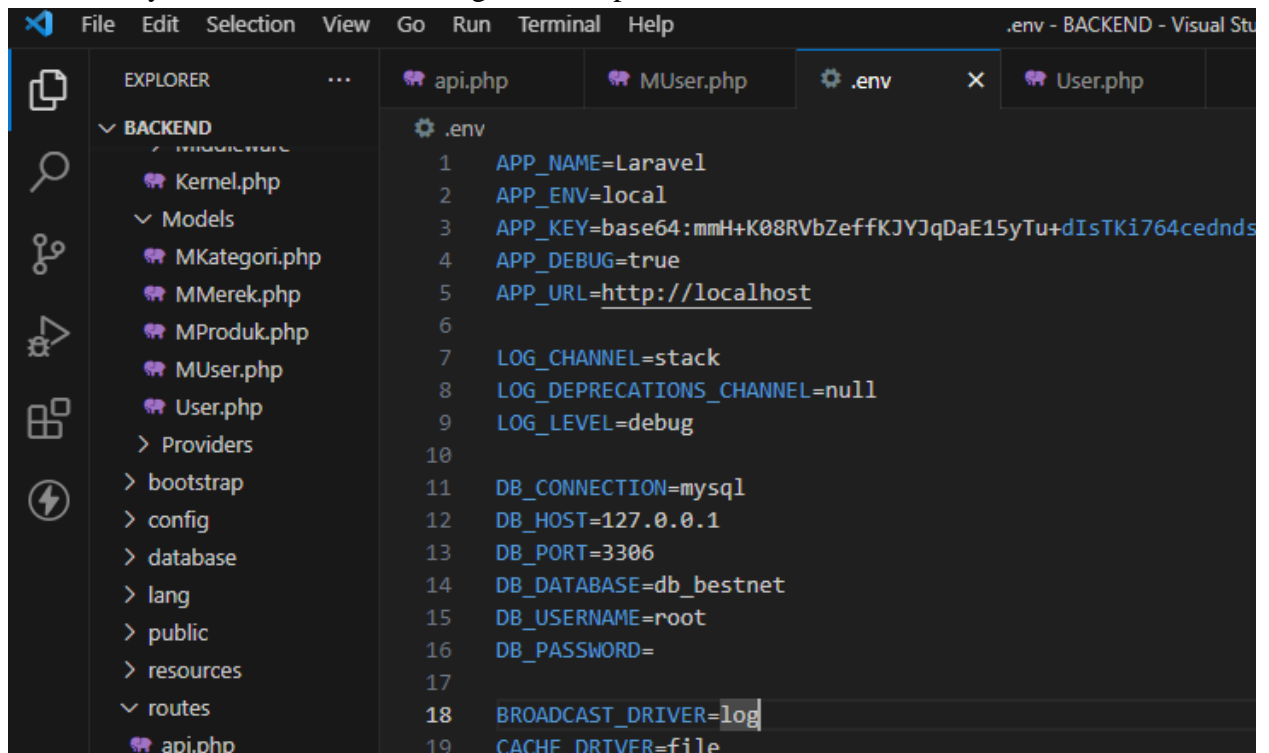


# Laporan Manual Book BackEnd PWBS-TA-Kopikap

Dalam hal ini saya sebagai backend dalam tim kopikap akan menjelaskan bagian yang akan saya buat. Pertama saya akan melakukan konfigurasi database ke dalam project. Selanjutnya saya akan membuat api yang akan nantinya di pakai oleh frontend. Api yang saya bikin disini ada empat yaitu api produk, merek, kategori dan user yang dimana keempat api ini akan digunakan oleh front end.

## 1. Pembuatan api produk

- 1) Pertama saya akan melakukan setting database pada file .env



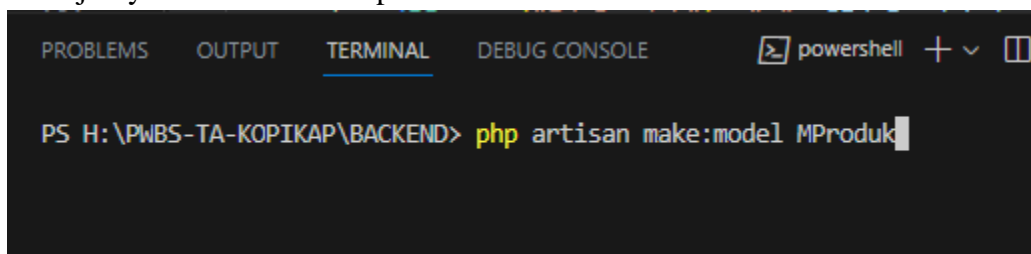
The screenshot shows the Visual Studio Code editor interface with the .env file open. The Explorer sidebar on the left shows the project structure under the 'BACKEND' folder, including files like Kernel.php, Models (MKategori.php, MMerek.php, MProduk.php, MUser.php), User.php, Providers, bootstrap, config, database, lang, public, resources, routes, and api.php. The .env file content is as follows:

