Nama : Kevin Elvio C. P
NIM : 222410102088
GitHub: https://github.com/KevinElvio/Pemrograman-Antarmuka-Aplikasi/tree/main/Modul_3/PercobaanAPI1

Percobaan 1: Menambahkan JWT Authentikasi

Hasil dari percobaan 1 :

- Berhasil mengimplementasikan Jwt
- Berhasil menjalakan pada Postgresql
- Berhasil menjalankan pada Postman
- Berhasil mengamankan sebuah data

Analisa percoabaan 1 :

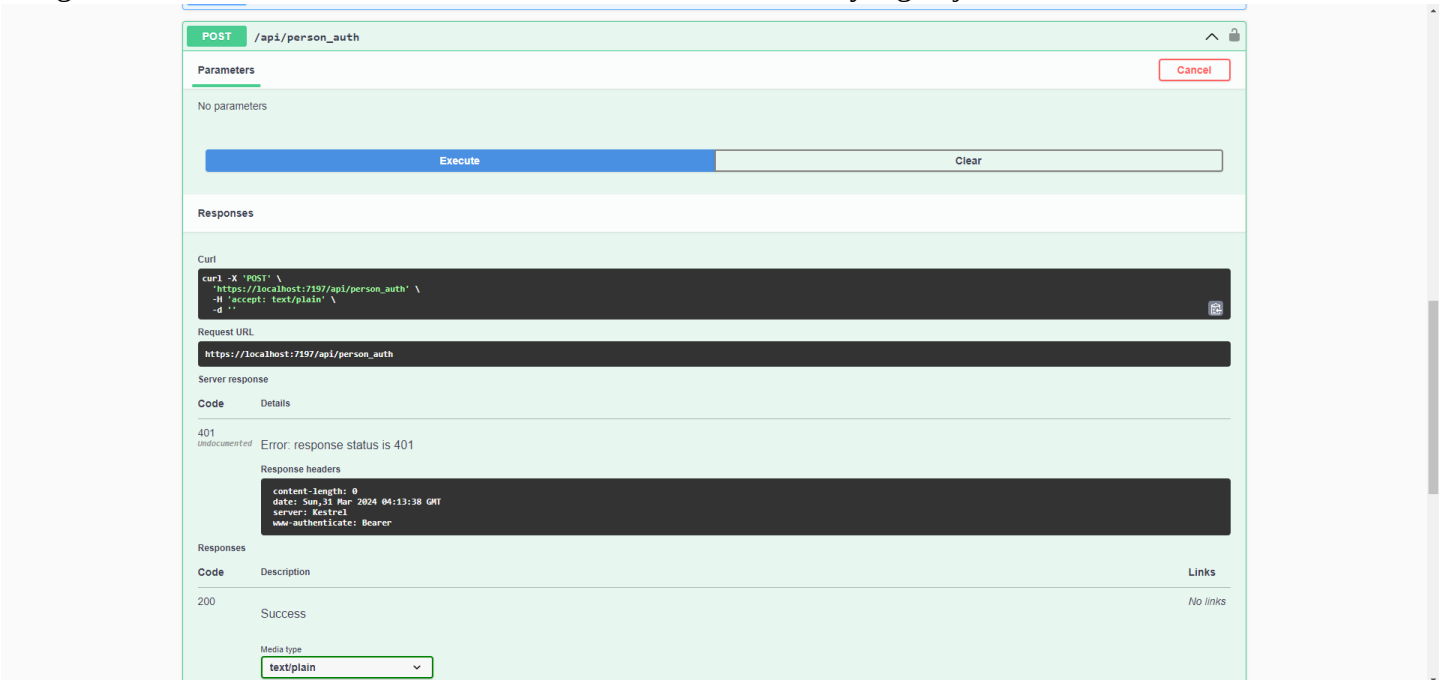
Pada percobaan satu, membuat autentikasi menggunakan Json Web Token, yang dimana nantinya Jwt ini akan berfungsi untuk mengamankan sebuah data, Jwt sendiri memiliki tiga bagian, yaitu header yang digunakan untuk menyimpan informasi terkait token dan algoritma yang digunakan, selanjutnya terdapat payload, yang diguanakan untuk menyimpan klaim yang berisi mengenai informasi pengguna dan otorisasi, dan yang terakhir adalah signature, yang digunakan untuk memverifikasi keaslian sebuah pesan dengan cara menggabungkan header, payload, dan secretkey.

