git commit -m “Initial Commit”  git checkout -b main  git add .  git commit -m “Second Commit”  git remote add origin https://github.com/f-dz/fatin\_payment\_app.git  git push -u origin main |



1. **Deploy aplikasi di Heroku ☑**
2. Buat aplikasi di Heroku dengan cara Create New App dan masukkan nama aplikasi
3. Pada menu Setting tambahkan buildpack Jincod (<https://github.com/jincod/dotnetcore-buildpack>)
4. Pada menu Deploy pilih deploy dengan GitHub dan masukkan url repositori yang berisi projek Payment API lalu klik Deploy Branch
5. Aplikasi selesai di deploy dan didapatkan link untuk membuka aplikasi (<https://fatin-payment-app.herokuapp.com/>)



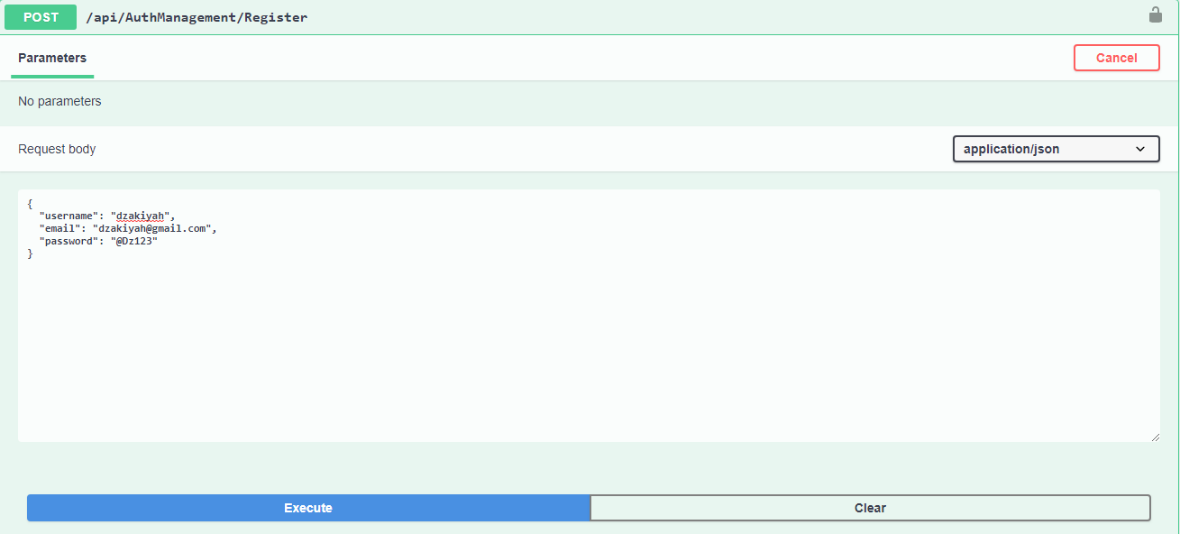
Tampilan halaman utama

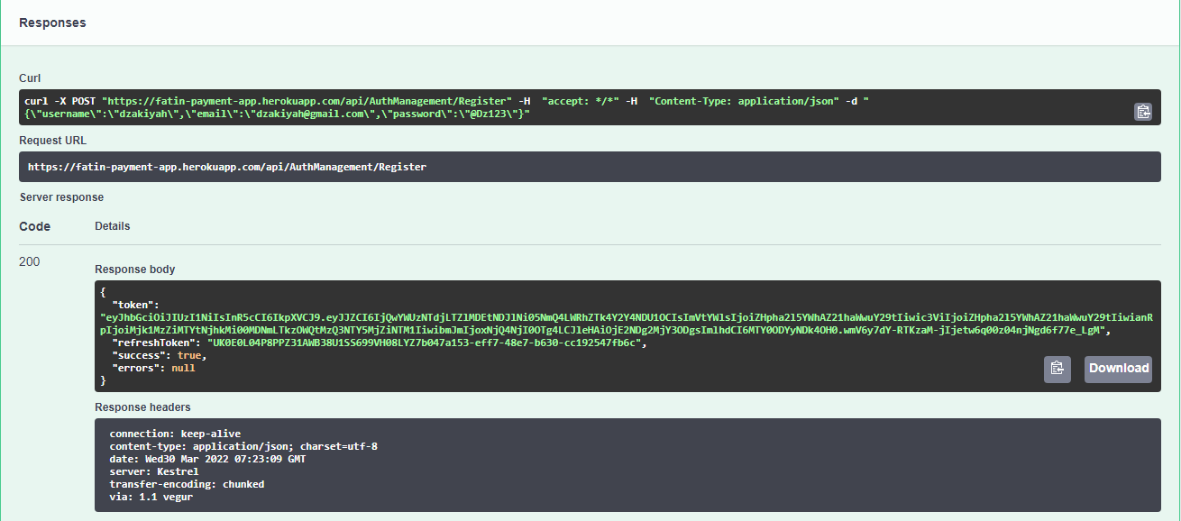
Tampilan halaman swagger

1. **Pengujian Payment API dengan Swagger dan Postman ☑**
2. **Swagger**

* REGISTER

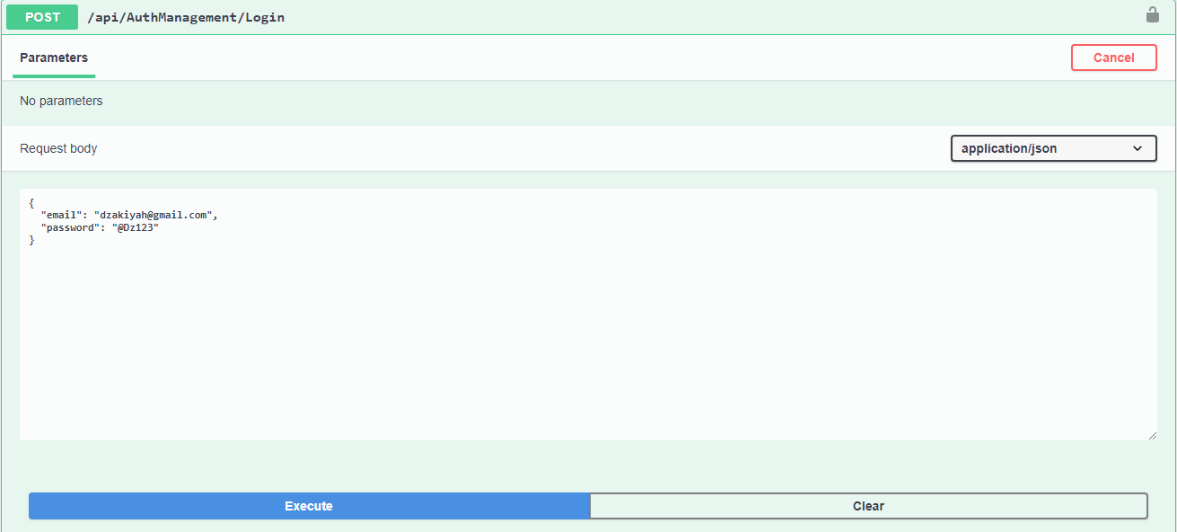
Masukkan username, email, dan password baru.

