TUJUAN PRAKTIKUM

1. Mahasiswa mampu memahami interaksi database dengan framework
2. Mahasiswa mampu mengetahui penulisan query builder dalam interaksi database

TEORI SINGKAT

Query Builder merupakan salah satu cara untuk menjalankan query database dengan lebih mudah, Query Builder juga telah dilengkapi dengan fitur keamanan untuk mencegah terjadinya SQL Injextion (adalah sebuah aksi hacking yang dilakukan di aplikasi client dengan cara memodifikasi perintah SQL yang ada di memori aplikasi client). Selain itu kita dapat menggunakan query builder tanpa harus membuat model terlebih dahulu.

Sricpt query bulder cukup mudah dipahami, misalnya untuk menampilkan data dari tabel produk yang dibuat sebelumnya pertama-tama kita harus mendefinisikan penggunaan class dari QUERY BUILDER dengan cara menambahkan script **`use DB;`** pada bagian atas controller.

```
use Illuminate\Support\Facades\DB;
```

Dengan pendefinisian mengunakan script use DB; kita telah menambahkan class QUERY BUILDER ke dalam controller **resmhsController**. Lalu untuk menampilkan data dari tabel **mhs** yang telah dibuat sebelumnya bisa menggunakan script berikut:

```
DB::table('mhs')->get();
```

Pada script diatas kita akan mengambil data dari tabel mhs yang berada di dalam database websaya yang telah dibuat sebelumnya, adapun isi dari tabel mhs tersebut adalah sebagai berikut:

	nim	nama	prodi	angkatan
Edit Copy Delete	22030001	Nadia	D3TI	2022
Edit Copy Delete	22030002	Zahra	D3TI	2022

DATA DENGAN QUERY BUILDER

Untuk mempraktekan contoh dari latihan menampilkan data dengan QUERY BUILDER pertama kita harus memeriksa file **.env** apakah projek sudah terhubung dengan database atau belum. File ini terletak pada bagian luar projek laravel yang dibuat, dan pastikan pada bagian mysql sudah terkonfigurasi seperti pada gambar dibawah:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=websaya
DB_USERNAME=root
DB_PASSWORD=
```

Database yang akan kita gunakan adalah database dengan nama **websaya**, yang telah kita buat pada latihan pembuatan migration sebelumnya. Kemudian pada file **resmhsController.php** yang berada pada folder **app\Http\Controller** telah dibuat pada latihan sebelumnya buatlah satu fungsi yang bernama index() atau modifikasi jika sudah ada dan tambahkan script sebagai berikut:

```
public function index(){
    $title = "Mahasiswa WebSaya.Com";
    $slug = "mahasiswa";
    $mhs = "Nadia";
    $dataMhs = DB::table('mhs')->get();
    return view('resmhs.index',
            compact('mhs','title','slug','dataMhs'));
}
```

Fungsi index() diatas akan mengambil semua data yang ada pada tabel mhs dan kemudian akan mengirimkannya ke view **index.blade.php** yang ada di dalam folder **resources/views/resmhs**. Jadi pada view **index.blade.php** yang ada pada folder **resourcess/views/resmhs** (jika belum ada silahkan dibuat) modifikasi script nya menjadi seperti dibawah:

```
@extends('layouts.main')
@section('title',$title)
@section('content')
    <h2>Nama: {{ $mhs}}</h2>
    <table class="table">
        <thead>
          <tr>
            <th scope="col">NIM</th>
            <th scope="col">NAMA</th>
            <th scope="col">PRODI</th>
            <th scope="col">ANGKATAN</th>
          </tr>
        </thead>
        <tbody>
        @foreach ($dataMhs as $item)
        <tr>
            <td>{{ $item->nim }}</td>
            <td>{{ $item->nama }}</td>
            <td>{{ $item->prodi }} </td>
            <td>{{ $item->angkatan }} </td>
          </tr>
        @endforeach
    </tbody>
</table>
@endsection
```

Sebelum mencoba menjalankan skript diatas pertama-tama cobalah untuk memeriksa route yang ada pada file web.php yang ada pada folder **routes/web.php** dan periksa apakah route

Route::get('/resmhs',[resmhsController::class,'index']);

sudah didefinisikan atau belum. Jika sudah maka aktifkan server artisan pada cmd lalu akses **http://localhost:8000/resmhs** maka hasilnya akan nampak seperti pada gambar dibawah:

JOIN TABEL DENGAN QUERY BUILDER

Jika diperhatikan pada tampilan data mhs pada tabel diatas pada kolom prodi kita masih menampilkan kode prodi atau bukan nama program studi. Untuk memperbaikinya buatlah satu tabel baru dengan nama prodi menggunakan fasilitas migration pada Laravel dengan tahapan sebagai berikut:

1. Jalankan perintah artisan berikut pada cmd atau comand promt yang sudah terarah ke dalam directori projek laravel yang digunakan seperti pada contoh gambar:

```
php artisan make:migration create_prodi_table
```

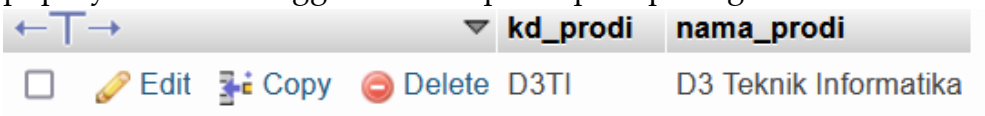
2. Langkah kedua bukalah file **2022_09_19_132052_create_prodi_table.php** yang ada folder database/migration dan modifikasi koding pada fungsi **up()** di dalamnya menjadi seperti dibawah:

```php
public function up()
{
    Schema::create('prodi', function (Blueprint $table) {
        $table->string('kd_prodi')->primary();
        $table->string('nama_prodi');
    });
}
```

3. Jalankan perintah php artisan migrate pada terminal seperti pada gambar dibawah.

```
php artisan migrate
```

4. Tambahkan beberapa data ke dalam tabel prodi secara manual melalui phpMyAdmin sehingga akan tampak seperti pada gambar dibawah:



Setelah membuat tabel prodi beserta dengan 1 contoh datanya sekarang bukalah class Controller **resmhsController.php** yang ada pada folder **app\Http\Controller** dan modifikasi fungsi **index()** yang ada di dalam controller tersebut menjadi seperti dibawah:

```php
public function index(){
    $title = "Mahasiswa WebSaya.Com";
    $slug = "mahasiswa";
    $mhs = "Nadia";
    $dataMhs = DB::table('mhs')
                ->join('prodi','mhs.prodi','=','prodi.kd_prodi')
                ->get();
    return view('resmhs.index',
            compact('mhs','title','slug','dataMhs'));
}
```

Kemudian pada view index.blade.php yang ada pada folder resources/views/resmhs/ modifikasi kodingnya menjadi seperti dibawah:

```
@extends('layouts.main')
@section('title',$title)
@section('content')
    <h2>Nama: {{ $mhs}}</h2>
    <table class="table">
        <thead>
          <tr>
            <th scope="col">NIM</th>
            <th scope="col">NAMA</th>
            <th scope="col">PRODI</th>
            <th scope="col">ANGKATAN</th>
          </tr>
        </thead>
        <tbody>
        @foreach ($dataMhs as $item)
        <tr>
            <td>{{ $item->nim }}</td>
            <td>{{ $item->nama }}</td>
            <td>{{ $item->nama_prodi }} </td>
            <td>{{ $item->angkatan }} </td>
          </tr>
        @endforeach
    </tbody>
</table>
@endsection
```

Jika sudah maka aktifkan server artisan pada cmd lalu akses http://localhost:8000/resmhs maka hasilnya akan nampak seperti pada gambar dibawah:

INTERAKSI DATABASE DENGAN QUERY BUILDER

Selain menggunakan cara diatas, ada berbagai variasi dalam menampilkan data menggunakan query builder. Berikut ini adalah berbagai script untuk menampilkan data dengan Query Builder:

QUERY BUILDER	KETERANGAN
count();	Menampilkan jumlah data
max('nama_kolom');	Menampilkan nilai maksiman dari sebuah kolom. Dengan cara yang sama juga kita dapat menggunakan min(), avg(), sum().
select('kolom1', 'kolom2', 'kolom3 as kolomketiga', 'kolom ke n')->get();	Memilih beberapa kolom dan membuat alias untuk kolom.
select(db::raw('count(*) as total'))->get();	Menggunakan row expression.
where('nama_kolom','=','value')	Menampilkan data dengan kondisi tertentu, selain menggunakan = kit juga dapat menggunakan >, <, >=, <=, <> dan like
where(['kolom1', '>', 'value'], ['kolom2', '<', 'value'])->get();	Menampilkan data dengan beberapa kondisi.
where('kolom1', '=', 'value')->orwhere ('kolom2', '=', 'value')->get();	Menggunakan or dalam menampilkan data dengan beberapa kondisi.

INTERAKSI DATABASE DENGAN QUERY BUILDER

QUERY BUILDER	KETERANGAN
whereBetween('nama_kolom',[parameter1, parameter2])->get();	Menampilkan data dengan nilai suatu kolom tidak di antara 2 batas nilai.
wherein('nama_kolom', [parameter1 ',' parameter2 ',' parameter3])->get();	Menampilkan data dengan nilai suatu kolom sesuai yang disebutkan. Untuk kebalikannya dapat menggunakan whereNotIn().
whereNull('nama_kolom')->get();	Menampilkan data sengan nilai suatu kolom null. Untuk kebalikannya dapat menggunakan WhereNotNull()
whereDate('nama_kolom_tanggal', '2017-12-12')->get();	Menampilkan data dengan membandingkan nilai tanggal. Juga terdapat pilihan lain seperti whereMonth(), whereYear(), dan whereDay()
whereColumns('kolom1', '<', 'kolom2')->get();	Menampilkan data dengan membandinkan dua kolom
orderBy('nama_kolom', 'desc')->get();	Menampilkan data dengan urutan
inRandomOrder()->get();	Menampilkan data dengan urutan acak
latest()->get();	Menampilkan data dengan data terbaru. Kebalikannya dapat menggunakan oldest()
groupby()->get();	Menampilkan data dengan menggunakan grup
limit(10)->offset(5)->get();	Menampilkan data dengan batas dan offset tertentu. Alternatif lain dapat menggunakan take() dan skip()
join('nama_table2', 'nama_tabel1.primary_key', '=', 'nama_tabel2.foreginkey')->get();	Menampilkan tabel dengan join
leftjoin('nama_table2', 'nama_tabel1.primary_key', '=', 'nama_tabel2.foreginkey')->get();	Menampilkan data dengan leftjoin, terdapat juga pilihan rightjoin()

MENAMBAH DATA DENGAN QUERY BUILDER

Bukalah file **resmhsController** yang ada pada folder **App/Http/Controller** dan tambahkan satu fungsi yang bernama **create()** dan isikan script sebagai berikut:

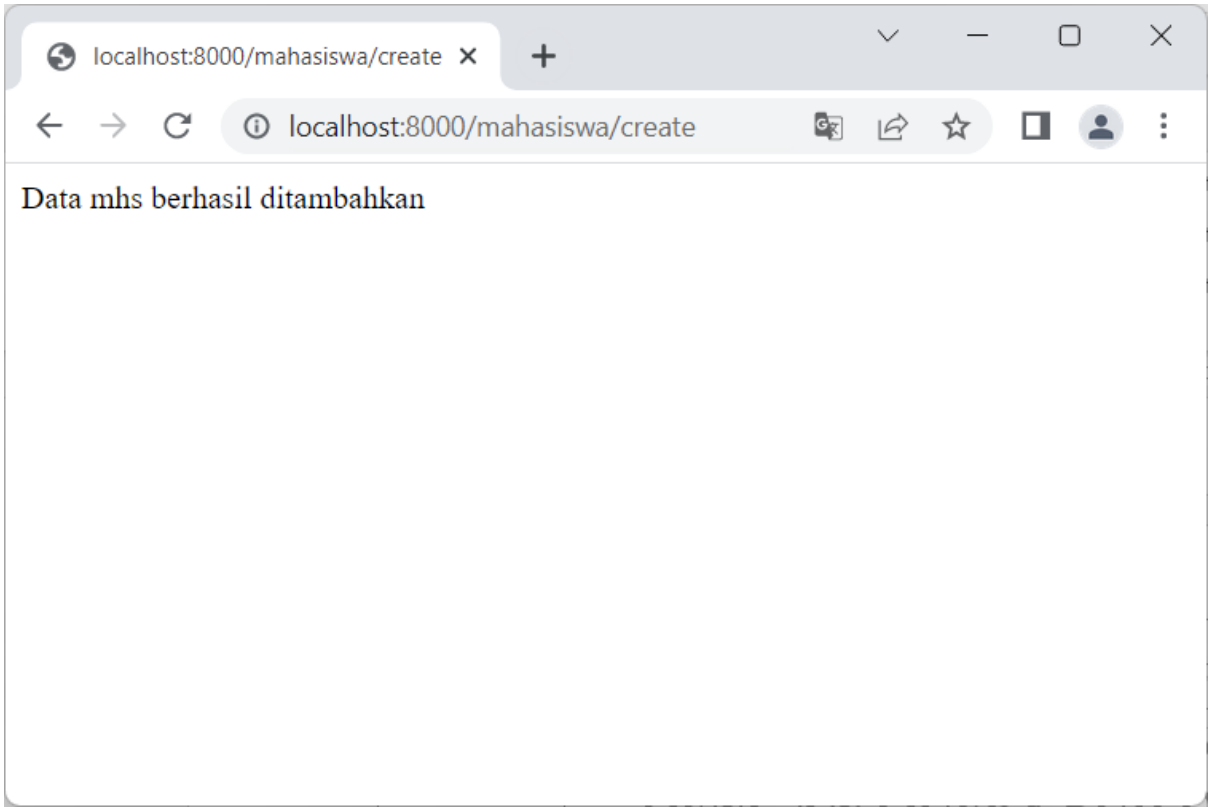
```
public function create()
{