Berikut merupakan percobaan yang berhasil dilakukan

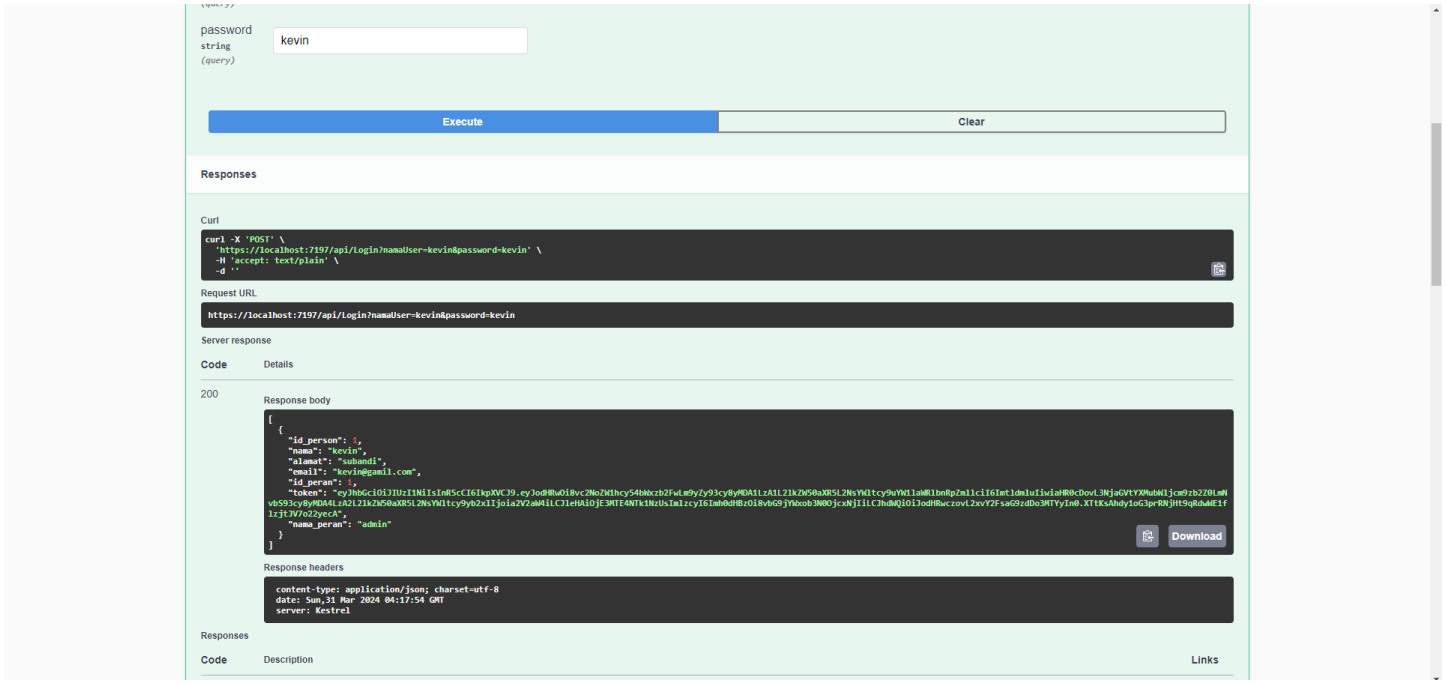
	id_person [PK] integer	nama text	alamat text	email text	username text	password text
1	1	kevin	subandi	kevin@gamil.com	kevin	kevin

,

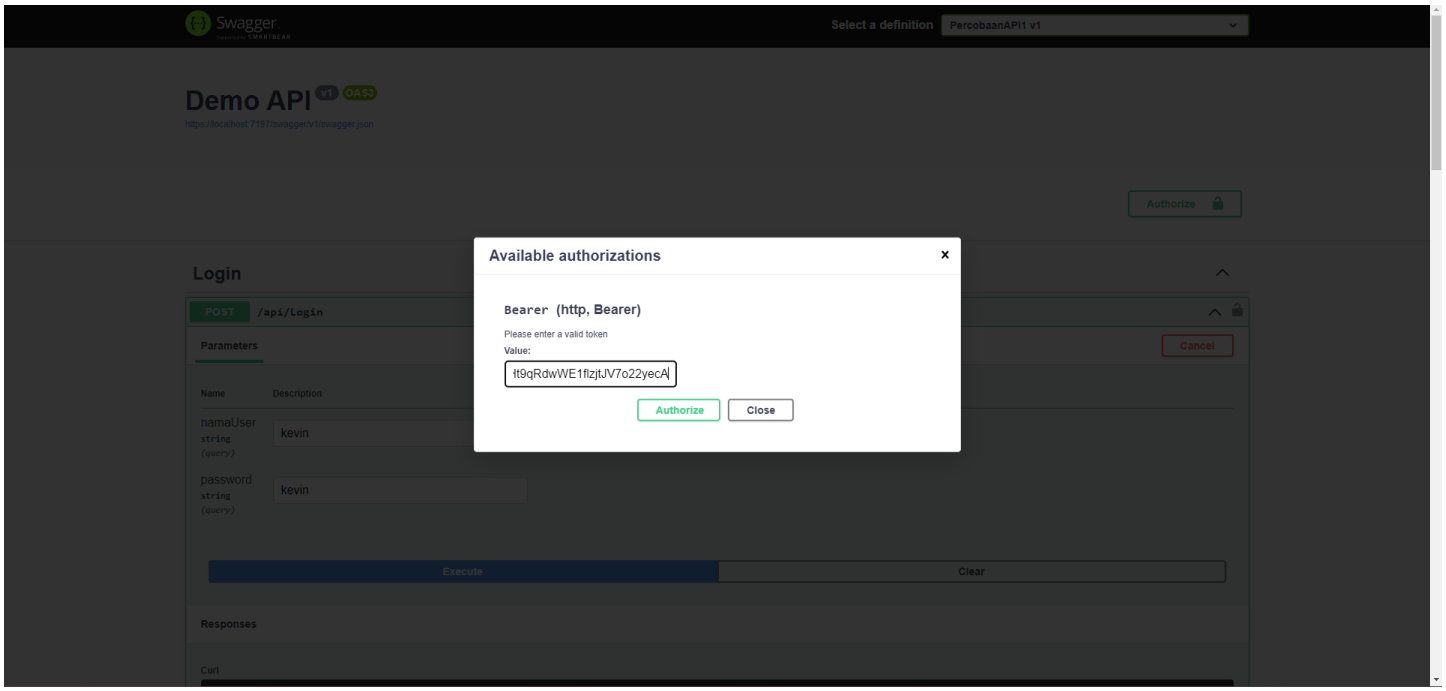
Disini saya memiliki data seperti ini, kita belum login kita tidak akan mendapatkan sebuah token, token tersebut berfungsi sebagai kunci autentikasi pada sebuah method, bisa dilihat gambar di bawah ini, jika kita akan mengambil data dari database, dan tidak memasukan token, maka yang terjadi adalah error.



