

Nama: Kevin Elvio C. P

NIM: 222410102088

Link Github: <https://github.com/KevinElvio/Pemrograman-Antarmuka-Aplikasi.git>

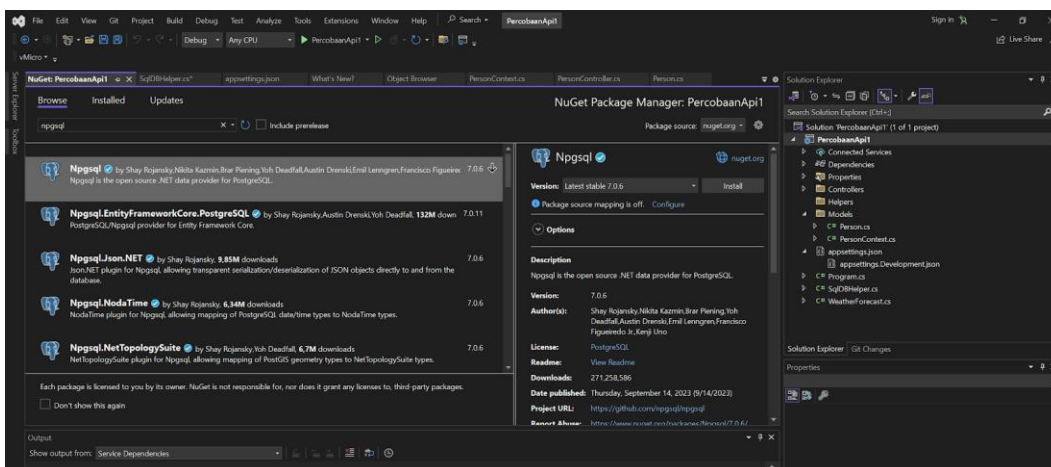
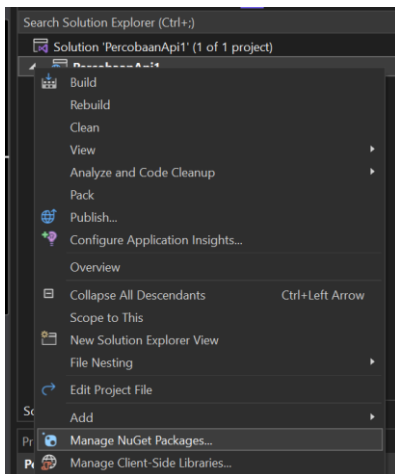
Modul 2

AKSES DATABASE POSTGRESQL

Percobaan 1: Membuat Tabel Database dan menampilkan pada API

Langkah-langkah percobaan:

1. Install database postgresql dan pgAdmin-4
2. Buatlah schema “users” kemudian buatlah dalam schema users tersebut sebuah table dengan nama tabel “person”, dengan kolom/field “id_person”, “nama”, “alamat”, “email”. Insert beberapa data ke dalam table “person”.
3. Install npgsql dari Nuget dengan klik kanan pada proyek VS2022, lalu pilih “Manage Nuget Library”, pilih “Browse” dan cari npgsql.



4. Buka file “appSetting.json” pada proyek dan ubahlah isinya untuk dapat terkoneksi dengan Postgresql seperti contoh di bawah ini dengan penyesuaian untuk komputer masing-masing.

```

1 https://json.schemastore.org/appsettings.json
2 {
3   "ConnectionStrings": {
4     "WebApiDatabase": "Host=127.0.0.1; Port=5432; Database=pbo2023; Username=istiyadi; Password=123456"
5   },
6   "Logging": {
7     "LogLevel": {
8       "Default": "Information",
9       "Microsoft.AspNetCore": "Warning"
10    }
11  },
12  "AllowedHosts": "*"
13 }