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:mmH+K08RVbZeffKJYJqDaE15yTu+dIsTKi764cednds
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=db_bestnet
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
```

- 2) Selanjutnya melakukan Setting route api untuk membuat route api produk pada halaman api.php

```
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {  
    return $request->user();  
});  
  
// Route untuk tampil data produk  
Route::get('/tampil', [Produk::class, 'tampil']);  
// Route Untuk detail produk  
Route::get('/detail/{parameter}', [Produk::class, 'detail']);  
// Route Untuk hapus data produk  
Route::delete('/delete/{parameter}', [Produk::class, 'delete']);  
// Route Untuk tambah data produk  
Route::post('/insert', [Produk::class, 'insert']);  
// Route untuk update data produk  
Route::put('/updateProduk/{parameter}', [Produk::class, 'updateProduk']);
```

- 3) Selanjutnya kita buat Model produk



The screenshot shows a PowerShell terminal window with the following tabs: PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The terminal title bar indicates it is a powershell window. The command entered in the terminal is:

```
PS H:\PWBS-TA-KOPIKAP\BACKEND> php artisan make:model MProduk
```

- 4) Selanjutnya buat model produk dan buat function untuk mengambil data dari database disini saya kasih nama function nya tampilData()

```

class MProduk extends Model
{
    protected $table = 'tbl_produk';

    function tampilData()
    {
        $query = DB::table('tbl_produk')
            ->select(
                "Id_Produk",
                "Nama_Produk",
                "Harga",
                "Stok_Produk",
                "Spesifikasi",
                "Foto_Produk",
                "Kategori",
                "Model"
            )
            ->orderBy("Id_Produk")
            ->get();

        return $query;
    }
}

```

- 5) Selanjutnya buat function detailData() di model produk untuk melihat detail dari data

```

function detailData($parameter){
    $query = DB::table('tbl_produk')
        ->select(
            "Id_Produk",
            "Nama_Produk",
            "Harga",
            "Stok_Produk",
            "Spesifikasi",
            "Foto_Produk",
            "Kategori",
            "Model",
        )
        ->where(DB::raw("Id_produk"), "=", $parameter)
        ->orderBy("Id_Produk")
        ->get();

    return $query;
}

```

- 6) Selanjutnya buat function untuk menghapus data di sini saya kasih nama deleteData()

```
// buat fungsi delete data
function deleteData($parameter)
{
    DB::table("tbl_produk")
        ->where(DB::raw("Id_Produk"), '=', $parameter)
        ->delete();
}
```

- 7) Selanjutnya buat function tambah data di modelproduk untuk menambahkan data, disini saya kasih nama saveData()

```
8) // buat fungsi tambah data
9) function saveData($Id_produk, $Nama_Produk, $Harga, $Stok_Produk,
    $Spesifikasi, $Foto_Produk,$Kategori, $Model)
10) {
11)     DB::table("tbl_produk")
12)     ->insert([
13)         "Id_Produk" => $Id_produk,
14)         "Nama_Produk" => $Nama_Produk,
15)         "Harga" => $Harga,
16)         "Stok_Produk" => $Stok_Produk,
17)         "Spesifikasi" => $Spesifikasi,
18)         "Foto_Produk" => $Foto_Produk,
19)         "Kategori" => $Kategori,
20)         "Model" => $Model
21)     ]);
22) }
```

- 8) Selanjutnya kita buat function untuk cek update data apakah data tersebut ada atau tidak

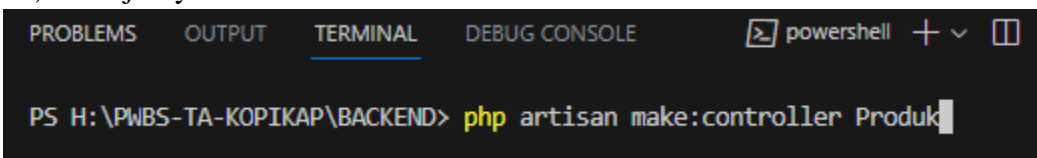
```
// Fungsi Untuk Cek Ubah Data
function checkUpdate($Id_Produk_Lama, $Id_Produk_baru)
{
    // tampilkan data
    $query = DB::table("tbl_produk")
        ->select("Id_Produk")
        ->where("Id_Produk", "=", $Id_Produk_baru)
        ->where(DB::raw("(Id_Produk)"), "!=" , $Id_Produk_Lama)
        ->get();

    return $query;
}
```

9) Setelah kita membuat function untuk cekupdate selanjutnya kita membuat function untuk update data.