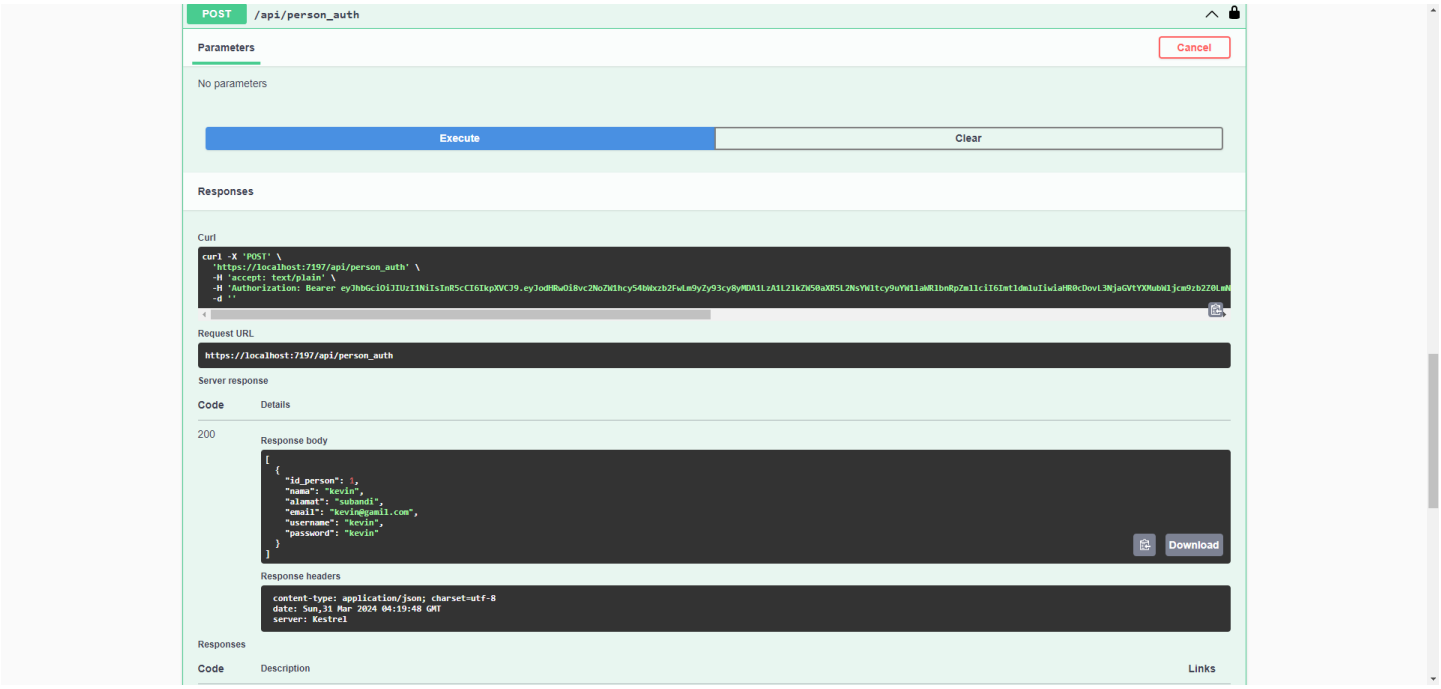
Kita harus login terlebih dahulu



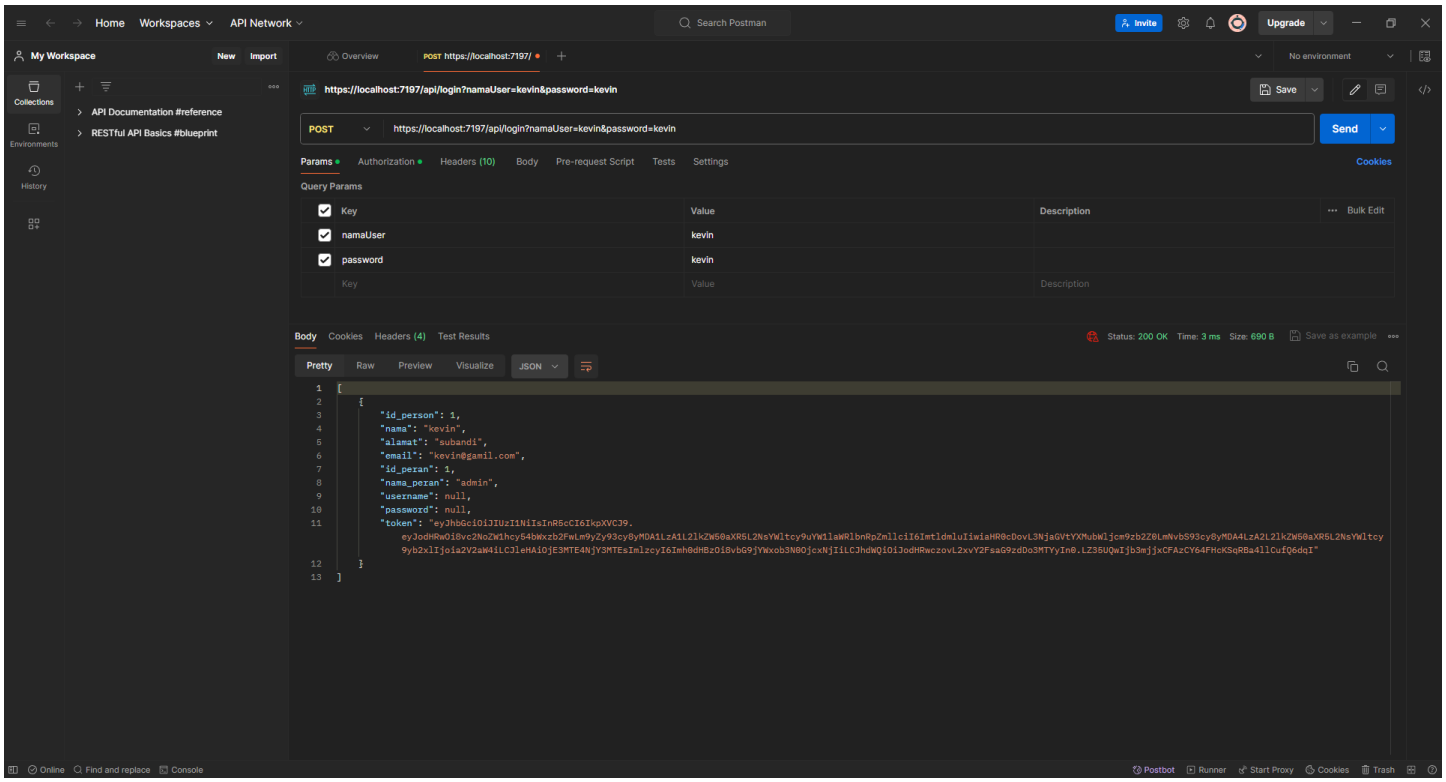
Jika kita telah berhasil login, maka sistem akan memberikan respon berupa json, yang berisikan token, sebagai berikut, dan selanjutnya kita harus memasukan token tersebut untuk mendapatkan akses