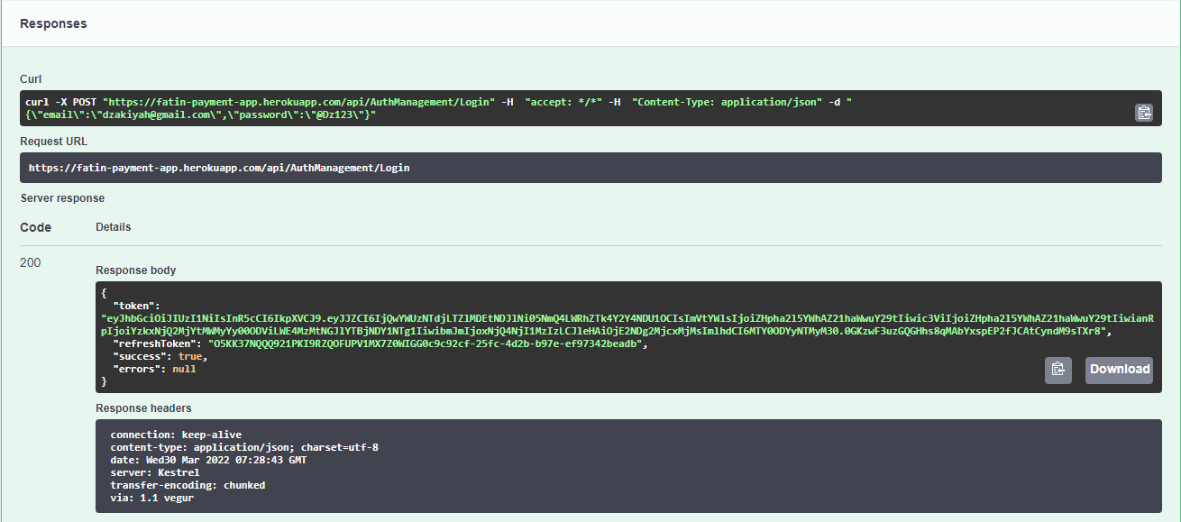




* LOGIN

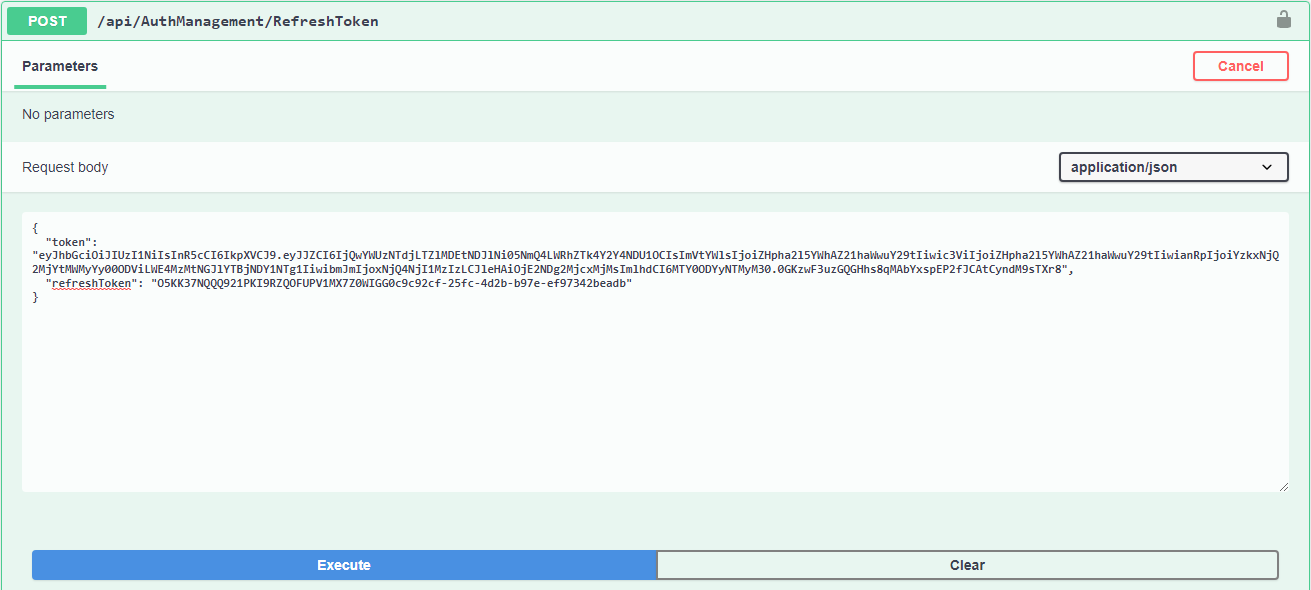
Login dengan menggunakan akun yang sebelumnya sudah didaftarkan, dan didapatkan token yang dapat digunakan untuk autentikasi. Token hanya bisa digunakan untuk mengakses menu Payment selama 30 menit setelah login. Setelah 30 menit, maka harus melakukan autentikasi ulang.