```
// Update Data Kamar
function updateData(
    $Id_Produk,
    $Nama_Produk,
    $Harga,
    $Stok_Produk,
    $Spesifikasi,
    $Foto_Produk,
    $Kategori,
    $Model,
    $Id_Produk_Lama
) {
    DB::table("tbl_produk")
        ->where(DB::raw("Id_Produk"), "=", $Id_Produk_Lama)
        ->update([
            "Id_Produk" => $Id_Produk,
            "Nama_Produk" => $Nama_Produk,
            "Harga" => $Harga,
            "Stok_Produk" => $Stok_Produk,
            "Spesifikasi" => $Spesifikasi,
            "Foto_Produk" => $Foto_Produk,
            "Kategori" => $Kategori,
            "Model" => $Model
        ]);
}
```

10) Selanjutnya buat controller Produk



The screenshot shows a PowerShell terminal window with the following content:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  powershell + -
```

PS H:\PWBS-TA-KOPIKAP\BACKEND> php artisan make:controller Produk

11) Selanjutnya kita buat function untuk memanggil model Mproduk di controller kita

```
function __construct()  
{  
    $this->model = new MProduk();  
}
```

```
use Illuminate\Http\Request;  
use App\Models\MProduk;
```

12) Selanjutnya kita buat fungsi tampil untuk menampilkan data di controller kita .

```
// Function Untuk Tampil Data  
function tampil()  
{  
    // ambil fungsi dari viewData(dari Mkaryawan)  
    $data = $this->model->tampilData();  
  
    // tampilkan hasil dari "tbkaryawan"  
    return response([  
        "Produk" => $data  
    ], http_response_code());  
}
```

13) Selanjutnya kita buat fungsi detail untuk melihat detailData() di controller kita

```
function detail($parameter)  
{  
    // ambil fungsi dari viewData(dari Mkaryawan)  
    $data = $this->model->detailData($parameter);  
  
    // tampilkan hasil dari "tbkaryawan"  
    return response([  
        "Produk" => $data  
    ], http_response_code());  
}
```

14) Selanjutnya kita buat fungsi delete di controller

```

// buat fungsi untuk delete data
function delete($parameter)
{
    // cek data dari tbl_karyawan
    //(berdasarkan nik)
    $data = $this->model->detailData($parameter);

    // jika data ditemukan
    if (count($data) != 0) {
        // lakukan penghapusan data
        $data = $this->model->deleteData($parameter);
        // buat pesan dan status hasil penghapusan data
        $status = 1;
        $pesan = "Data Berhasil di Hapus";
    }
    // jika data tidak ditemukan
    else {
        // tampilkan pesan data gagal dihapus
        $status = 1;
        $pesan = "Data Gagal di Hapus ! (NIK tidak ditemuka";
    }

    // tampilkan hasil respon

    return response([
        "status" => $status,
        "pesan" => $pesan
    ], http_response_code());
}

```

Activate Windows

15) Selanjutnya buat fungsi tambah data di controller kita

```

function insert(Request $req)
{
    // ambil data hasil input
    $data = array(
        "Id_Produk" => $req->Id_Produk,
        "Nama_Produk" => $req->Nama_Produk,
        "Harga" => $req->Harga,
        "Stok_Produk" => $req->Stok_Produk,
        "Spesifikasi" => $req->Spesifikasi,
        "Foto_Produk" => $req->Foto_produk,
        "Kategori" => $req->Kategori,
        "Model" => $req->Model,
    );
    // baruu
    $parameter = ($data["Id_Produk"]);
    // cek apakah data karyawan (nik) sudah pernah tersimp
    $check = $this->model->detailData($parameter);

```

```

// jika data tidak ditemukan
if (count($check) == 0) {
    // lakukan proses penyimpanan
    $this->model->saveData($data["Id_Produk"], $data["Nama_Produk"]);
    // buat pesan dan status hasil penyimpanan data
    $status = 1;
    $pesan = "Data Berhasil disimpan";
}
// jika data tidak ditemukan
else {
    // tampilkan pesan data gagal disimpan
    $status = 0;
    $pesan = "Data Gagal disimpan";
}
// tampilkan hasil respon

return response([
    "status" => $status,
    "pesan" => $pesan
], http_response_code());

```



16) Selanjutnya buat fungsi update pada controller produk.