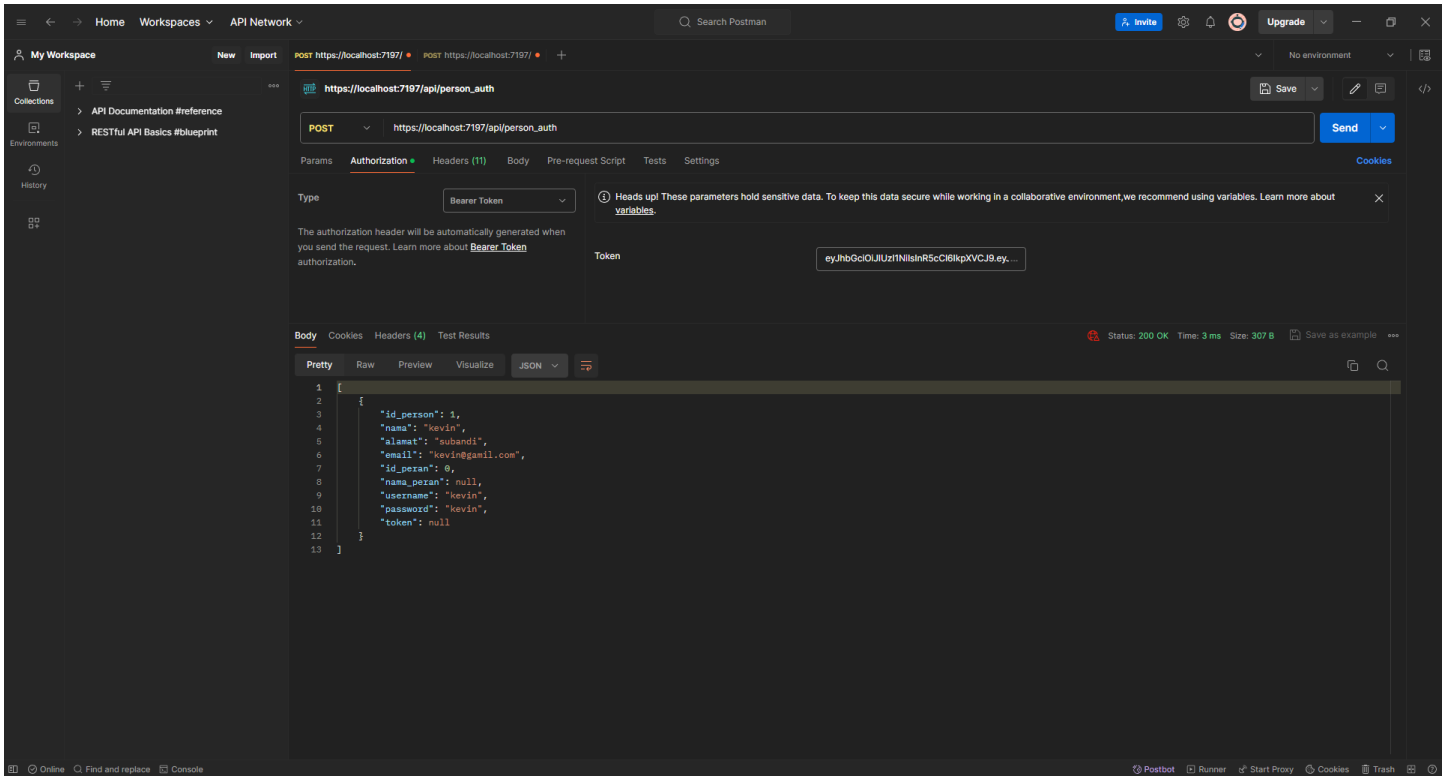
Selanjutnya klik authorize, sehingga kita dapat mengambil data