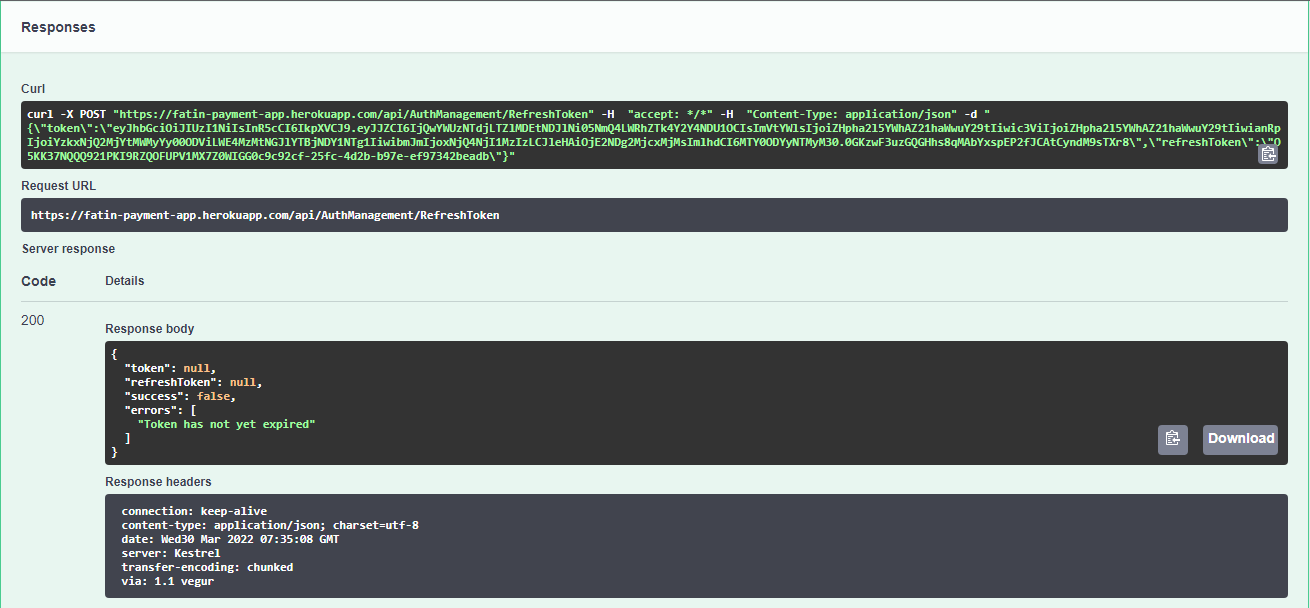


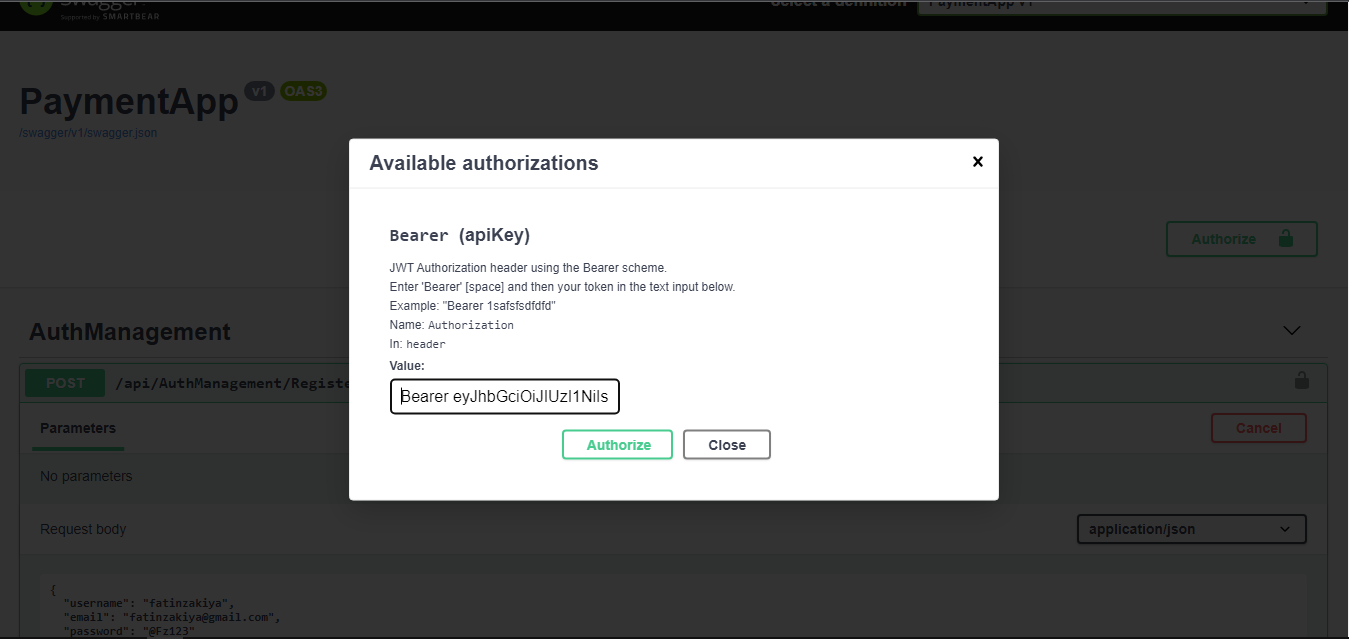


* REFRESH TOKEN

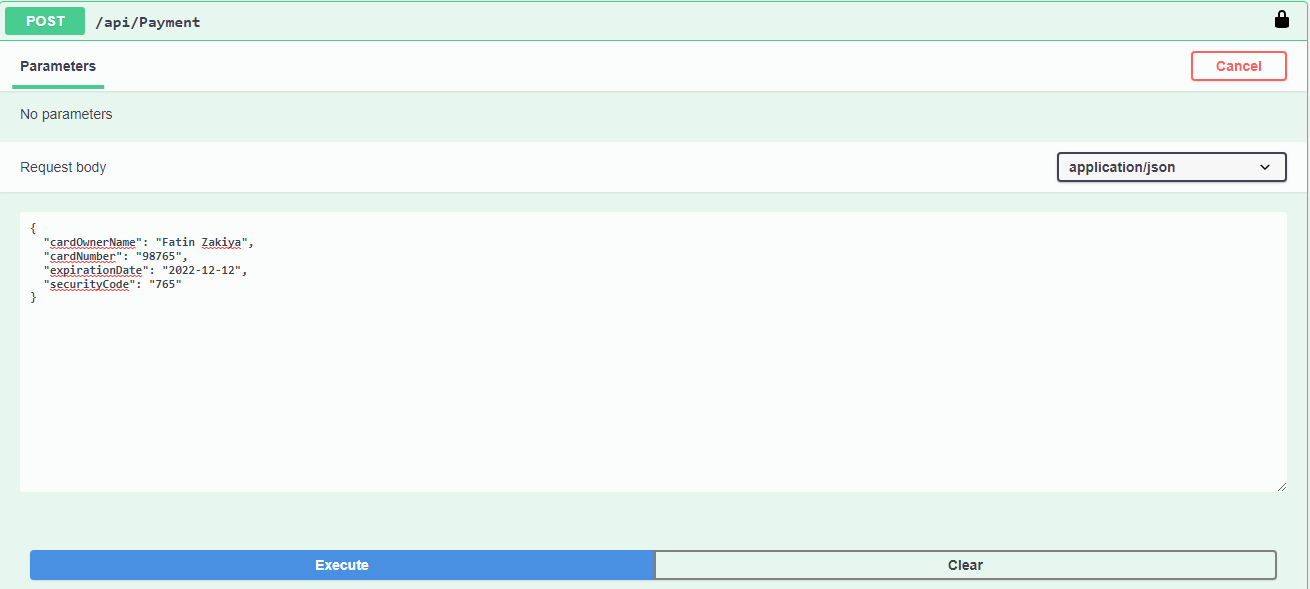
Percobaan untuk melakukan refresh token sebelum 30 menit dan mendapat pesan gagal karena token belum expired.

