Final Project - Payment

No. Peserta : FSDO001ONL001

Nama : Rara Danniswara Musin

A. Penjelasan Program

Program ini dibuat dengan menggunakan dua database MySql yang berbeda, database yang pertama dijalankan di localhost sehingga dapat dilakukan testing melalui swagger dan postman. Sedangkan database yang kedua adalah database MySql yang telah di hosting sehingga hanya dapat melakukan testing melalui postman.

Program ini didevelop dengan menggunakan MySql database, ASP .Net, tools yang digunakan selama proses development adalah :

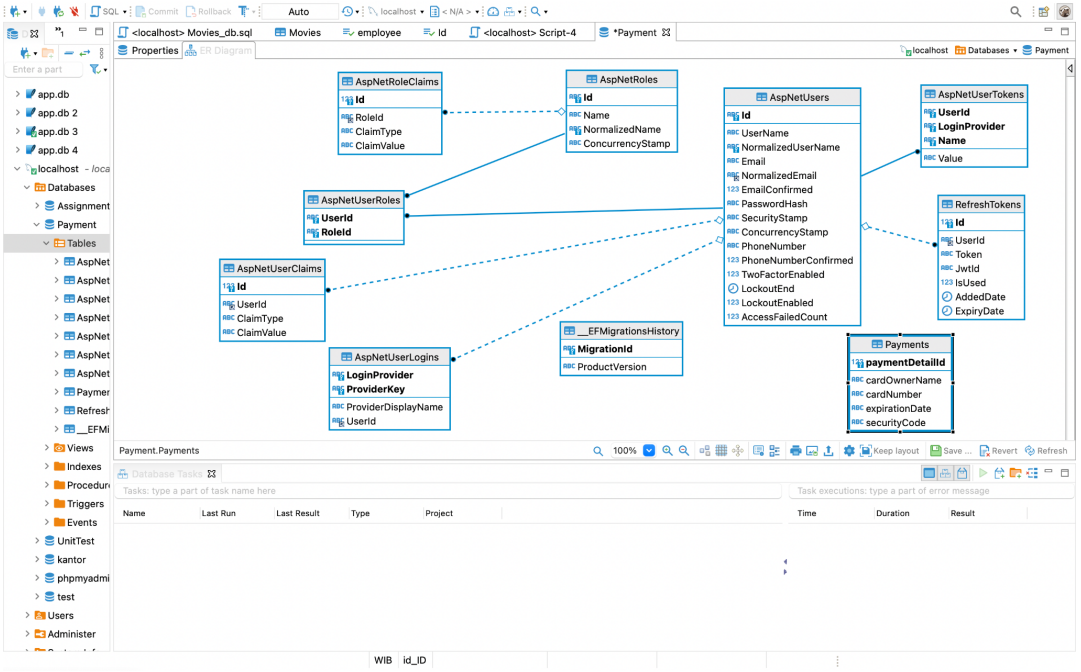
1. Visual Studio Code : code editor yang digunakan dalam membuat program
2. Postman : testing api yang telah dibuat
3. DBeaver : mengecek database yang sedang digunakan baik lokal maupun yang sudah di hosting
4. Swagger UI : testing api yang telah dibuat
5. Github : code documentation
6. Heroku : deploy untuk running program menggunakan database yang telah di hosting

Seluruh endpoint yang terdapat pada program ini menggunakan authentication. Authentication digunakan untuk melindungi data yang terdapat di database dari tindakan pencurian data. Sebelum menjalankan HTTP Request(GET, POST, PUT, DELETE), user harus melakukan register dan login untuk mendapatkan token yang akan digunakan untuk authentication.