Begitupun pada tampilan Postman, berikut jika dijalankan pada Postman



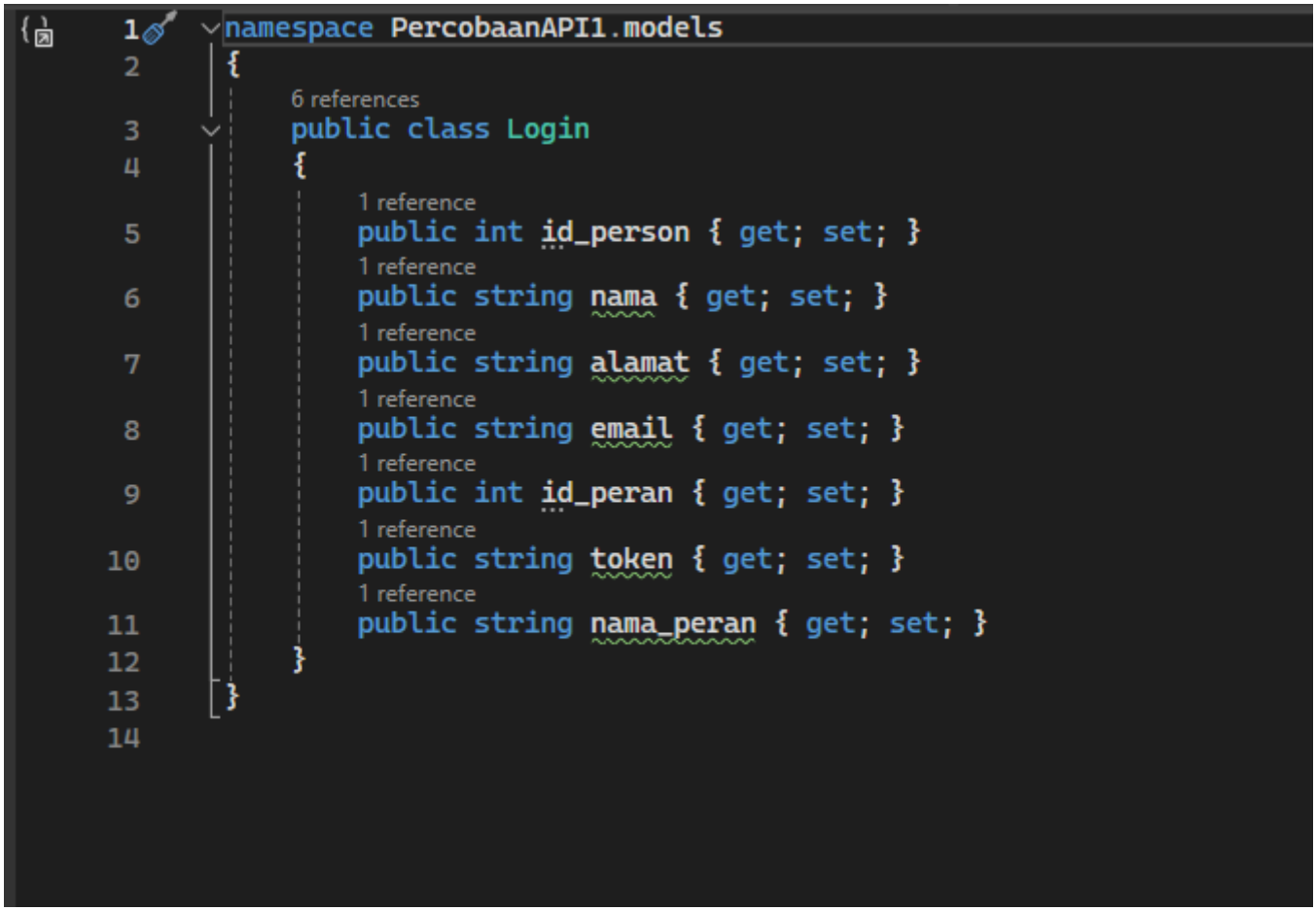
Dan sistem akan otomatis terbuat



lalu selanjutnya kita bisa login menggunakan token yang telah dibuat tadi

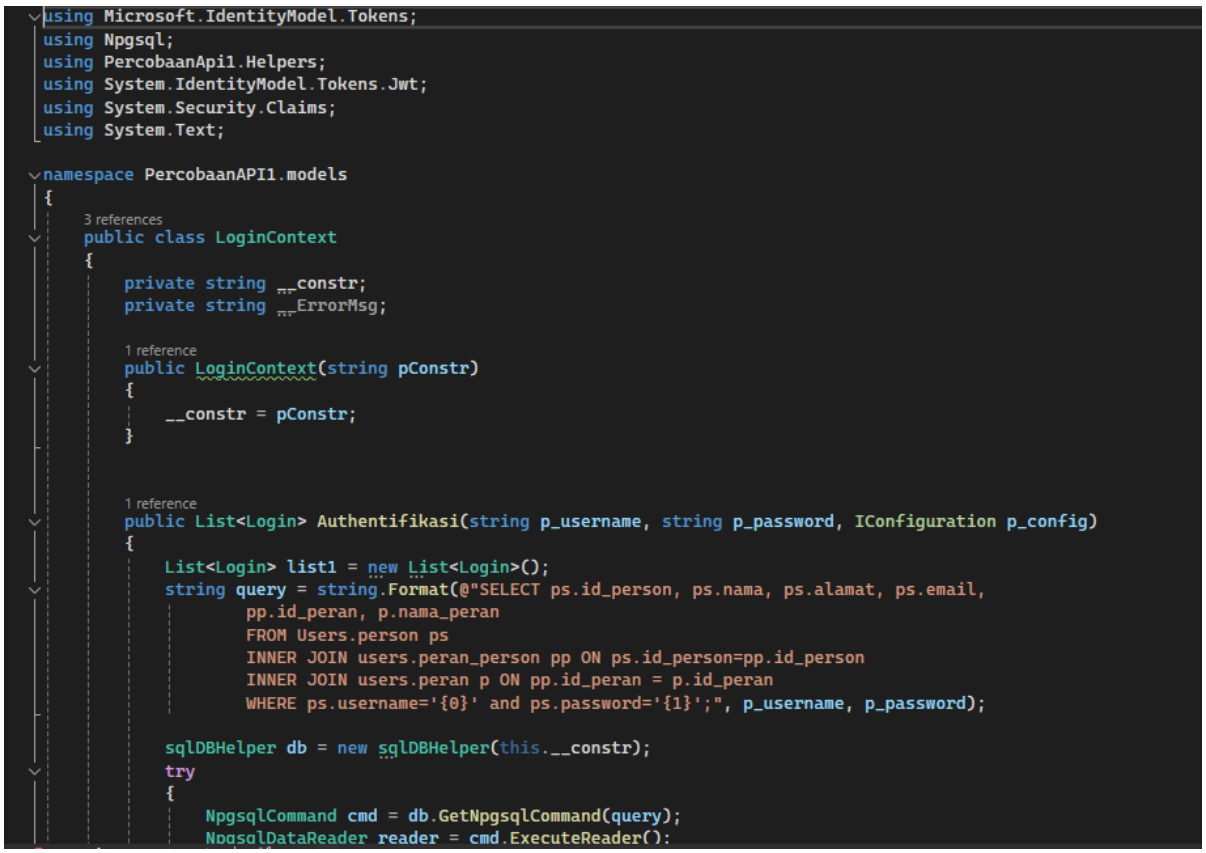
Kesimpulan:
Jadi Autentikasi Jwt memiliki peran yang penting dalam transfer data, yang dimana nantinya sistem akan membuat sebuah token, yang bisa kita gunakan untuk mengambil sebuah data, data lebih aman dan keasliannya dapat dipertanggungjawabkan