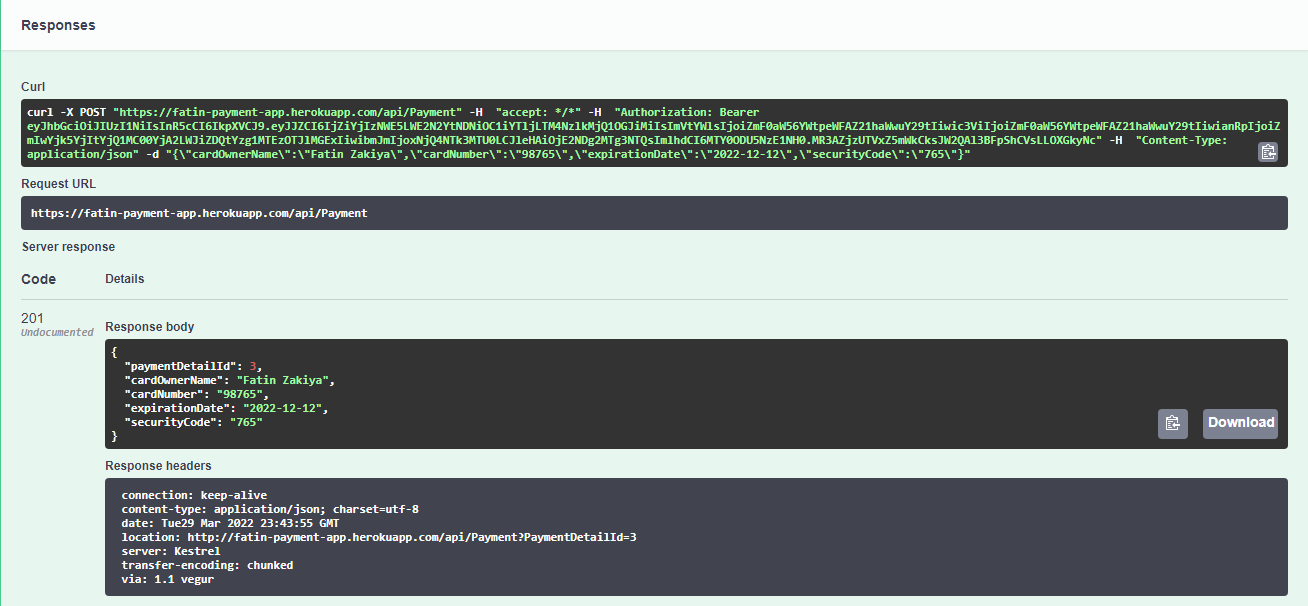




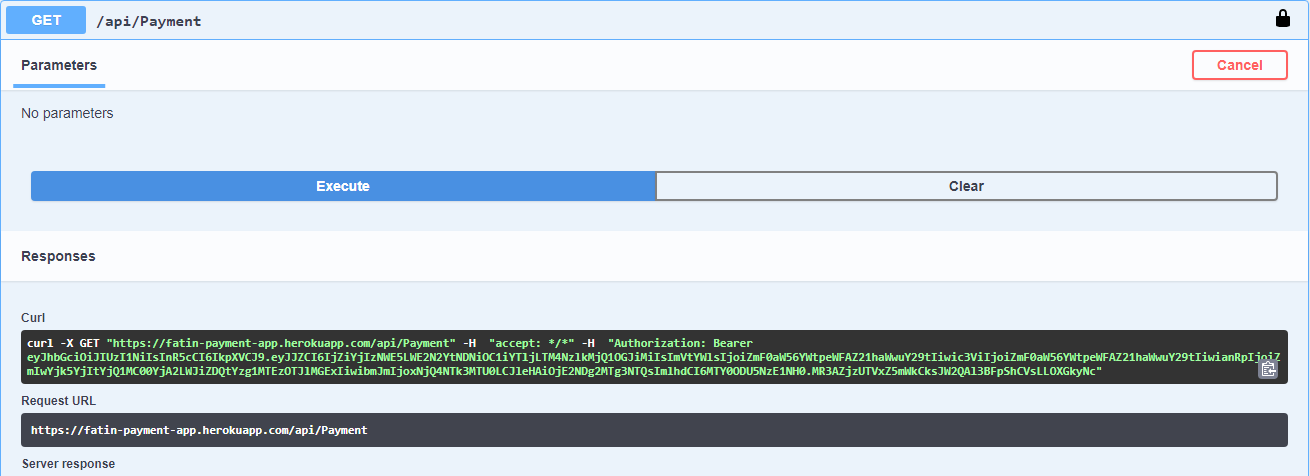
* AUTHORIZE
* PAYMENT

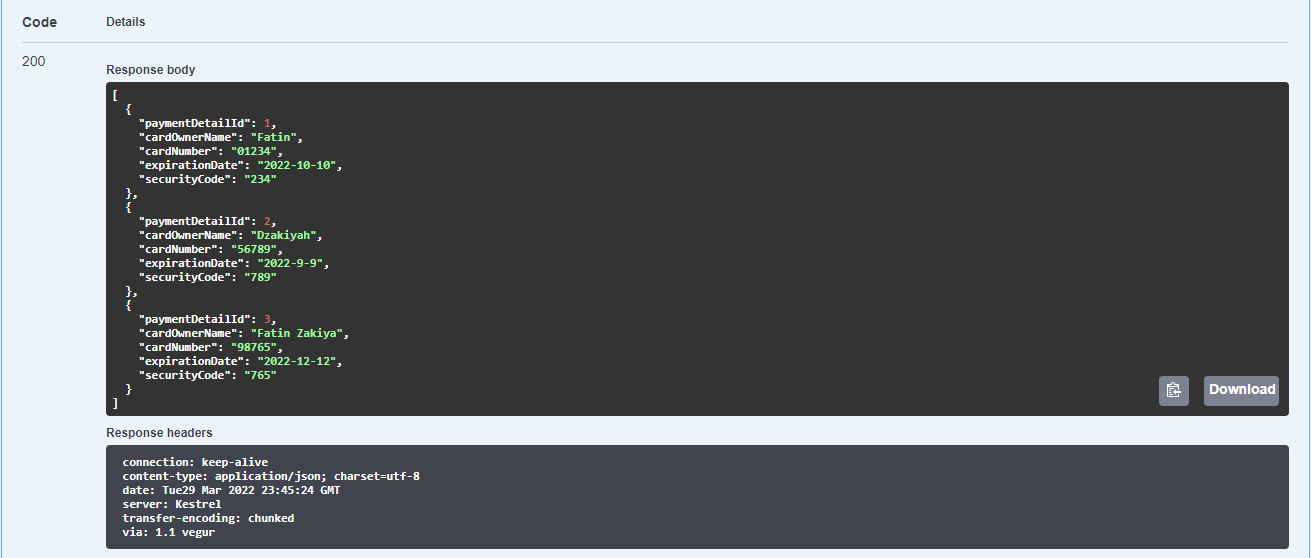
Payment POST

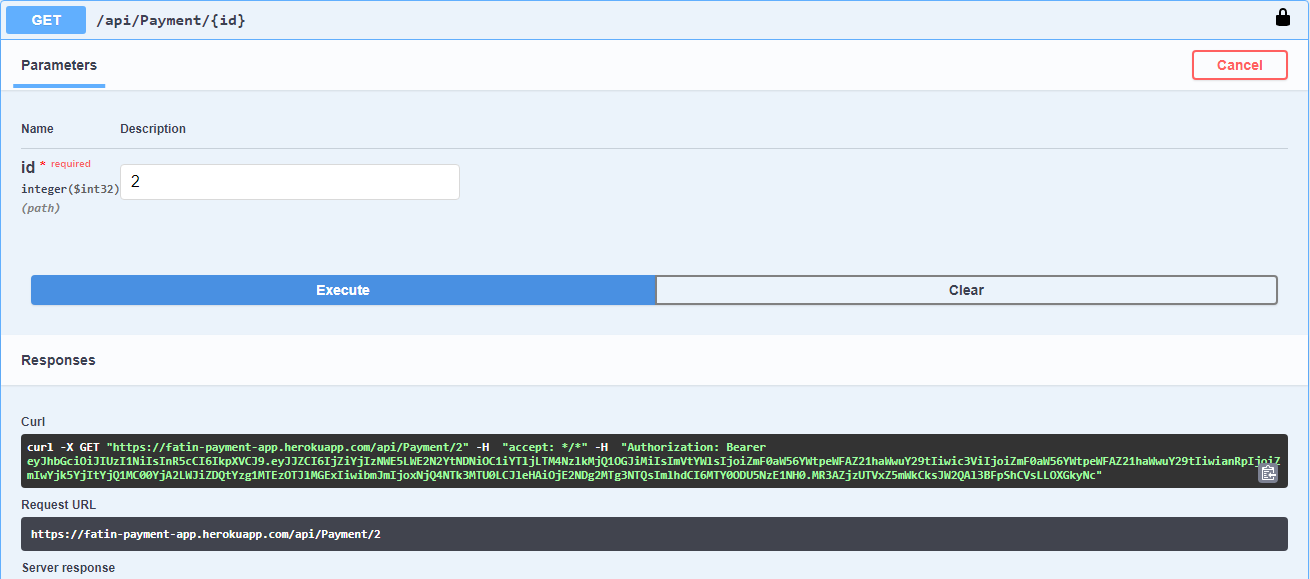
Membuat detail payment baru dengan mengisi nama, tanggal expired, dan kode keamanan.