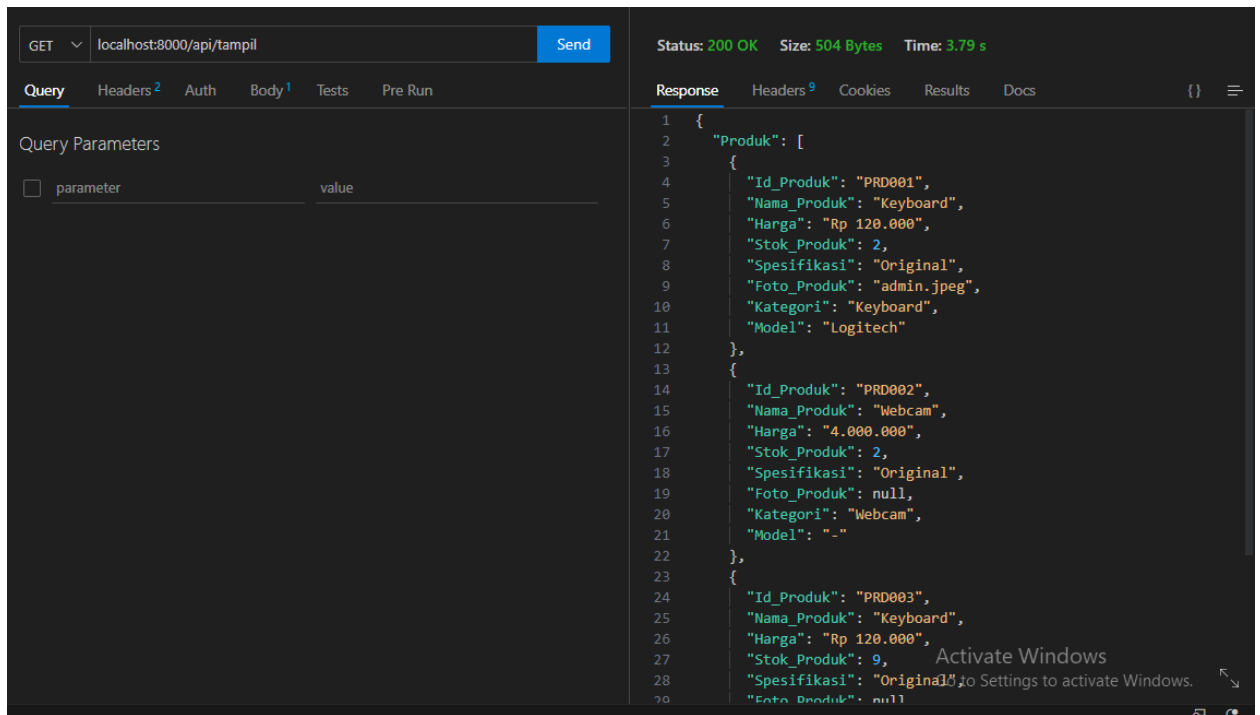
```
// Function untuk Update Data Kamar
function updateProduk(
    $parameter,
    Request $req
) {
    // Ambil data hasil input
    $data = array(
        "Id_Produk" => $req->Id_Produk,
        "Nama_Produk" => $req->Nama_Produk,
        "Harga" => $req->Harga,
        "Stok_Produk" => $req->Stok_Produk,
        "Spesifikasi" => $req->Spesifikasi,
        "Foto_Produk" => $req->Foto_Produk,
        "Kategori" => $req->Kategori,
        "Model" => $req->Model,
    );
}
```

```

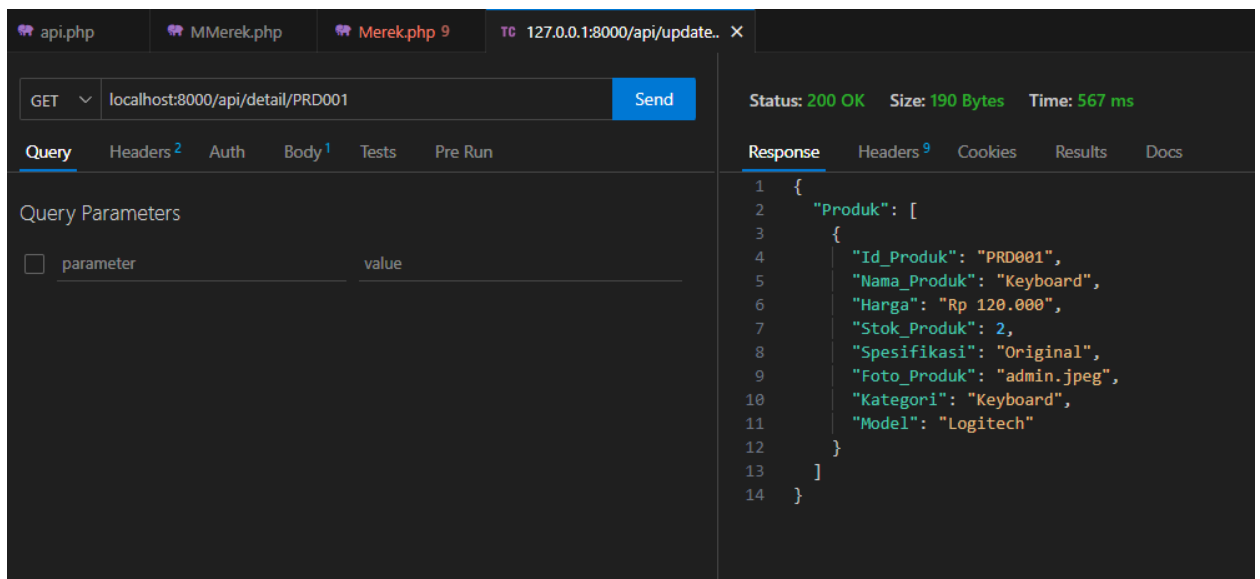
// Cek apakah data kamar tersedia/tidak
$cek = $this->model->checkUpdate($parameter, $data["Id_Produk"]);
// Jika data tidak ditemukan
if (count($cek) == 0) {
    // Ubah data kamar
    $this->model->updateData(
        $data["Id_Produk"],
        $data["Nama_Produk"],
        $data["Harga"],
        $data["Stok_Produk"],
        $data["Spesifikasi"],
        $data["Foto_Produk"],
        $data["Kategori"],
        $data["Model"],
        $parameter
    );
    // tampilkan pesan
    $status = "1";
    $pesan = "Data Berhasil di Ubah";
}
// Jika data tidak ditemukan
else {
    $status = 0;
    $pesan = "Data Gagal Diubah ! (Kode_Kamar Sudah Pernah Tersimpan)";
}
// Tampilkan pesan
return response([
    "status" => $status,
    "pesan" => $pesan
], http_response_code());
}