Tugas Mandiri :
Gunakan kembali program yang telah kalian buat pada modul 2 dan tambahkan fitur login (1 user saja untuk admin) dan implementasikan JWT sebagai metode auth.



```
namespace PercobaanAPI1.models
{
    6 references
    public class Login
    {
        1 reference
        public int id_person { get; set; }
        1 reference
        public string nama { get; set; }
        1 reference
        public string alamat { get; set; }
        1 reference
        public string email { get; set; }
        1 reference
        public int id_peran { get; set; }
        1 reference
        public string token { get; set; }
        1 reference
        public string nama_peran { get; set; }
    }
}
```

Pertama yang harus dilakukan adalah, kita membuat class login yang menginisiasikan data seperti gambar diatas



```
using Microsoft.IdentityModel.Tokens;
using Npgsql;
using PercobaanApi1.Helpers;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace PercobaanAPI1.models
{
    3 references
    public class LoginContext
    {
        private string __constr;
        private string __ErrorMsg;

        1 reference
        public LoginContext(string pConstr)
        {
            __constr = pConstr;
        }


        1 reference
        public List<Login> Authentifikasi(string p_username, string p_password, IConfiguration p_config)
        {
            List<Login> list1 = new List<Login>();
            string query = string.Format(@"SELECT ps.id_person, ps.nama, ps.alamat, ps.email,
                    pp.id_peran, p.nama_peran
                    FROM Users.person ps
                    INNER JOIN users.peran_person pp ON ps.id_person=pp.id_person
                    INNER JOIN users.peran p ON pp.id_peran = p.id_peran
                    WHERE ps.username='{0}' and ps.password='{1}';", p_username, p_password);

            sqlDBHelper db = new sqlDBHelper(this.__constr);
            try
            {
                NpgsqlCommand cmd = db.GetNpgsqlCommand(query);
                NpgsqlDataReader reader = cmd.ExecuteReader();
```

Selanjutnya terdapat class yang bernama LoginContex yang dimana class ini memiliki method LoginContex yang membawa parameter pConstr yang bertipe data string, lalu didalamnya terdapat informasi koneksi ke database, lalu terdapat method dengan nama Authentifikasi yang digunakan untuk melakukan validasi data pengguna dengan menjalankan query SQL, method ini menerima parameter password dan juga username dan menerima parameter p_config, yang digunakan untuk mendapatkan token JWT, lalu query yang telah di

eksekusi menggunakan kelas NpgSqlCommand dimasukan kedalam list Login yang bersamaan dengan dibuatnya token JWT dengan perintah GenerateJwtToken yang tersimpan kedalam variabel token

```
35                  NpgsqlDataReader reader = cmd.ExecuteReader();
36                  while (reader.Read())
37                  {
38                      list1.Add(new Login()
39                      {
40                          id_person = int.Parse(reader["id_person"].ToString()),
41                          nama = reader["nama"].ToString(),
42                          alamat = reader["alamat"].ToString(),
43                          email = reader["email"].ToString(),
44                          id_peran = int.Parse(reader["id_peran"].ToString()),
45                          nama_peran = reader["nama_peran"].ToString(),
46                          token = GenerateJwtToken(p_username, p_password, p_config)
47                      });
48                  }
49                  cmd.Dispose();
50                  db.closeConnection();
51              }
52              catch (Exception ex)
53              {
54                  __ErrorMsg = ex.Message;
55              }
56              return list1;
57
58          }
59
59
                1 reference
60          private string GenerateJwtToken(string namaUser, string peran, IConfiguration pConfig)
61          {
62              var securityKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(pConfig["Jwt:Key"]));
63              var credential = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);
64              var claims = new[]
65              {
66                  new Claim(ClaimTypes.NameIdentifier, namaUser),
67                  new Claim(ClaimTypes.Role, peran),
68              };
69              var token = new JwtSecurityToken(pConfig["Jwt:Issuer"],
70                  pConfig["Jwt:Audience"],
71                  claims,
72                  expires: DateTime.Now.AddMinutes(15),
73                  signingCredentials: credential);
74              return new JwtSecurityTokenHandler().WriteToken(token);
75          }
76
77
78      }
79  }
80
```