```

5. Tambahkan file "sqlDBHelper.cs" pada folder "Helpers" yang isinya adalah sebagai berikut:

```

1 using Npgsql;
2 using System.Data;
3
4 namespace PercobaanApi1.Helpers
5 {
6     3 references
7     public class SqlDBHelper
8     {
9         private NpgsqlConnection connection;
10        private string __constr;
11        1 reference
12        public SqlDBHelper(string pConstr)
13        {
14            __constr = pConstr;
15            connection = new NpgsqlConnection();
16            connection.ConnectionString = __constr;
17        }
18        1 reference
19        public NpgsqlCommand getNpgsqlCommand(string query)
20        {
21            connection.Open();
22            NpgsqlCommand cmd = new NpgsqlCommand();
23            cmd.Connection = connection;
24            cmd.CommandText = query;
25            cmd.CommandType = CommandType.Text;
26            return cmd;
27        }
28        1 reference
29        public void closeConnection()
30        {
31            connection.Close();
32        }
33    }
34 }

```

6. Ubah file "PersonContext.cs" menjadi sebagai berikut:

```

1  using Npgsql;
2  using PercobaanApi1.Helpers;
3
4  namespace PercobaanApi1.Models
5  {
6      3 references
7      public class PersonContext
8      {
9          private string __constr;
10         private string __ErrorMsg;
11         1 reference
12         public PersonContext(string pConstr)
13         {
14             __constr = pConstr;
15         }
16         1 reference
17         public List<Person> ListPerson()
18         {
19             List<Person> list1 = new List<Person>();
20             string query = string.Format(@"SELECT id_person, nama, alamat,
21             email FROM users.person;");
22             SqlDBHelper db = new SqlDBHelper(this.__constr);
23             try
24             {
25                 NpgsqlCommand cmd = db.GetNpgsqlCommand(query);
26                 NpgsqlDataReader reader = cmd.ExecuteReader();
27                 while (reader.Read())
28                 {
29                     list1.Add(new Person()
30                     {
31                         id_person = int.Parse(reader["id_person"].ToString()),
32                         nama = reader["nama"].ToString(),
33                         alamat = reader["alamat"].ToString(),
34                         email = reader["email"].ToString()
35                     });
36                 }
37                 cmd.Dispose();
38                 db.closeConnection();
39             }
40             catch (Exception ex)
41             {
42                 __ErrorMsg = ex.Message;
43             }
44             return list1;
45         }
46     }
47 }

```

7. Ubah file “PersonController.cs” menjadi sebagai berikut:

```

1  using Microsoft.AspNetCore.Authorization;
2  using Microsoft.AspNetCore.Mvc;
3  using PercobaanApi1.Models;
4
5  namespace PercobaanApi1.Controllers
6  {
7      1 reference
8      public class PersonController : Controller
9      {
10         private string __constr;
11         0 references
12         public PersonController(IConfiguration configuration)
13         {
14             __constr = configuration.GetConnectionString("WebApiDatabase");
15         }
16         0 references
17         public IActionResult Index()
18         {
19             return View();
20         }
21         [HttpGet("api/person")]
22         0 references
23         public ActionResult<Person> ListPerson()
24         {
25             PersonContext context = new PersonContext(this.__constr);
26             List<Person> ListPerson = context.ListPerson();
27             return Ok(ListPerson);
28         }
29     }
30 }

```

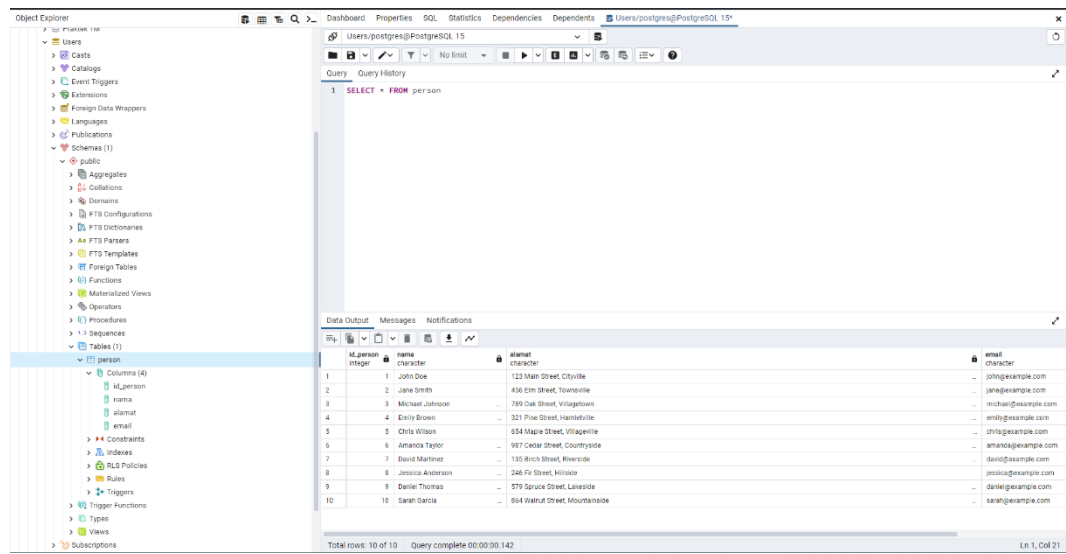
8. Jalankan aplikasi dan tampilkan hasilnya.