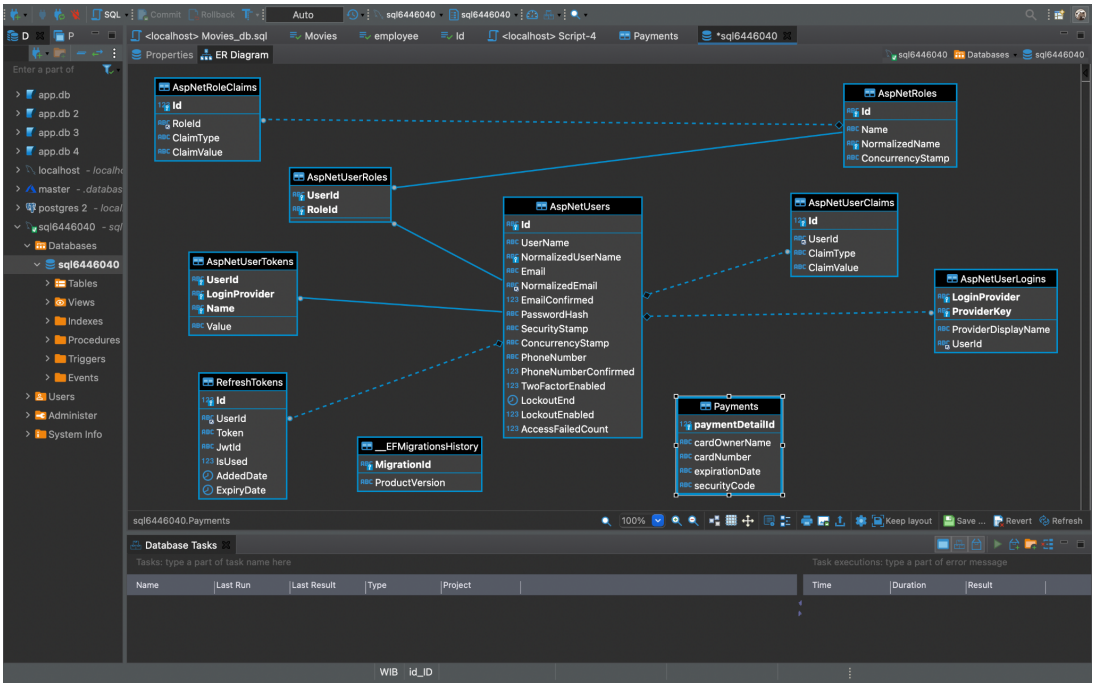
B. Kendala yang Ditemukan

Saya mengalami kendala dalam hal export database ke dalam bentuk .sql, maka dari itu saya akan melampirkan screen capture dari database yang sudah dibuat:

1. Mysql Local



2. Mysql yang sudah di hosting

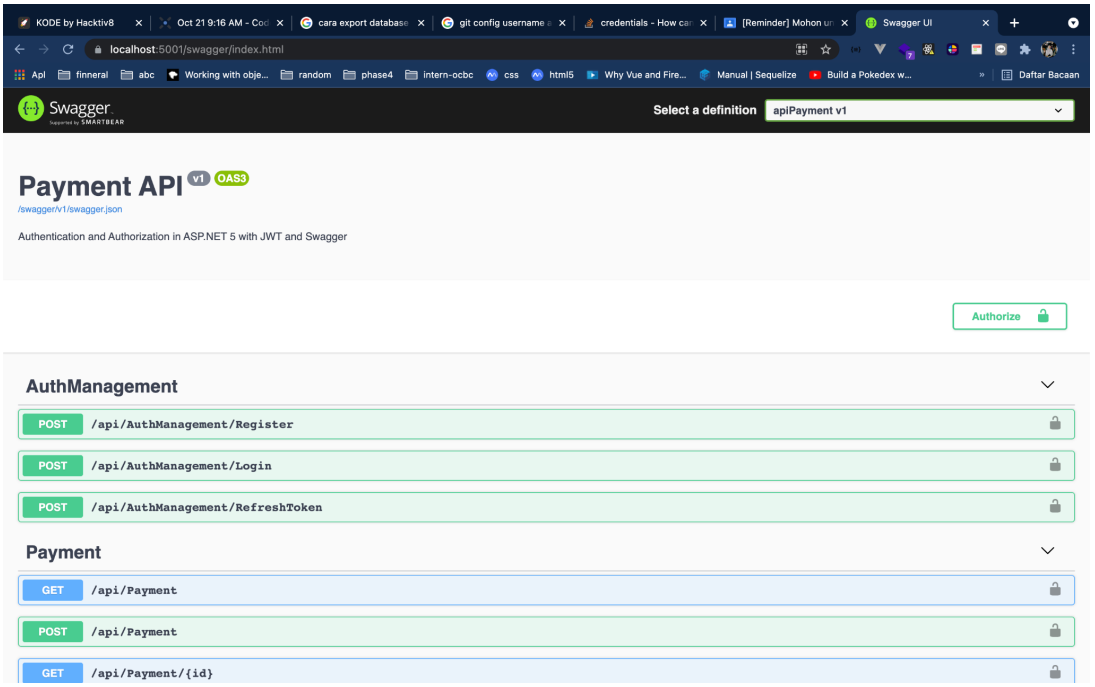


seluruh tabel ini ter-generate ketika saya menjalankan migrasi dan update database melalui dotnet cli.