    DB::table('mhs')
                ->insert([
                    'nim' => 22030003,
                    'nama' => 'Suzana',
                    'prodi' => 'D3TI',
                    'angkatan' => 2022]);
    echo "Data mhs berhasil ditambahkan";
}
```

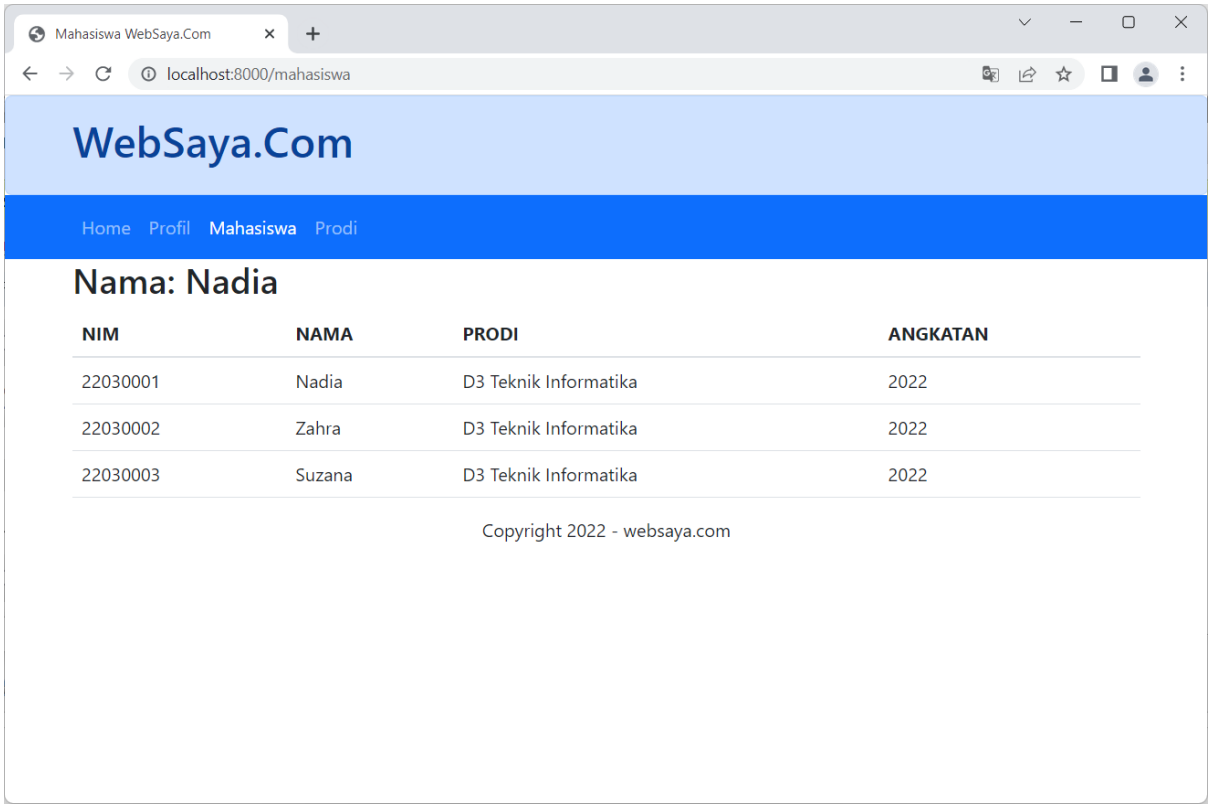
Tambahkan route pada file web.php seperti berikut:

```
Route::get('/mahasiswa/create',[mahasiswaController::class,'create']);
```

Pada script diatas adalah script untuk menambah data ke tabel yang ada di dalam database, pada kasus diatas tabel yang akan ditambah datanya adalah tabel mhs. Laravel sendiri menggunakan array assosiatif untuk parameter kolom dan nilai yang akan ditambah dengan ketentuan index dari array akan menjadi nama kolom dan nilai dari array akan menjadi nilai yang akan disimapn ke kolom. Jika kita menjalankan http://localhost:8000/mahasisw/create pada browser akan muncul tampilan seperti berikut:



Dan bila kita memeriksa hasilnya yaitu akses laman http://localhost:8000/resmhs tabel mhs yang ada di dalam database websaya maka akan ada satu data baru seperti pada gambar dibawah:



MEMPERBAHARUI DATA QUERY BUILDER

Untuk memperbaharui (update) data, kita harus menentukan terlebih dahulu data mana yang akan di update. Pada contoh ini kita akan memperbaharui data yang baru saja ditambahkan yaitu data dengan nim 22030003

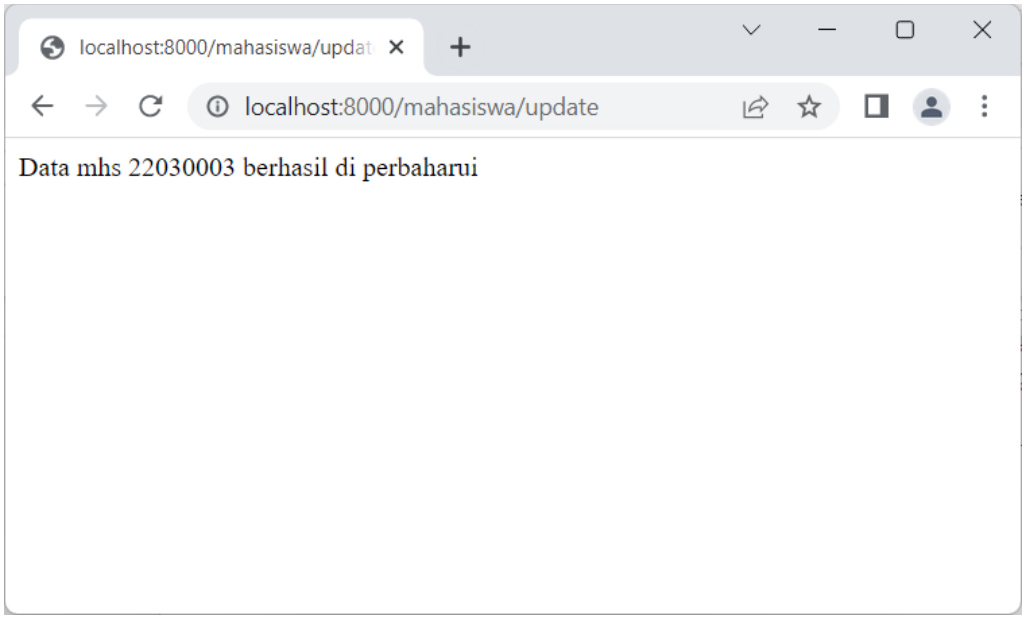
Pada route diatas jika kita membuka http://localhost:8000/resmhs/update maka kita akan dibawa ke function update() yang ada pada file resmhsController.php, maka dari itu perbaharui fungsi update() di file resmhsController.php yang ada pada folder App/Http/Controller dengan script sebagai berikut:

```php
public function update()
{
    DB::table('mhs')
                ->where('nim',22030003)
                ->update([
                    'nama' => 'Bokir',
                    'prodi' => 'D3TI',
                    'angkatan' => 2022]);
    echo "Data mhs 22030003 berhasil di perbaharui";
}
```

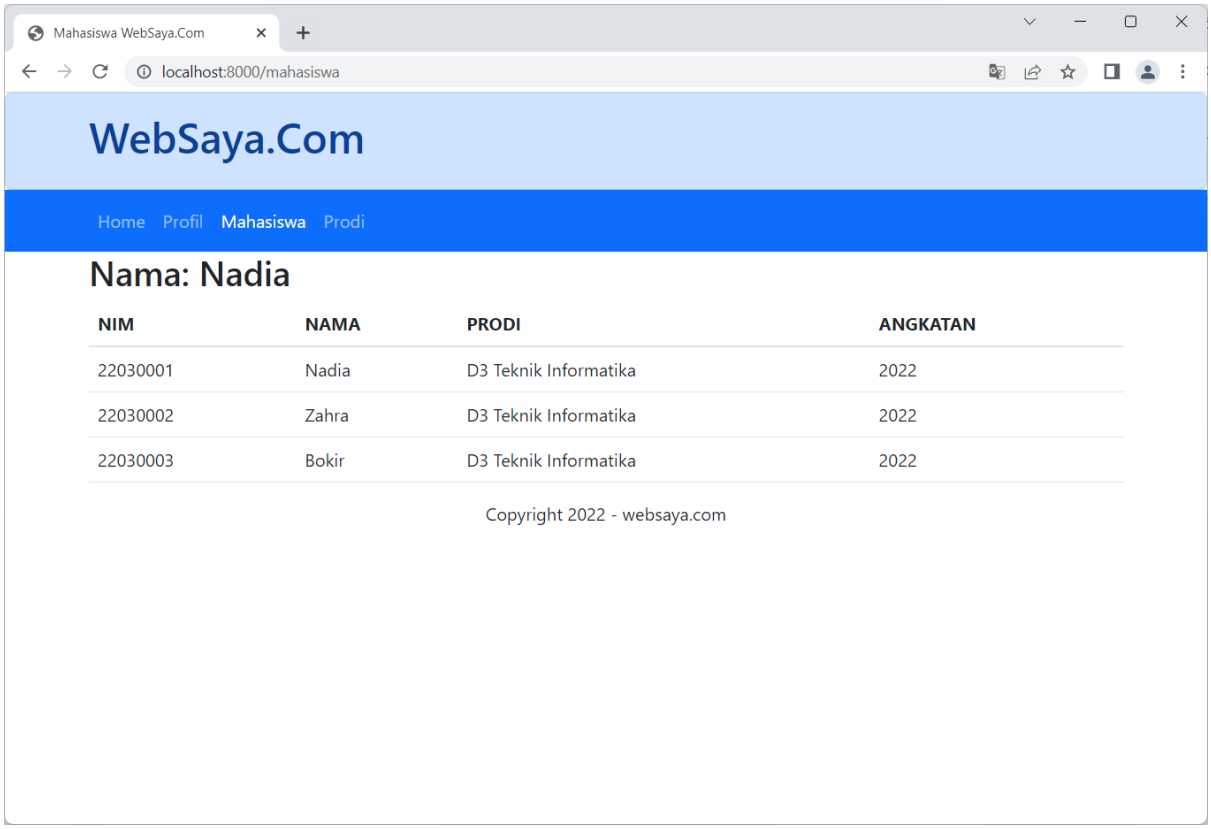
Tambahkan route pada file web.php seperti berikut:

```php
Route::get('/mahasiswa/update',[mahasiswaController::class,'update']);
```

Pada script diatas merupakan script untuk memperbaharui data menggunakan Query Builder. Pertama kita harus menentukan data mana yang akan diperbaharui dengan menggunakan fungsi where() dan diikuti dengan fungsi update() untuk memperbaharui data yang ada pada kolom. Dan jika dilihat pada tabel mhs yang ada pada database penjualan data dengan nim 22030003 akan berubah. Jika kita menjalankan http://localhost:8000/mahasisw/update pada browser akan muncul tampilan seperti berikut:



Dan untuk melihat perubahan datanya maka silakhan akses alamat http://localhost:8000/resmhs/. Perhatikan pada NIM 22030003 yang sebelumnya bernama Suzana sekarang berubah menjadi Bokir seperti gambar berikut:



MENGHAPUS DATA QUERY BUILDER

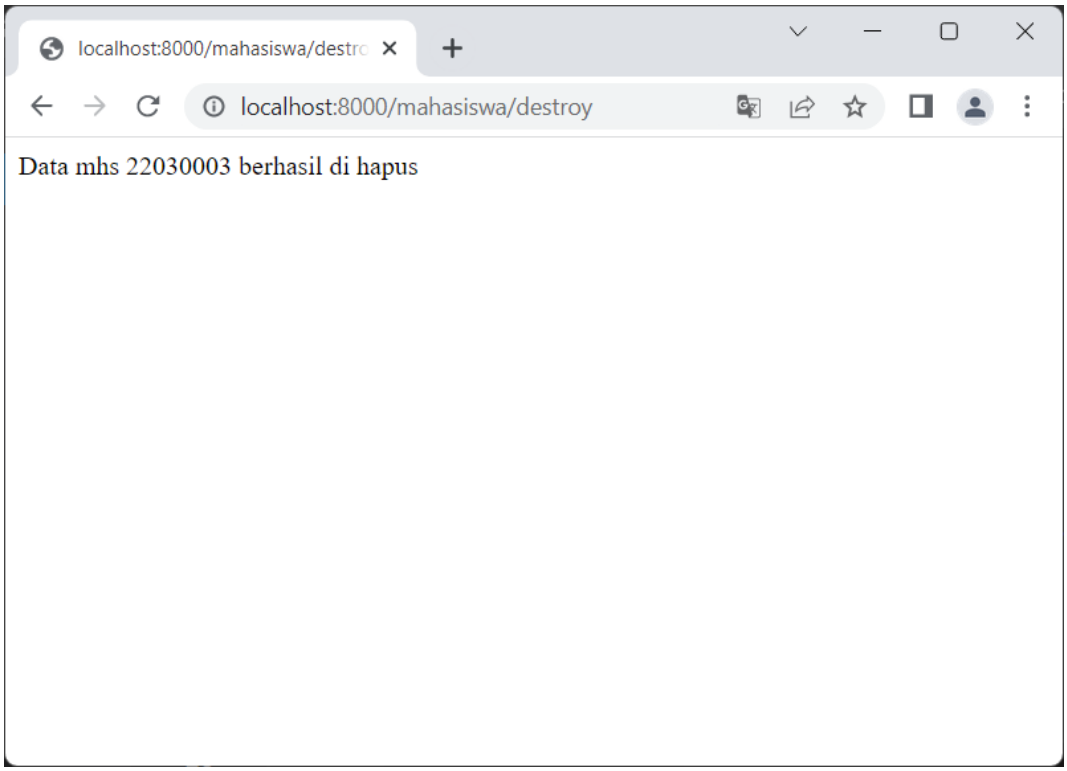
Untuk menghapus data kita perlu menentukan terlebih dahulu data mana yang akan dihapus lalu diikuti dengan fungsi **delete().** Pertama buatlah route seperti berikut pada file web.php yang ada pada folder routes:

```
Route::get('/mahasiswa/destroy',[mahasiswaController::class,'destroy']
```

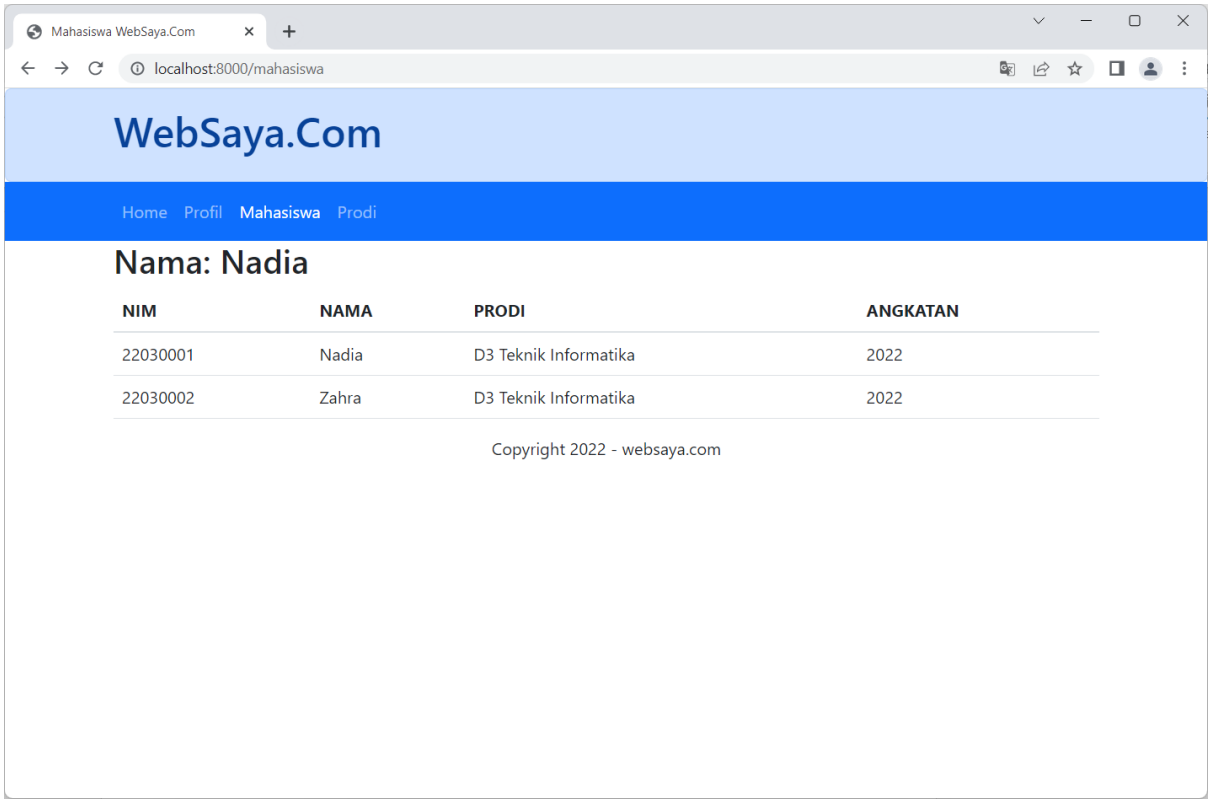
Pada route diatas jika kita mengakses **http://localhost:8000/resmhs/destroy** pada browser maka kita akan diarahkan ke fungsi delete yang ada pada file **resmhsController,** maka dari itu buatlah fungsi yang bernama delete pada file **resmhsController.php** dan isikan script sebagai berikut:

```php
public function destroy()
{
    DB::table('mhs')
                ->where('nim',22030003)
                ->delete();
    echo "Data mhs 22030003 berhasil di hapus";
}
```

Pada script diatas adalah script untuk menghapus data menggunakan Query Builder. Pertama kita harus menentukan terlebih dahulu data mana yang akan dihapus menggunakan fungsi where() untuk menghapus data dengan syarat tertentu. Lalu diikuti dengan fungsi delete() untuk menghapus data. Jika kita menjalankan **http://localhost:8000/resmhs/destroy** maka akan muncul halaman seperti berikut:



Dan jika dilihat pada laman http://localhost:8000/resmhs tabel mhs data dengan NIM 22030003 telah dihapus.



TUGAS

1. Implementasikan query builder untuk tabel **Profil** dan **Prodi**
   - Menampilkan data
   - Membuat proses Insert
   - Membuat proses Update
   - Membuat proses Hapus/Destroy
2. Layout halamannya menggunakan master template blade

Catatan:

- Langkah-langkah pembuatannya dapat mengikuti praktikum
- Laporan praktikum tugas dan sourcecode dikumpulkan ke elearning

-