```

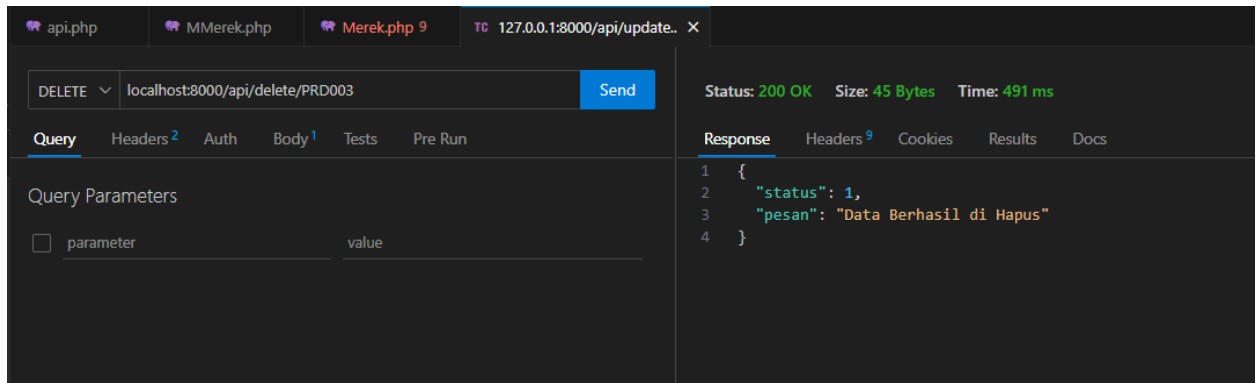
17) Selanjutnya setelah kita selesai buat model Mproduk dan Controller Produk kita testing apakah api yang kita buat sudah bisa tampil atau belum. Disini saya akan testing untuk tampil produk dengan menggunakan thunder client.



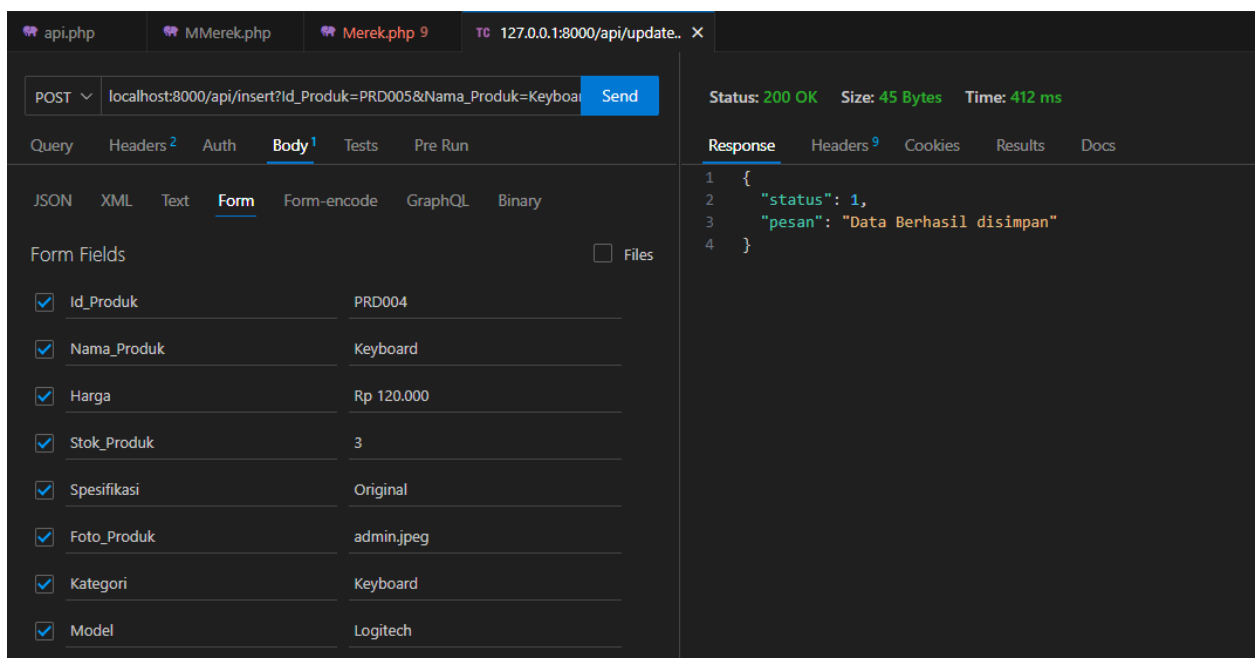
18) Testing api untuk detail produk.