HASIL DAN ANALISIS DATA

Percobaan 1: Membuat Tabel Database dan menampilkan pada API

Hasil:

- Berhasil menampilkan terminal untuk memonitoring aplikasi
- Berhasil menampilkan data dari database dan ditampilkan di browser dan Postman

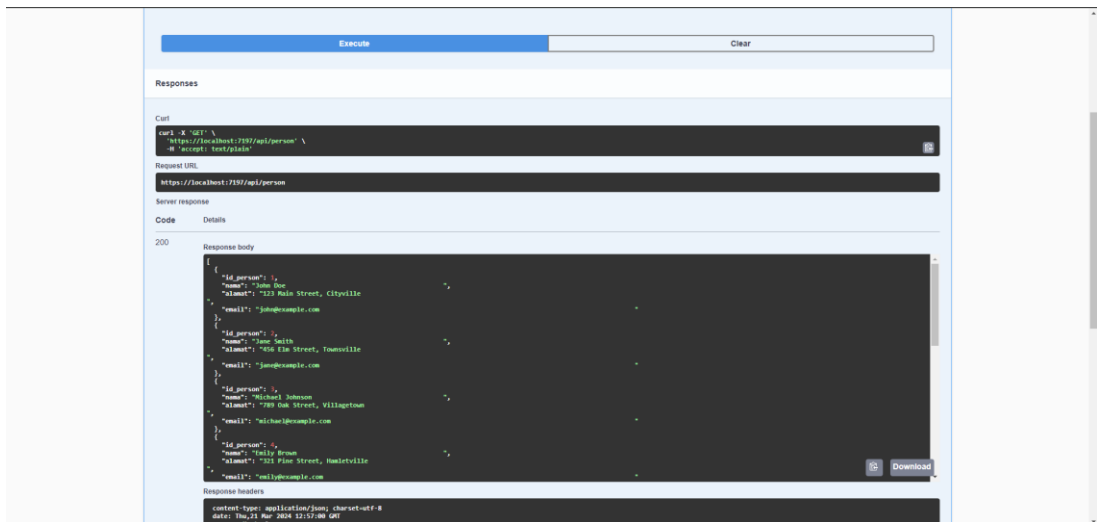


The screenshot shows the PostgreSQL Object Explorer on the left, displaying the 'person' table under the 'public' schema. The main window shows the SQL query editor with the following query:

```
SELECT * FROM person
```

The Data Output pane displays the results of the query, showing 10 rows of data. The columns are 'id_person' (integer), 'name' (character), 'alamat' (character), and 'email' (character).