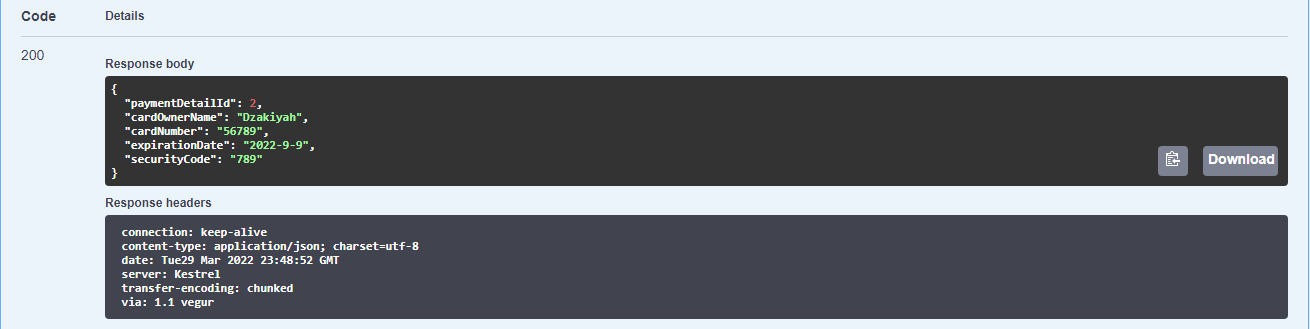


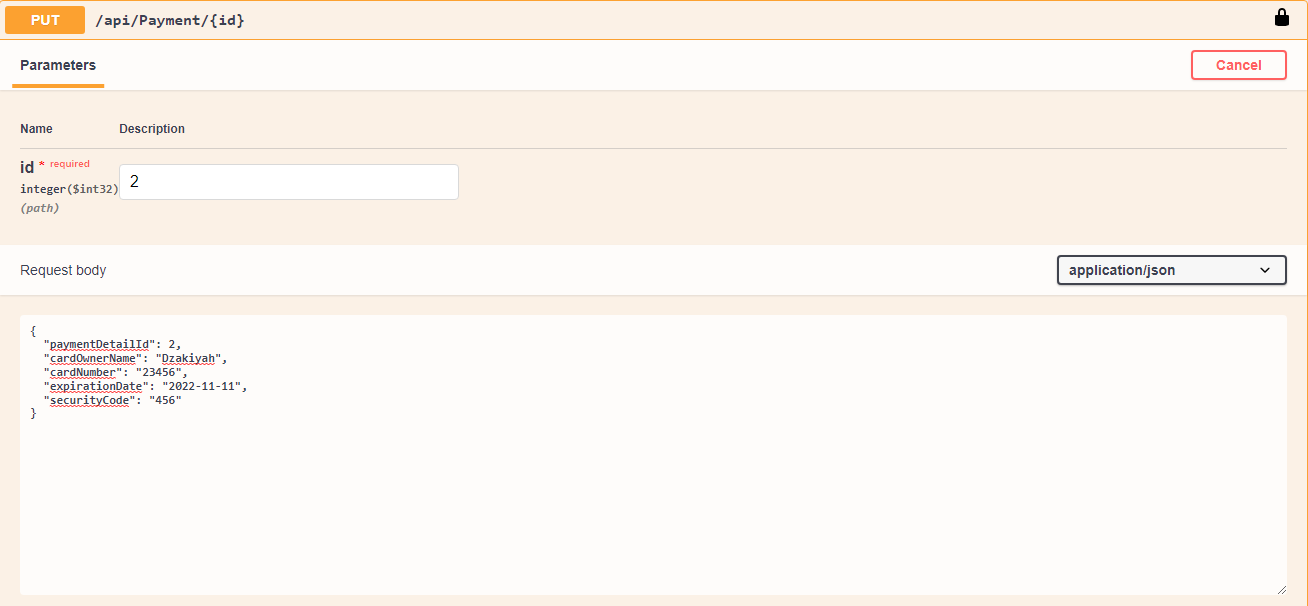
Payment GET

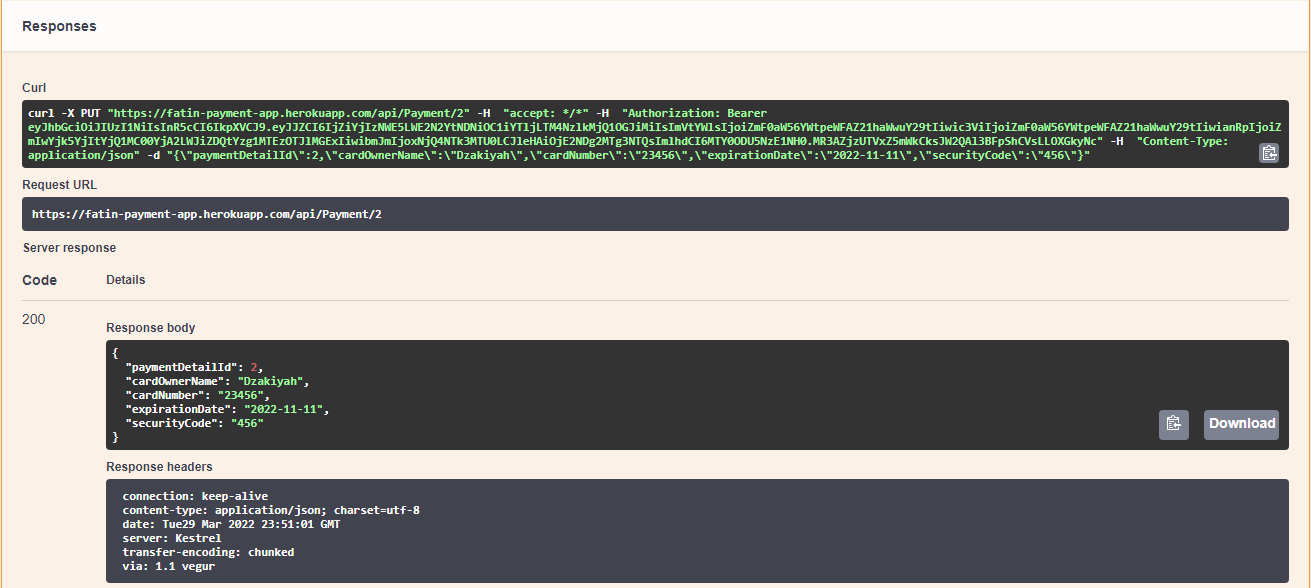
Mendapatkan semua payment yang ada pada database 