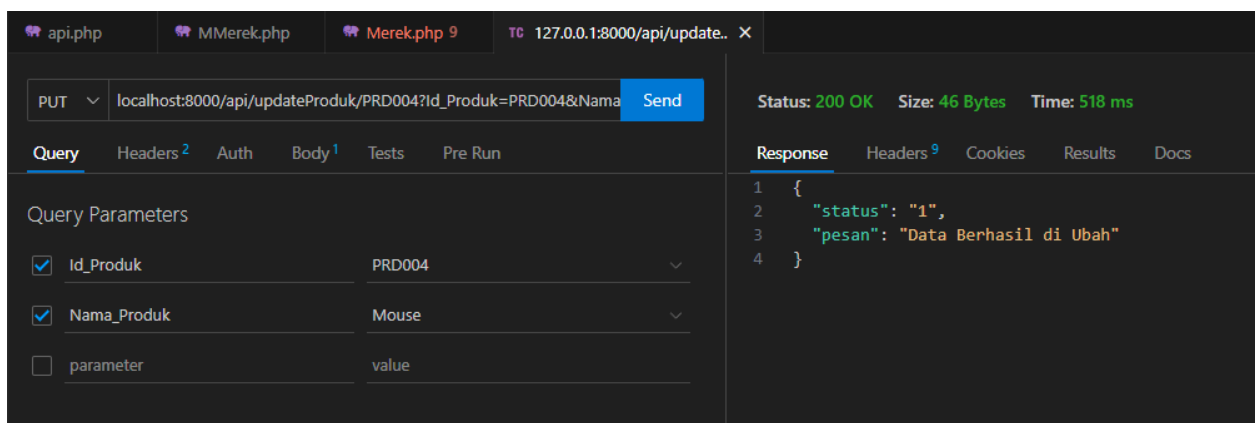
19) Testing api untuk delete produk.



20) Testing api untuk menyimpan data produk.



21) Testing api untuk update data produk

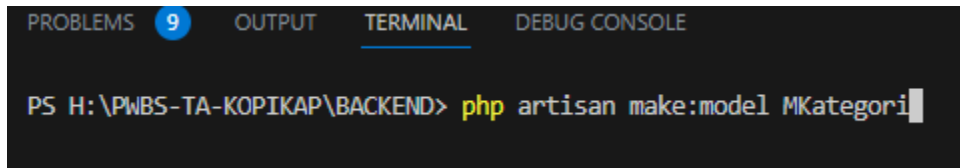


## 2. Membuat Api Kategori

- 1) Buka route/api lalu buat route CRUD untuk api Kategori

```
// Route untuk tampil data kategori
Route::get('/tampilkategori', [Kategori::class, 'tampilkategori']);
// Route untuk tampil data kategori
Route::get('/detailkategori/{parameter}', [Kategori::class, 'detailkategori']);
// Route Untuk hapus data kategori
Route::delete('/deletekategori/{parameter}', [Kategori::class, 'deletekategori']);
// Route Untuk tambah data kategori
Route::post('/insertkategori', [Kategori::class, 'insertkategori']);
// Route untuk update data kategori
Route::put('/updateKategori/{parameter}', [Kategori::class, 'updateKategori']);
```

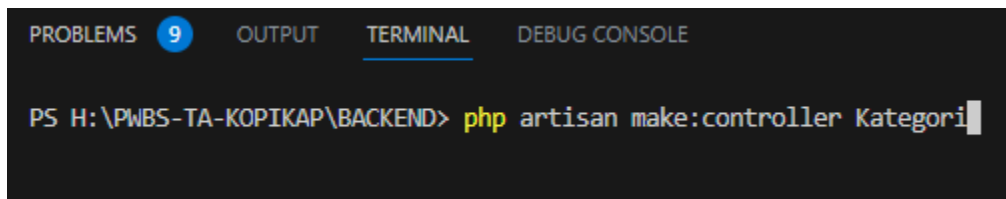
- 2) Buat model untuk api kategori dari terminal



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is active. The command entered is `php artisan make:model MKategori`.

```
PS H:\PWBS-TA-KOPIKAP\BACKEND> php artisan make:model MKategori
```

- 3) Buat controller untuk api kategori dari terminal



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is active. The command entered is `php artisan make:controller Kategori`.

```
PS H:\PWBS-TA-KOPIKAP\BACKEND> php artisan make:controller Kategori
```

- 4) Selanjutnya kita buat function `tampilDataKategori()` pada class model `Mkategori`

```

protected $table = 'tbl_kategori';

function tampilDataKategori()
{
    $query = DB::table('tbl_kategori')
        ->select(
            "Id_Kategori",
            "Nama_Kategori",
        )
        ->orderBy("Id_Kategori")
        ->get();

    return $query;
}

```

- 5) Selanjutnya buat function detailDataKategori() untuk melihat detail pada data.

```

function detailDataKategori($parameter){
    $query = DB::table('tbl_kategori')
        ->select(
            "Id_Kategori",
            "Nama_Kategori",
        )
        ->where(DB::raw("Id_Kategori"), "=", $parameter)
        ->orderBy("Id_Kategori")
        ->get();

    return $query;
}

```

- 6) Selanjutnya kita buat function untuk deleteDataKategori() untuk menghapus data kategori.

```

// buat fungsi delete data
function deleteDataKategori($parameter)
{
    DB::table("tbl_kategori")
        ->where(DB::raw("Id_Kategori"), '=', $parameter)
        ->delete();
}