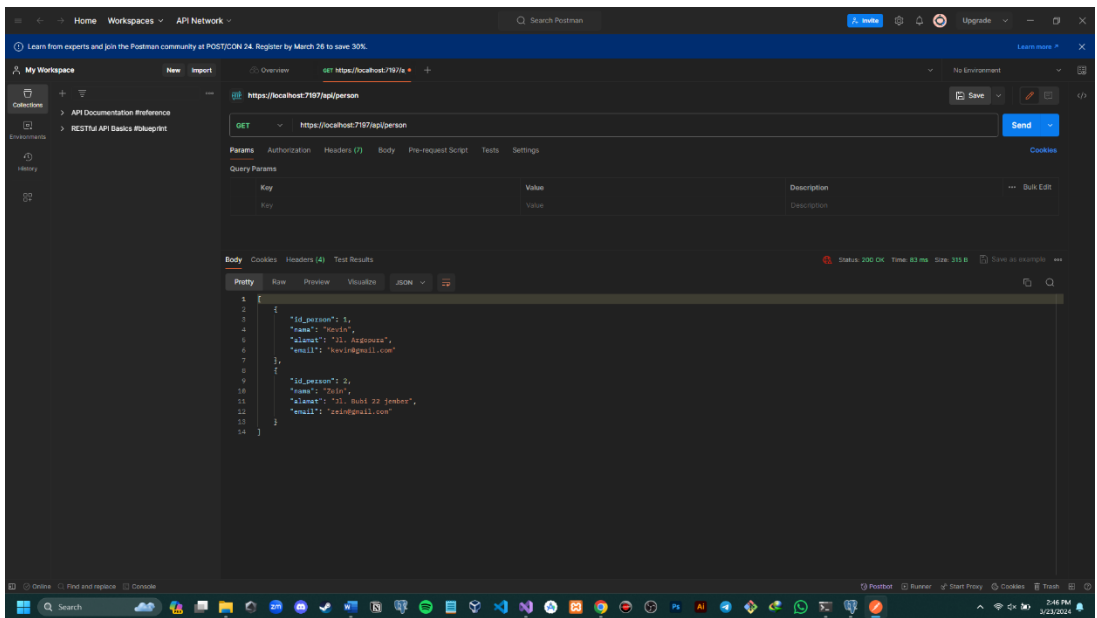
id_person	name	alamat	email
1	John Doe	123 Main Street, Cityville	john@example.com
2	Jane Smith	456 Elm Street, Townsville	jane@example.com
3	Michael Johnson	789 Oak Street, Villagetown	michael@example.com
4	Emily Brown	321 Pine Street, Hamletville	emily@example.com
5	Chris Wilson	654 Maple Street, Villagetown	chris@example.com
6	Amanda Taylor	987 Cedar Street, Countyside	amanda@example.com
7	David Martinez	135 Birch Street, Riverside	david@example.com
8	Jessica Anderson	246 Fir Street, Hillside	jessica@example.com
9	Daniel Thomas	578 Spruce Street, Lakeside	daniel@example.com
10	Sarah Garcia	894 Walnut Street, Mountside	sarah@example.com



The screenshot shows a web browser displaying the response of a GET request to <https://localhost:7197/api/person>. The response is a JSON array of person objects, with a status code of 200.

```
curl -X GET "https://localhost:7197/api/person" -H "accept: text/plain"
```

```
{
  "id_person": 1,
  "name": "John Doe",
  "alamat": "123 Main Street, Cityville",
  "email": "john@example.com",
  "id_person": 2,
  "name": "Jane Smith",
  "alamat": "456 Elm Street, Townsville",
  "email": "jane@example.com",
  "id_person": 3,
  "name": "Michael Johnson",
  "alamat": "789 Oak Street, Villagetown",
  "email": "michael@example.com",
  "id_person": 4,
  "name": "Emily Brown",
  "alamat": "321 Pine Street, Hamletville",
  "email": "emily@example.com",
  "id_person": 5,
  "name": "Chris Wilson",
  "alamat": "654 Maple Street, Villagetown",
  "email": "chris@example.com",
  "id_person": 6,
  "name": "Amanda Taylor",
  "alamat": "987 Cedar Street, Countyside",
  "email": "amanda@example.com",
  "id_person": 7,
  "name": "David Martinez",
  "alamat": "135 Birch Street, Riverside",
  "email": "david@example.com",
  "id_person": 8,
  "name": "Jessica Anderson",
  "alamat": "246 Fir Street, Hillside",
  "email": "jessica@example.com",
  "id_person": 9,
  "name": "Daniel Thomas",
  "alamat": "578 Spruce Street, Lakeside",
  "email": "daniel@example.com",
  "id_person": 10,
  "name": "Sarah Garcia",
  "alamat": "894 Walnut Street, Mountside",
  "email": "sarah@example.com"
}
```