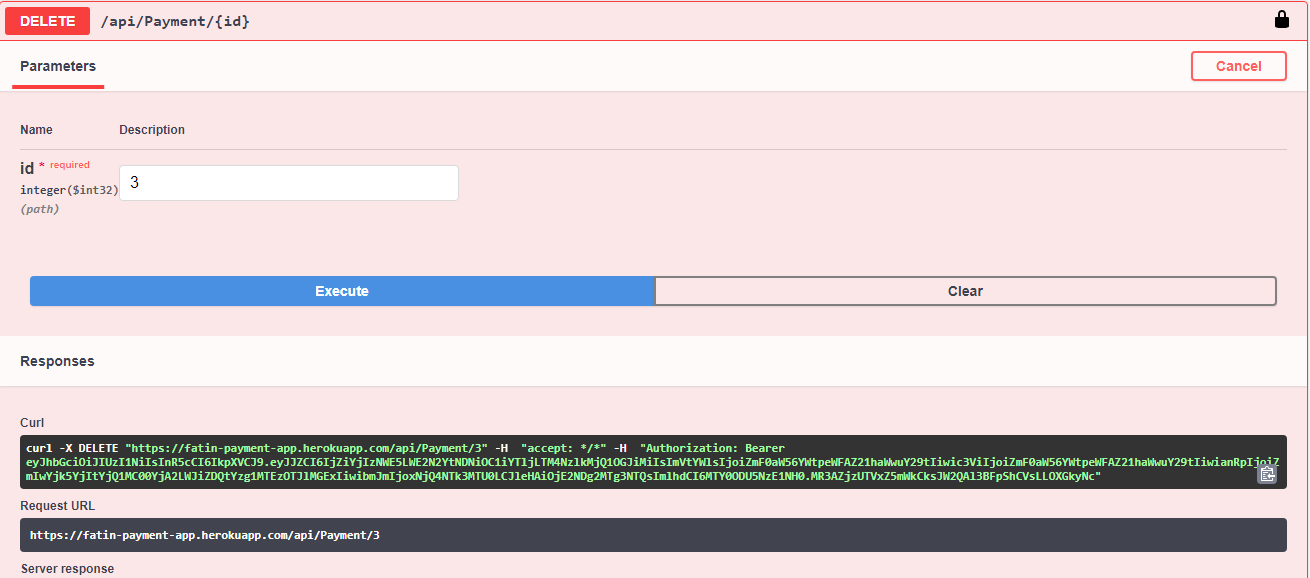


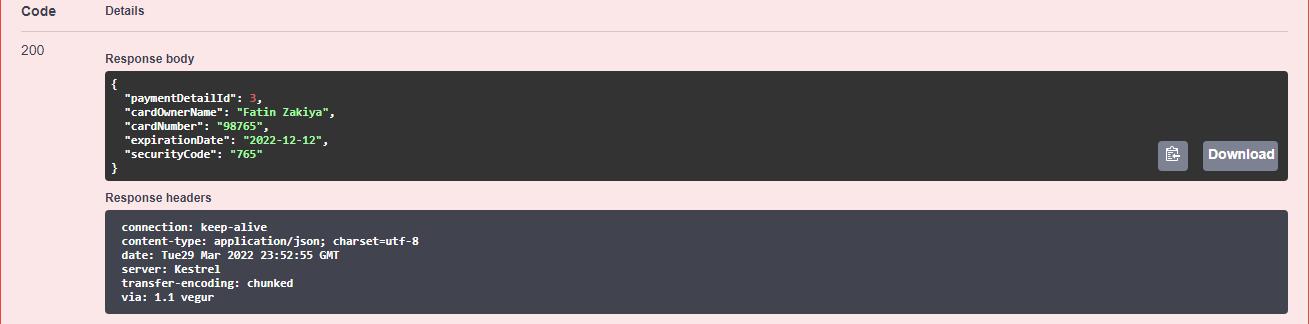
Payment GET WHERE ID



Payment PUT



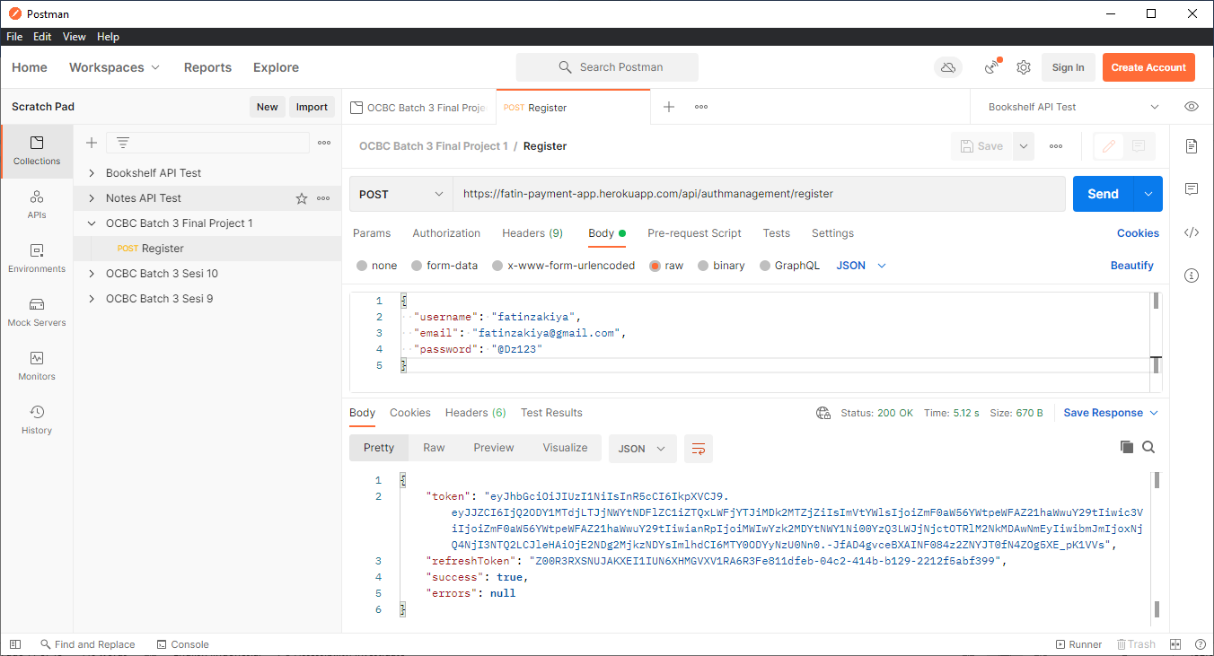
Payment DELETE



1. **Postman**

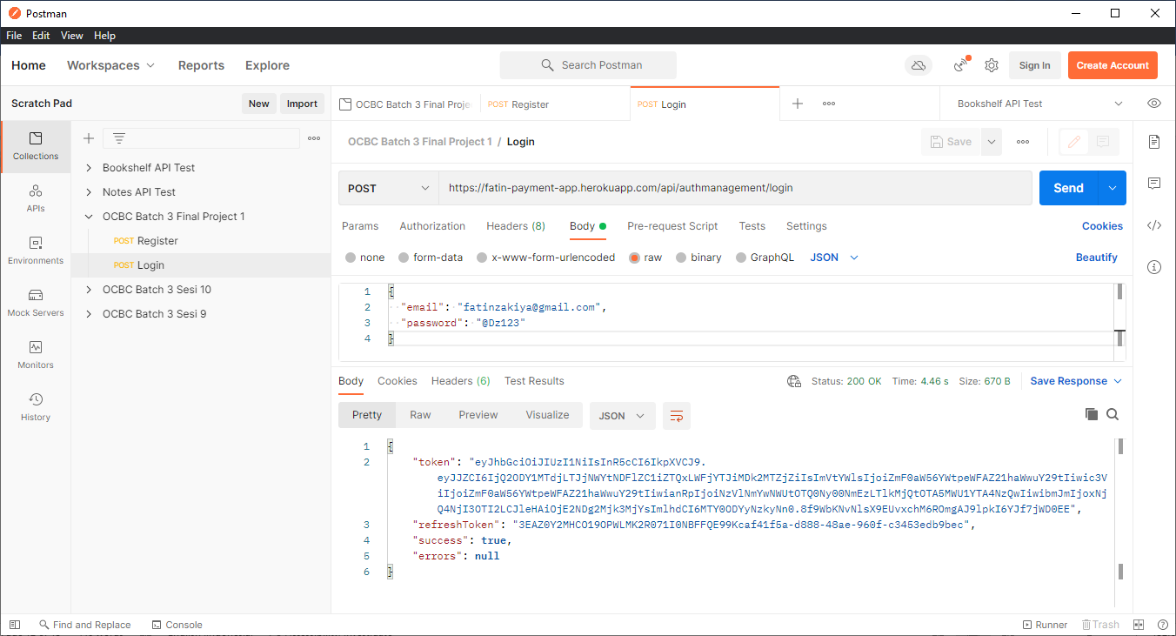
* REGISTER

Masukkan username, email, dan password baru.