```

7) Selanjutnya kita buat fungsi simpan data pada model MKategori

```
// buat fungsi tambah data
function saveDataKategori($Id_Kategori, $Nama_Kategori)
{
    DB::table("tbl_kategori")
    ->insert([
        "Id_Kategori" => $Id_Kategori,
        "Nama_Kategori" => $Nama_Kategori,
    ]);
}
```

8) Selanjutnya buat fungsi untuk cek update pada model Kategori

```
// Fungsi Untuk Cek Ubah Data
function CekUpdateKategori($Id_Kategori_Lama, $Id_Kategori_baru)
{
    // tampilkan data
    $query = DB::table("tbl_kategori")
        ->select("Id_Kategori")
        ->where("Id_Kategori", "=", $Id_Kategori_baru)
        ->where(DB::raw("(Id_Kategori)"), "!=" , $Id_Kategori_Lama)
        ->get();

    return $query;
}
```

9) Selanjutnya kita buat function untuk update data pada model MKategori

```
// Update Data Kamar
function updateDataKategori(
    $Id_Kategori,
    $Nama_Kategori,
    $Id_Kategori_Lama
) {
    DB::table("tbl_kategori")
        ->where(DB::raw("Id_Kategori"), "=", $Id_Kategori_Lama)
        ->update([
            "Id_Kategori" => $Id_Kategori,
            "Nama_Kategori" => $Nama_Kategori,
        ]);
}
```

- 10) Setelah selesai kita buat fungsi CRUD di model MKategori maka kita buat function CRUD pada controller Kategori

```
function __construct()
{
    $this->model = new MKategori();
}

// Function Untuk Tampil Data
function tampilkategori()
{
    // ambil fungsi dari viewData(dari Mkaryawan)
    $data = $this->model->tampilDataKategori();

    // tampilkan hasil dari "tbkaryawan"
    return response([
        "Kategori" => $data
    ], http_response_code());
}
```

- 11) Buat fungsi untuk detailkategori pada Controller Kategori.



```

function detailKategori($parameter)
{
    // ambil fungsi dari viewData(dari Mkaryawan)
    $data = $this->model->detailDataKategori($parameter);

    // tampilkan hasil dari "tbkaryawan"
    return response([
        "Kategori" => $data
    ], http_response_code());
}

```

12) Buat Fungsi untuk menghapus data pada Controller Kategori.

```

// buat fungsi untuk delete data
function deleteKategori($parameter)
{
    // cek data dari tbl_karyawan
    //(berdasarkan nik)
    $data = $this->model->detailDataKategori($parameter);

    // jika data ditemukan
    if (count($data) != 0) {
        // lakukan penghapusan data
        $data = $this->model->deleteDataKategori($parameter);
        // buat pesan dan status hasil penghapusan data
        $status = 1;
        $pesan = "Data Berhasil di Hapus";
    }
    // jika data tidak ditemukan
    else {
        // tampilkan pesan data gagal dihapus
        $status = 1;
        $pesan = "Data Gagal di Hapus ! (NIK tidak ditemukan !)";
    }

    // tampilkan hasil respon

    return response([
        "status" => $status,
        "pesan" => $pesan
    ], http_response_code());
}

```

13) Buat Fungsi tambah data pada Controller Kategori

```
// buat function untuk insert data
function insertkategori(Request $req)
{
    // ambil data hasil input
    $data = array(
        "Id_Kategori" => $req->Id_Kategori,
        "Nama_Kategori" => $req->Nama_Kategori,
    );
    // baruu
    $parameter =($data["Id_Kategori"]);
    // cek apakah data karyawan (nik) sudah pernah tersimpan/belum
    $check = $this->model->detailDataKategori($parameter);

    // jika data tidak ditemukan
    if (count($check) == 0) {
        // lakukan proses penyimpanan
        $this->model->saveDataKategori($data["Id_Kategori"], $data["Nama_Kategori"]);
        // buat pesan dan status hasil penyimpanan data
        $status = 1;
        $pesan = "Data Berhasil disimpan";
    }
    // jika data tidak ditemukan
    else {

        // tampilkan pesan data gagal disimpan
        $status = 0;
        $pesan = "Data Gagal disimpan";
    }
    // tampilkan hasil respon

    return response([
        "status" => $status
    ], $status);
}
```