C. Panduan Menjalankan Program

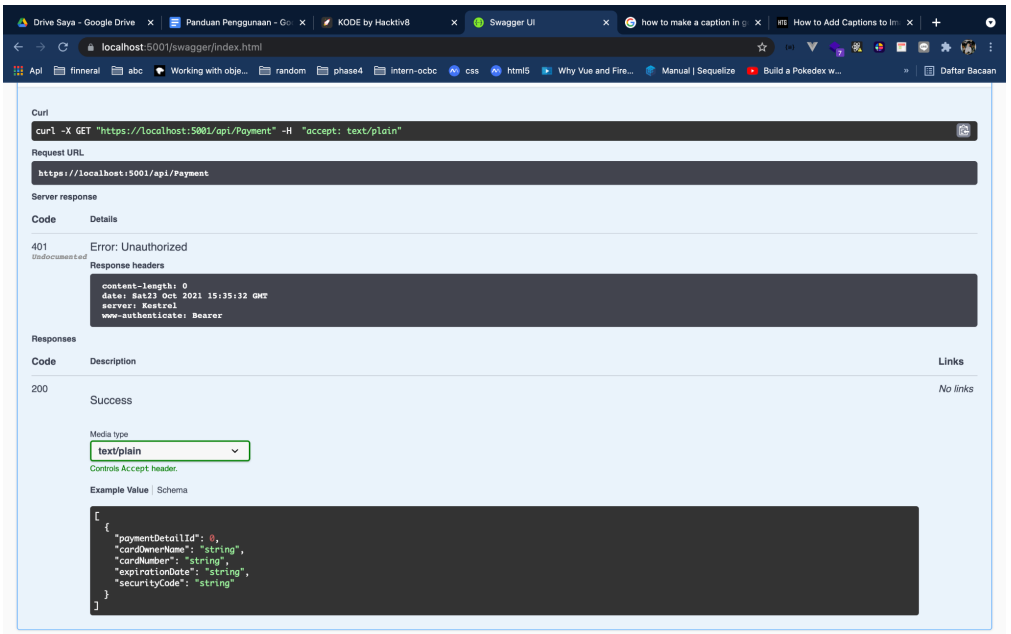
a. Menjalankan program secara local

1. clone repository dari github : https://github.com/LiliAmber/001_RaraDanniswara_FinalProject
2. buka folder apiPayment
3. pastikan xampp sudah berjalan(karena menggunakan db MySql local)
4. buka file *appsettings.json*, pastikan *ConnectionStrings* yang digunakan adalah:
   "ConnectionStrings: {
   "myconn":
   "Server=localhost;Database=Payment;Uid=root;Pwd;SslMode=none"
   }"
5. buka terminal pada direktori apiPayment, lalu migrate db di local dengan cara mengetikan *dotnet ef migrations "initial migrate"*, selanjutnya update db dengan cara *dotnet ef database update*. Hal ini dilakukan dengan asumsi belum ada db yang akan digunakan pada device.
6. lalu ketikan *dotnet run*, lalu klik button enter untuk menjalankan program, jika program berjalan lancar maka akan mengarahkan ke browser dan membuka page swagger UI, bila hal ini tidak terjadi maka user dapat mengakses link berikut : https://localhost:5001/swagger/index.html



Tampilan Swagger

7. Pada apiPayment terdapat 8 endpoint yang terbagi ke dalam 2 kategori, yaitu AuthManagement(3 endpoint) dan Payment(5 endpoint). Untuk mengakses seluruh endpoint yang terdapat pada kategori Payment, user harus memasukan token yang didapat pada proses login, jika user mencoba untuk mengakses tanpa token maka akan mendapatkan response error authentication.