Pada method GenerateJwtToken, menerima parameter namaUser, peran, dan juga pConfig, didalam method ini, pada line 62 merupakan kode untuk membuat sebuah objek baru, yang digunakan untuk menandai token Jwt yang di mana, kunci ini diambil dari Jwt:Key, lalu untuk menentukan metode bagaimana token akan ditandatangani, menggunakan perintah SigningCredentials dengan mengguanakn algortima HmacSha256

```
1  {
2      "ConnectionStrings": {
3          "webApiDatabase": "Host=127.0.0.1; Port= 5432; Database=Users; Username=postgres; Password=timotius"
4      },
5
6      "Logging": {
7          "LogLevel": {
8              "Default": "Information",
9              "Microsoft.AspNetCore": "Warning"
10         }
11     },
12
13     "Jwt": {
14         "Key": "sjenfojunasoanjwn334n4j32n4jn2kj3224e2kb4b@#@(Y$B@K$",
15         "Issuer": "https://localhost:7162",
16         "Audience": "https://localhost:7162"
17     },
18     "AllowedHosts": "*"
19 }
20
```

```
using Microsoft.AspNetCore.Mvc;
using PercobaanAPI1.models;

namespace PercobaanAPI1.Controllers
{
    1 reference
    public class LoginController : Controller
    {

        private string __constr;
        private IConfiguration __config;

        0 references
        public LoginController(IConfiguration configuration)
        {
            __config = configuration;
            __constr = configuration.GetConnectionString("WebApiDatabase");
        }
        0 references
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost("api/Login")]
        0 references
        public IEnumerable<Login> LoginUser(string namaUser, string password)
        {
            LoginContext context = new LoginContext(this.__constr);
            List<Login> listlogin = context.Authentifikasi(namaUser, password, __config);
            return (listlogin).ToArray();
        }
    }
}
```

Di dalam kelas LoginController terdapat dua atribut yang bersifat privat yaitu __constr yang digunakan untuk menyimpan informasi koneksi ke database, dan __config digunakan untuk mengambil konfigurasi aplikasi yang disimpan pada appsettings.json. lalu terdapat method LoginController yang digunakan untuk menginisiasikan informasi tentang database.

Terdapat method dengan nama index, yang digunakan untuk menangani perintah HTTP GET, dan yang terakhir, terdapat pendefinisian route yaitu [HttpPost("api/Login")] yang di mana berfungsi untuk menangani permintaan HTTP POST yang diterima pada rute api/Login lalu pada method yang bernama LoginUser. Terdapat method Authentifikasi dari LoginContex yang memiliki parameter namaUser, password, dan konfigurasi __config, hasil dari autentifikasi login akan dikembalikan dalam bentuk array

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using PercobaanApi1.Models;

namespace PercobaanAPI1.Controllers
{
    1 reference
    public class PersonController : Controller
    {
        private string __constr;

        0 references
        public PersonController(IConfiguration configuration)
        {
            __constr = configuration.GetConnectionString("webApiDatabase");
        }

        0 references
        public IActionResult Index()
        {
            return View();
        }

        [HttpGet("api/person"), Authorize]

        0 references
        public ActionResult<Person> ListPerson()
        {
            PersonContext contex = new PersonContext (this.__constr);
            List<Person> ListPerson = contex.ListPerson();
            return Ok(ListPerson);
        }



        [HttpPost("api/person/create"), Authorize]
        0 references
        public ActionResult AddPerson(Person person)
        {
            try
            {
                PersonContext context = new PersonContext(__constr);
                context.AddPerson(person);
                return Ok("Berhasil menambahkan data");
            }
            catch (Exception ex)
            {
                return BadRequest("Gagal menambahkan data: " + ex.Message);
            }
        }
```

```
        [HttpPatch("api/person/update/{id_person}"), Authorize]
        0 references
        public ActionResult UpdatePerson(int id_person, Person person)
        {
            try
            {
                PersonContext context = new PersonContext(__constr);
                context.UpdatePerson(id_person, person);
                return Ok("Berhasil Mengubah Data {id_person}");
            }
            catch
            {
                return BadRequest("Gagal Mengubah Data");
            }
        }


        [HttpDelete("api/person/delete/{id_person}"), Authorize]
        0 references
        public ActionResult DeletePerson(int id_person)
        {
            try
            {
                PersonContext context = new PersonContext(__constr);
                context.DeletePerson(id_person);
                return Ok("Data Berhasil Dihapus");
            }
            catch
            {
                return BadRequest();
            }
        }



    }
}
```

Authorize

Login

POST /api/Login

Person

GET /api/person

POST /api/person/create

PATCH /api/person/update/{id_person}

DELETE /api/person/delete/{id_person}

WeatherForecast

GET /WeatherForecast

Parameters

Cancel

Name	Description
namaUser string (query)	kevin
password string (query)	kevin

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'https://localhost:7197/api/Login?namaUser=kevin&password=kevin' \
  -H 'accept: text/plain' \
  -d ''
```

Request URL

```
https://localhost:7197/api/Login?namaUser=kevin&password=kevin
```

Server response

Code	Details
200	Response body


```
[
  {
    "id_person": 1,
    "nama": "kevin",
    "alamat": "subandi",
    "email": "kevin@gamil.com",
    "id_peran": 1,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6ImIdmluIiwiaHR0cDovL3MjaGVtYXMubWljcm9zb2Z0LmNvb593cy8yMDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2xlIjoia2V2aW4iLCJleHAiOjE3MTE4NzU3NjQsImlzcyI6Imh0dHBzOi8vbG9jYWxob3N0OjcxNjciLCJhdWQiOiJodHRwczovL2xvY2FsaG9zdDo3MTYyIn0.9nDTvJd_w3MEg7slmQaqRwAsuL9AvrV0mQKga76gAWA",
    "nama_peran": "admin"
  }
]
```


Download