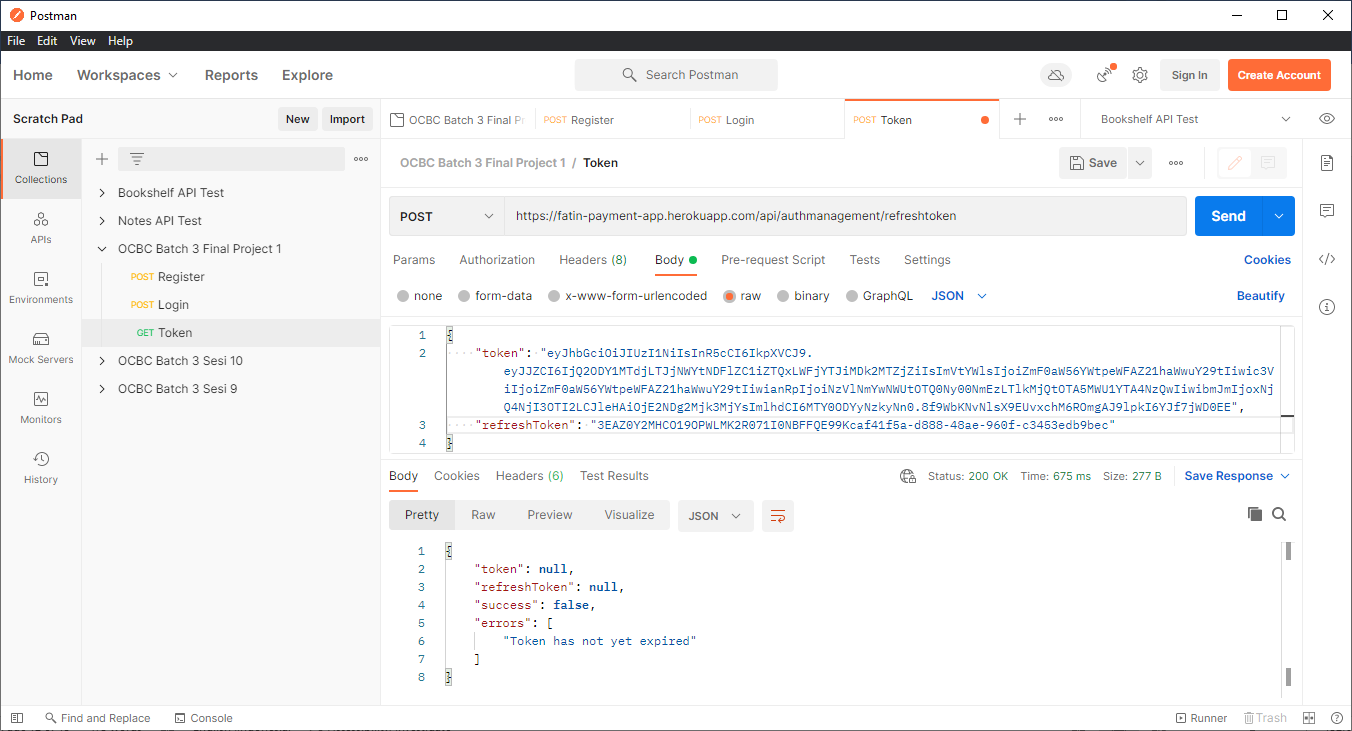
* LOGIN

Login dengan menggunakan akun yang sebelumnya sudah didaftarkan, dan didapatkan token yang dapat digunakan untuk autentikasi. Token hanya bisa digunakan untuk mengakses menu Payment selama 30 menit setelah login. Setelah 30 menit, maka harus melakukan autentikasi ulang.



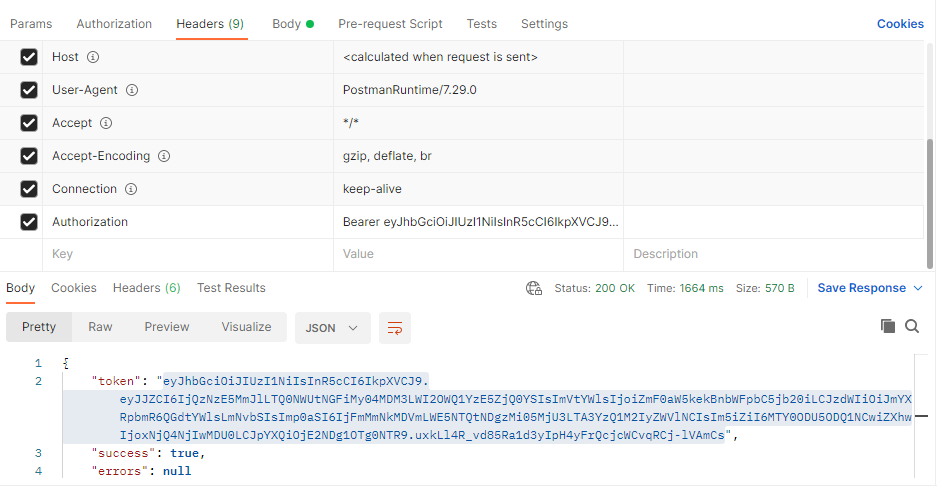
* REFRESH TOKEN

Percobaan melakukan refresh token sebelum 30 menit dan mendapatkan pesan gagal.



* AUTHORIZE

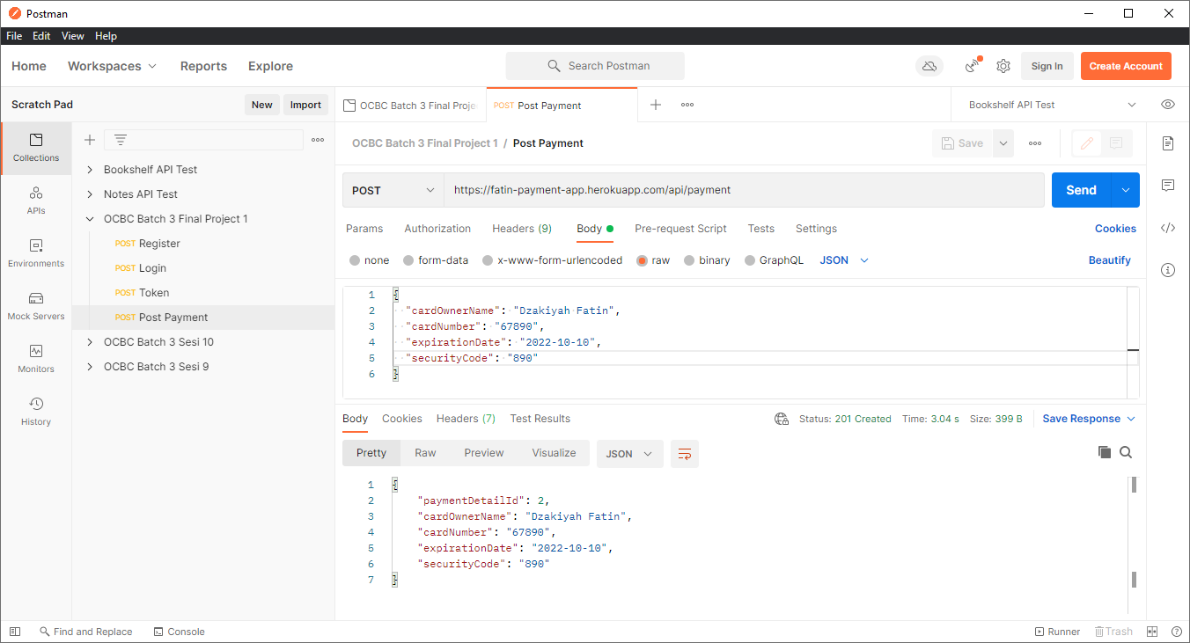
Melakukan autorisasi dengan menambahkan Key baru “Authorization” dan diisi dengan “Bearer key” di mana key diisi dengan token yang didapatkan dari hasil login.



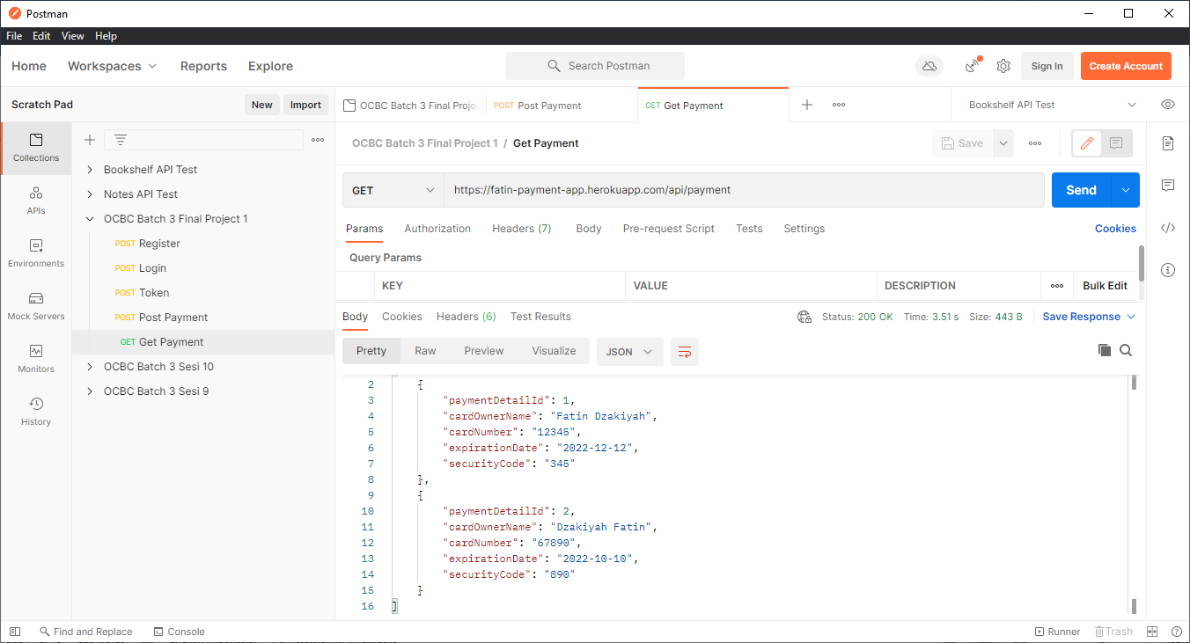
* PAYMENT

Payment POST

Membuat detail payment baru dengan mengisi nama, tanggal expired, dan kode keamanan.