8. Test masing-masing endpoint:

   Pada kategori AuthManagement terdapat 3 endpoint, yaitu:

   - POST api/AuthManagement/Register

     endpoint ini digunakan untuk melakukan registrasi bagi user yang belum memiliki akun untuk melakukan proses login. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

     ```
     {
       "username": "string",
       "email": "user@example.com",
       "password": "string"
     }
     ```

     Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program.

     Note : harus menggunakan format email yang benar, password harus terdiri dari 1 alphanumeric, 1 angka dan 1 huruf kapital. Contoh: User1!

   - POST api/AuthManagement/Login

     endpoint ini digunakan untuk melakukan login bagi user yang sudah memiliki akun. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

     ```
     {
       "email": "user@example.com",
       "password": "string"
     }
     ```

     Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program. User akan mendapatkan token yang dapat digunakan untuk mengakses endpoint yang terdapat pada kategori payment. User dapat menginput token yang didapat pada button *Authorization* (di pojok kanan atas) dengan format sebagai berikut:

     Bearer (masukan token)

- POST api/AuthManagement/RefreshToken

  endpoint ini digunakan untuk melakukan validasi token yang user dapatkan pada proses login. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

  ```
  {
    "token": "string",
    "refreshToken": "string"
  }
  ```

  Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program.

Pada kategori Payment terdapat 5 endpoint, yaitu:

- GET api/Payment

  endpoint ini digunakan untuk mendapatkan seluruh data payment yang terdapat didalam database. Endpoint ini tidak memiliki request body, sehingga user bisa klik button execute tanpa melengkapi request body.

- POST api/Payment

  endpoint ini digunakan untuk memasukan data baru ke dalam database. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

  ```
  {
    "paymentDetailId": 0,
    "cardOwnerName": "string",
    "cardNumber": "string",
    "expirationDate": "string",
    "securityCode": "string"
  }
  ```

  note : user bisa mengabaikan value dari key *paymentDetailId*, karena value nya sudah dibuat auto increment.

  Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program.

- GET api/Payment/{id}

  endpoint ini digunakan untuk mendapatkan data payment yang terdapat didalam database sesuai dengan id yang di input oleh user. User harus melengkapi value dari key yang terdapat pada request parameters, yaitu:

  ```
  {
    "id": "<payment details id>"
  }
  ```

  Setelah melengkapi request parameters, klik button execute untuk mendapatkan response dari program.

- PUT api/Payment/{id}

  endpoint ini digunakan untuk memperbarui data payment yang terdapat didalam database sesuai dengan id yang di input oleh user. User harus melengkapi value dari key yang terdapat pada request parameters, yaitu:

  ```
  {
    "id": "<payment details id>"
  }
  ```

  User juga harus melengkapi value dari key yang terdapat pada request body, yaitu:

  ```
  {
    "paymentDetailId": 0,
    "cardOwnerName": "string",
    "cardNumber": "string",
    "expirationDate": "string",
    "securityCode": "string"
  }
  ```

  Setelah melengkapi request parameters dan request body, klik button execute untuk mendapatkan response dari program.

- DELETE api/Payment/{id}

  endpoint ini digunakan untuk menghapus secara permanent data payment yang terdapat didalam database sesuai dengan id yang di input oleh user.

User harus melengkapi value dari key yang terdapat pada request parameters, yaitu:

```
{
  "id": "<payment details id>"
}
```

Setelah melengkapi request parameters, klik button execute untuk mendapatkan response dari program.

b. Menjalankan program menggunakan database yang sudah di hosting

1. clone repository dari github : https://github.com/LiliAmber/001_RaraDanniswara_FinalProject
2. buka DBeaver, klik new connection, lalu pilih mysql. Lengkapi data sesuai dengan value dari *myconn*(pada point 4), lalu klik *Test Connection* untuk membuat koneksi ke mysql db yang sudah di hosting.
3. buka folder apiPayment
4. buka file *appsettings.json*, pastikan *ConnectionStrings* yang digunakan adalah:
   "ConnectionStrings: {
   "myconn":
   "Server=sql6.freemysqlhosting.net;Database:sql16446040;Username=sql6446040;Pwd=EDm1dgBiLZ;SslMode=none"
   }"
5. buka terminal pada direktori apiPayment, lalu migrate db di local dengan cara mengetikan *dotnet ef migrations "initial migrate"*, selanjutnya update db dengan cara *dotnet ef database update*. Hal ini dilakukan dengan asumsi belum ada db yang akan digunakan pada device.
6. lalu ketikan *dotnet run*, lalu klik button enter untuk menjalankan program. Lalu buka postman untuk menguji semua endpoint yang terdapat pada program. Base url untuk mengakses seluruh endpoint adalah: https://apipayment.herokuapp.com+endpoint