The screenshot shows Postman displaying a GET request to <https://localhost:7197/api/person>. The response is a JSON array of person objects, with a status code of 200 OK.

```
GET https://localhost:7197/api/person
```

```
{
  "id_person": 1,
  "name": "John Doe",
  "alamat": "123 Main Street, Cityville",
  "email": "john@example.com",
  "id_person": 2,
  "name": "Jane Smith",
  "alamat": "456 Elm Street, Townsville",
  "email": "jane@example.com",
  "id_person": 3,
  "name": "Michael Johnson",
  "alamat": "789 Oak Street, Villagetown",
  "email": "michael@example.com",
  "id_person": 4,
  "name": "Emily Brown",
  "alamat": "321 Pine Street, Hamletville",
  "email": "emily@example.com",
  "id_person": 5,
  "name": "Chris Wilson",
  "alamat": "654 Maple Street, Villagetown",
  "email": "chris@example.com",
  "id_person": 6,
  "name": "Amanda Taylor",
  "alamat": "987 Cedar Street, Countyside",
  "email": "amanda@example.com",
  "id_person": 7,
  "name": "David Martinez",
  "alamat": "135 Birch Street, Riverside",
  "email": "david@example.com",
  "id_person": 8,
  "name": "Jessica Anderson",
  "alamat": "246 Fir Street, Hillside",
  "email": "jessica@example.com",
  "id_person": 9,
  "name": "Daniel Thomas",
  "alamat": "578 Spruce Street, Lakeside",
  "email": "daniel@example.com",
  "id_person": 10,
  "name": "Sarah Garcia",
  "alamat": "894 Walnut Street, Mountside",
  "email": "sarah@example.com"
}
```

Analisis: Bisa dilihat bahwa untuk menampilkan sebuah data json di Swagger bisa menggunakan method get, yang dimana fungsi dari method ini digunakan untuk mengambil data, dalam kasus ini mengambil data dari database dengan tabel person dan memiliki coloum id_person, nama, alamat, email

Kesimpulan:

Kode ini merupakan implementasi API sederhana dengan fungsi CRUD (Create, Read, Update, Delete) menggunakan framework ASP.NET Core dan NuGet Npgsql untuk terhubung dengan database. Koneksi database tersebut berada pada file appsettings.json, dan diuji menggunakan Postman dan Swagger.

TUGAS MANDIRI

Buatlah API CRUD untuk melakukan manajemen data murid dalam suatu kelas yang mengimplementasikan koneksi database untuk menyimpan data.

1. Create

```
1 reference
public void AddPerson(Person person)
{
    string query = string.Format("INSERT INTO person (nama, alamat, email) VALUES (@nama, @alamat, @email)");
    sqlDBHelper db = new sqlDBHelper(this.__constr);

    try
    {
        using (NpgsqlCommand cmd = db.GetNpgsqlCommand(query))
        {
            cmd.Parameters.AddWithValue("@nama", person.nama);
            cmd.Parameters.AddWithValue("@alamat", person.alamat);
            cmd.Parameters.AddWithValue("@email", person.email);

            cmd.ExecuteNonQuery();
        }
    }
    catch (Exception ex)
    {
        __ErrorMsg = ex.Message;
    }
}
```

Pertama yang harus dilakukan adalah membuat method person dengan parameter Person person lalu di dalam method tersebut terdapat variabel query bertipe data string, lalu terdapat metode string.format yang digunakan untuk memformat nilai menjadi string. Terdapat code untuk membuat sebuah objek baru dari kelas sqlDBHelper yang bernama db yang memiliki parameter this.__constr, dalam parameter ini memiliki informasi yang di perlukan untuk database, lalu menggunakan try dan catch berfungsi untuk menampilkan pesan error

```
0 references
[HttpPost("api/person/create")]
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);
    }
}
```

Lalu kita bisa membuat code sebagai berikut pada controller, menggunakan method post untuk mengirimkan data dengan path api/person/create, lalu terdapat sebuah fungsi yang bernama AddPerson yang memiliki tipe data ActionResult dan memiliki parameter Person

POST /api/person/create

Parameters

Name	Description
id_person integer (int32) (query)	1
nama string (query)	Hari
alamat string (query)	Jl. Gatot Kaca
email string (query)	hari@gmail.com

Responses

201

	id_person [PK] integer	nama character varying	alamat character varying	email character varying
1	1	Kevin	Jl. Argopura	kevin@gmail.com
2	2	Zein	Jl. Bubi 22 jember	zein@gmail.com
3	3	Hari	Jl. Gatot Kaca	hari@gmail.com