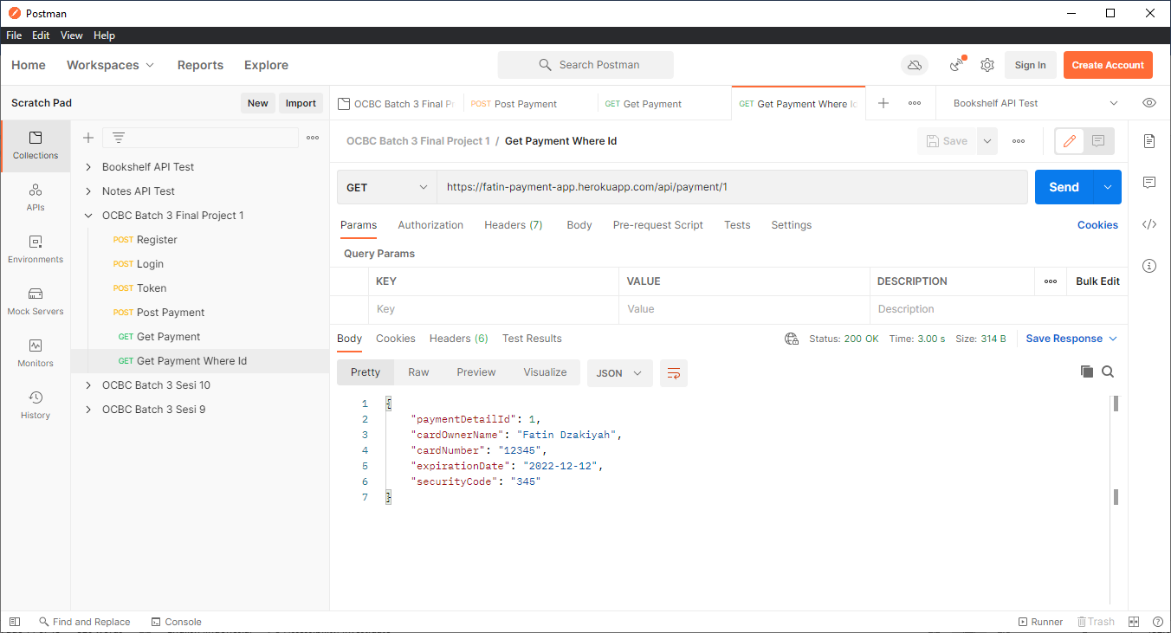


Payment GET

Mendapatkan semua payment yang ada pada database.

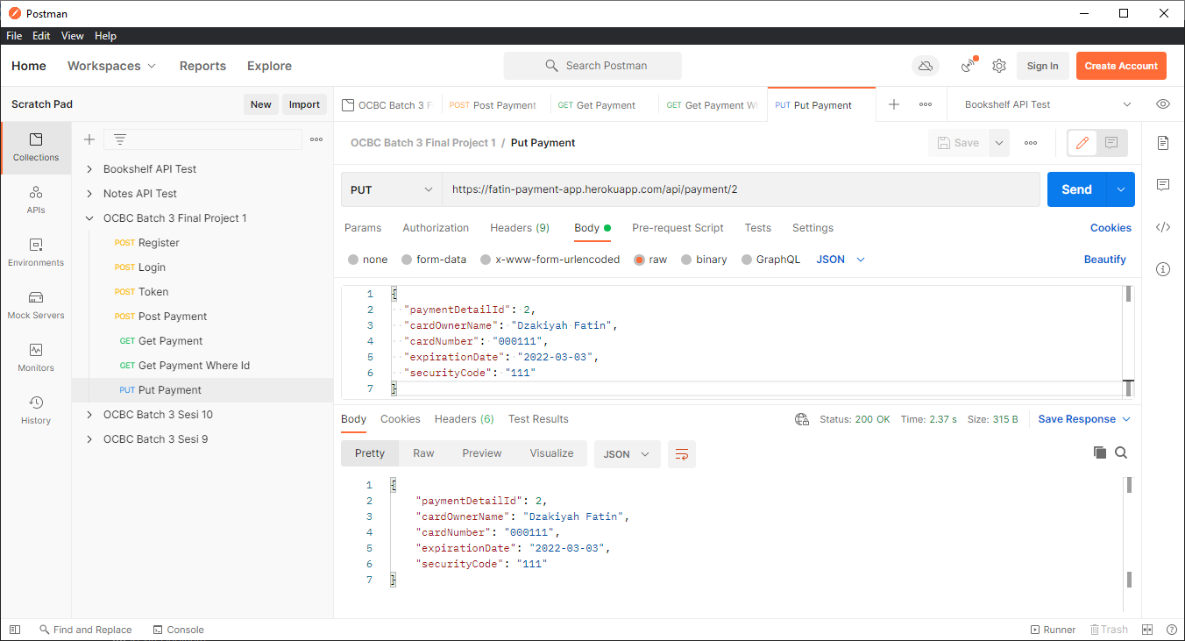
Payment GET WHERE ID

Mendapatkan payment dengan id tertentu



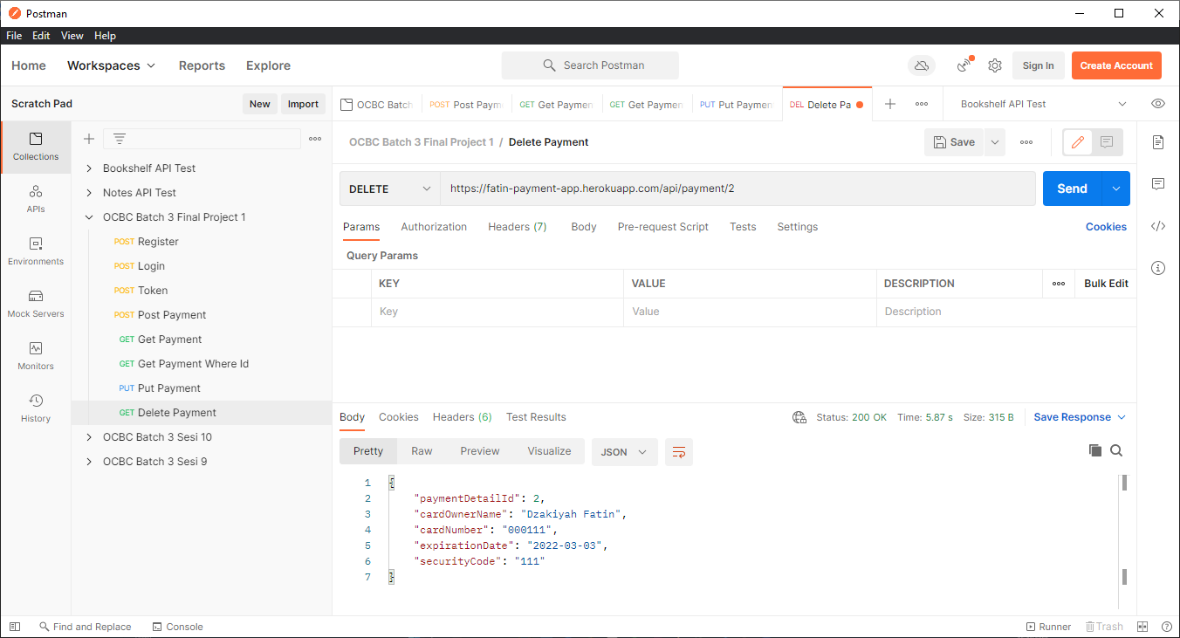
Payment PUT

Mengedit data dengan id tertentu, dan menampilkan response beruba data baru hasil perubahan.



Payment DELETE

Menghapus data dengan id tertentu dan menampilkan response berupa data yang berhasil dihapus.



1. **Save Hasil CRUD dari Postman ☑**