contohnya:
https://apipayment.herokuapp.com/api/AuthManagement/Register

7. Pada apiPayment terdapat 8 endpoint yang terbagi ke dalam 2 kategori, yaitu AuthManagement(3 endpoint) dan Payment(5 endpoint). Untuk mengakses seluruh endpoint yang terdapat pada kategori Payment, user harus memasukan token yang didapat pada proses login, jika user mencoba untuk mengakses tanpa token maka akan mendapatkan response error authentication.
8. Test masing-masing endpoint:
   Pada kategori AuthManagement terdapat 3 endpoint, yaitu:
   - POST api/AuthManagement/Register
     endpoint ini digunakan untuk melakukan registrasi bagi user yang belum memiliki akun untuk melakukan proses login. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

     ```
     {
       "username": "string",
       "email": "user@example.com",
       "password": "string"
     }
     ```

     Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program.
     Note : harus menggunakan format email yang benar, password harus terdiri dari 1 alphanumeric, 1 angka dan 1 huruf kapital. Contoh: User1!
   - POST api/AuthManagement/Login
     endpoint ini digunakan untuk melakukan login bagi user yang sudah memiliki akun. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

     ```
     {
       "email": "user@example.com",
       "password": "string"
     }
     ```

Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program. User akan mendapatkan token yang dapat digunakan untuk mengakses endpoint yang terdapat pada kategori payment. User dapat menginput token yang didapat pada tab *Headers* dengan format sebagai berikut: *Authorization (masukan token)*

- POST api/AuthManagement/RefreshToken

  endpoint ini digunakan untuk melakukan validasi token yang user dapatkan pada proses login. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

```
{
  "token": "string",
  "refreshToken": "string"
}
```

  Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program.

Pada kategori Payment terdapat 5 endpoint, yaitu:

- GET api/Payment

  endpoint ini digunakan untuk mendapatkan seluruh data payment yang terdapat didalam database. Endpoint ini tidak memiliki request body, sehingga user bisa klik button execute tanpa melengkapi request body.

- POST api/Payment

  endpoint ini digunakan untuk memasukan data baru ke dalam database. User harus melengkapi value dari key yang terdapat pada request body, yaitu:

```
{
  "paymentDetailId": 0,
  "cardOwnerName": "string",
  "cardNumber": "string",
  "expirationDate": "string",
  "securityCode": "string"
}
```

note : user bisa mengabaikan value dari key *paymentDetailId*, karena value nya sudah dibuat auto increment.

Setelah melengkapi request body, klik button execute untuk mendapatkan response dari program.

- GET api/Payment/{id}

  endpoint ini digunakan untuk mendapatkan data payment yang terdapat didalam database sesuai dengan id yang di input oleh user. User harus melengkapi value dari key yang terdapat pada request parameters, yaitu:

  ```
  {
    "id": "<payment details id>"
  }
  ```

  Setelah melengkapi request parameters, klik button execute untuk mendapatkan response dari program.

- PUT api/Payment/{id}

  endpoint ini digunakan untuk memperbarui data payment yang terdapat didalam database sesuai dengan id yang di input oleh user. User harus melengkapi value dari key yang terdapat pada request parameters, yaitu:

  ```
  {
    "id": "<payment details id>"
  }
  ```

  User juga harus melengkapi value dari key yang terdapat pada request body, yaitu:

  ```
  {
    "paymentDetailId": 0,
    "cardOwnerName": "string",
    "cardNumber": "string",
    "expirationDate": "string",
    "securityCode": "string"
  }
  ```

  Setelah melengkapi request parameters dan request body, klik button execute untuk mendapatkan response dari program.

- DELETE api/Payment/{id}

  endpoint ini digunakan untuk menghapus secara permanent data payment yang terdapat didalam database sesuai dengan id yang di input oleh user.

  User harus melengkapi value dari key yang terdapat pada request parameters, yaitu:

  ```
  {
    "id": "<payment details id>"
  }
  ```

  Setelah melengkapi request parameters, klik button execute untuk mendapatkan response dari program.