2. Read

Read telah dijelaskan pada bagian analisis dan telah dijadikan percobaan pada percobaan1, menggunakan method get untuk mengambil data dari database

3. Update

Membuat fungsi pada personContext yang bernama UpdatePerson dan memiliki parameter id_person yang bertipe data integer dan person merupakan objek dari Person, fungsi dari id_person merupakan id yang nantinya akan menjadi acuan untuk memperbaharui sebuah data, lalu terdapat variabel query yang bertipe data string yang berisi data query yang diformat kedalam bentuk string. Terdapat objek baru yang bernama db yang berfungsi untuk koneksi pada database.

```

1 reference
public void UpdatePerson(int id_person, Person person)
{
    string query = string.Format(@"UPDATE person SET nama = @nama, alamat = @alamat, email = @email WHERE id_person = @id_person");
    sqlDBHelper db = new sqlDBHelper(this.__constr);

    using (NpgsqlCommand cmd = db.GetNpgsqlCommand(query))
    {
        cmd.Parameters.AddWithValue("@id_person", person.id_person);
        cmd.Parameters.AddWithValue("@nama", person.nama);
        cmd.Parameters.AddWithValue("@alamat", person.alamat);
        cmd.Parameters.AddWithValue("@email", person.email);

        cmd.ExecuteNonQuery();
    }
}

```

Lalu pada Person controller menggunakan method Patch yang dimana kegunaan dari method ini adalah untuk mengupdate data, sama seperti put tetapi patch dan put memiliki perbedaan, patch akan menghapus data yang lama dan menggantinya dengan data yang baru sedangkan put menempa data yang lama dengan data yang baru, sehingga untuk mengupdate data saya menggunakan patch

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

PATCH /api/person/update/{id_person}

Parameters

Name	Description
id_person * required Integer(\$int32) (path)	1
id_person Integer(\$int32) (query)	1
nama string (query)	Sule
alamat string (query)	Jl. Bagaskara
email string (query)	sule@gmail.com

Execute **Clear**

Responses

Curl

```
curl -X 'PATCH' \
  'https://localhost:7197/api/person/update/1?id_person=1&nama=Sule&alamat=Jl. Bagaskara&email=sule@gmail.com' \
  -H 'Accept: */*' -H 'Content-Type: application/json'
```

Request URL

```
https://localhost:7197/api/person/update/1?id_person=1&nama=Sule&alamat=Jl. Bagaskara&email=sule@gmail.com
```

Server response

	id_person [PK] integer	nama character varying	alamat character varying	email character varying
1	2	Zein	Jl. Bubi 22 jember	zein@gmail.com
2	3	Hari	Jl. Gatot Kaca	hari@gmail.com
3	1	Sule	Jl. Bagaskara	sule@gmail.com

4. Delete

Pada personContext terdapat fungsi yang bernama DeletePerson dan memiliki parameter id_person yang bertipe data integer, pada fungsi DeletePerson berisi query yang berfungsi untuk menghapus sebuah data dari database

```
1 reference
public void DeletePerson(int id_person)
{
    string query = string.Format(@"DELETE FROM person WHERE id_person = @id_person");
    sqlDBHelper db = new sqlDBHelper(this.__constr);

    using (NpgsqlCommand cmd = db.GetNpgsqlCommand(query))
    {
        cmd.Parameters.AddWithValue("@id_person", id_person);
        cmd.ExecuteNonQuery();
    }
}
```



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

DELETE /api/person/delete/{id_person}

Parameters
 Cancel

Name	Description
id_person * required integer (5 int 32) (path)	1

Execute
Clear

Responses

Curl


```
curl -X 'DELETE' \
  'https://localhost:7197/api/person/delete/1' \
  -H 'accept: */*'

```

Request URL


```
https://localhost:7197/api/person/delete/1

```

Server response

Code	Details
200	Response headers <pre> content-length: 0 date: Sat, 23 Mar 2024 12:24:24 GMT server: Kestrel </pre>

Responses

Code	Description	Links
200	Success	No links

	id_person [PK] integer	nama character varying	alamat character varying	email character varying
1	2	Zein	Jl. Bubi 22 jember	zein@gmail.com
2	3	Hari	Jl. Gatot Kaca	hari@gmail.com