14) Buat Fungsi untuk update data pada Controller Kategori

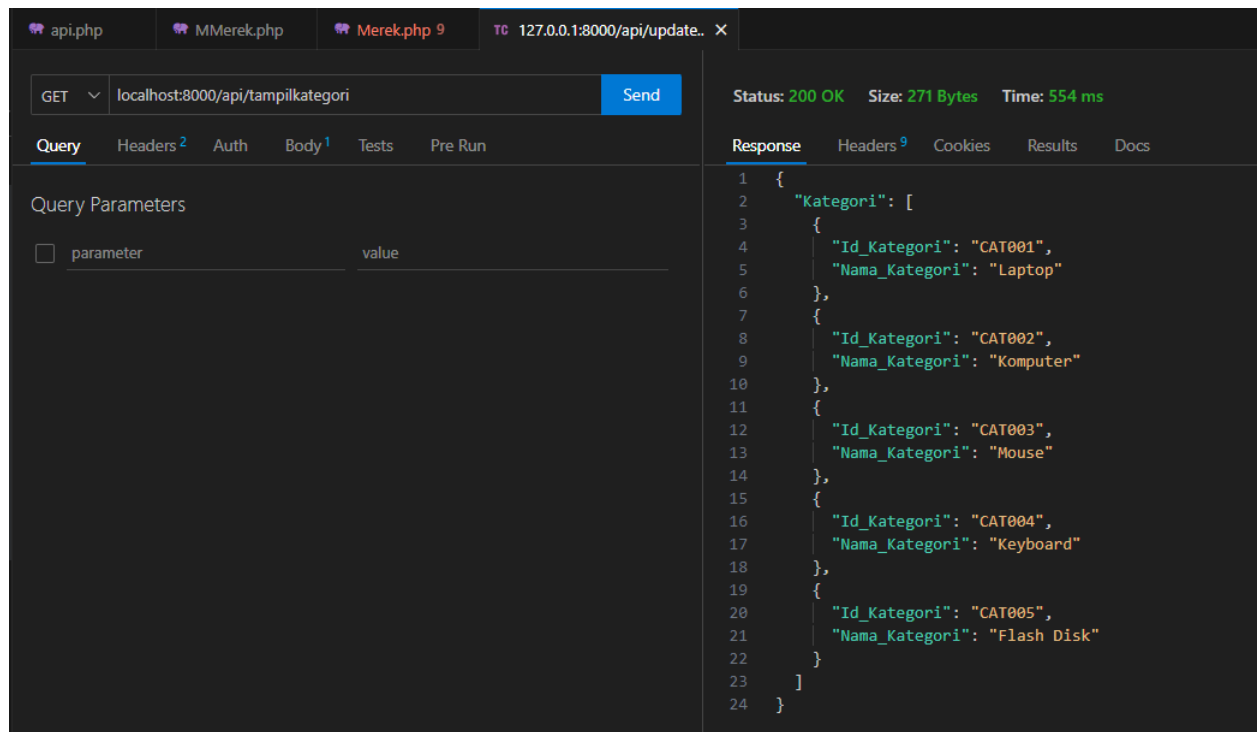
```

// Function untuk Update Data Kamar
function updateKategori(
    $parameter,
    Request $req
) {
    // Ambil data hasil input
    $data = array(
        "Id_Kategori" => $req->Id_Kategori,
        "Nama_Kategori" => $req->Nama_Kategori,
    );

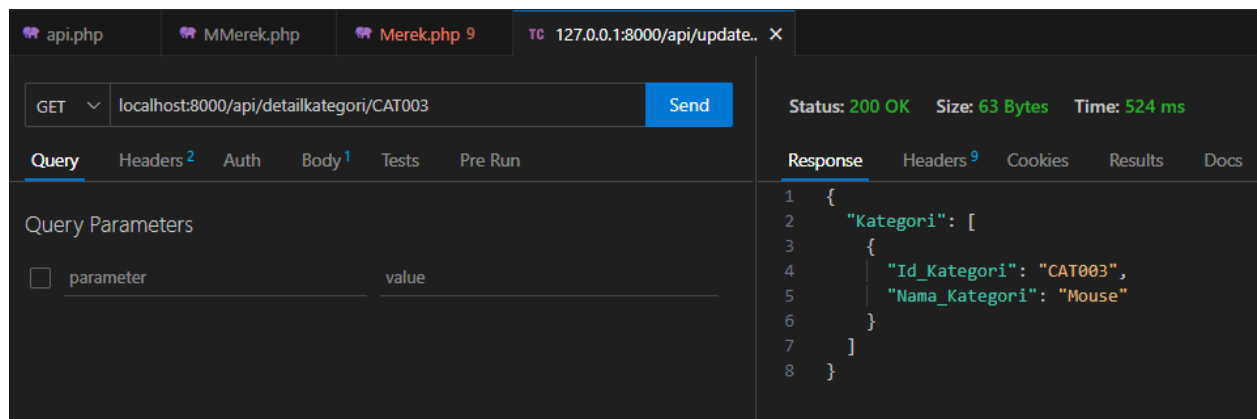
    // Cek apakah data kamar tersedia/tidak
    $cek = $this->model->CekUpdateKategori($parameter, $data["Id_Kategori"]);
    // Jika data tidak ditemukan
    if (count($cek) == 0) {
        // Ubah data kamar
        $this->model->updateDataKategori(
            $data["Id_Kategori"],
            $data["Nama_Kategori"],
            $parameter
        );
        // tampilkan pesan
        $status = "1";
        $pesan = "Data Berhasil di Ubah";
    }
    // Jika data tidak ditemukan
    else {
        $status = 0;
        $pesan = "Data Gagal Diubah ! (Kode_Kamar Sudah Pernah Tersimpan)";
    }
    // Tampilkan pesan
    return response([

```

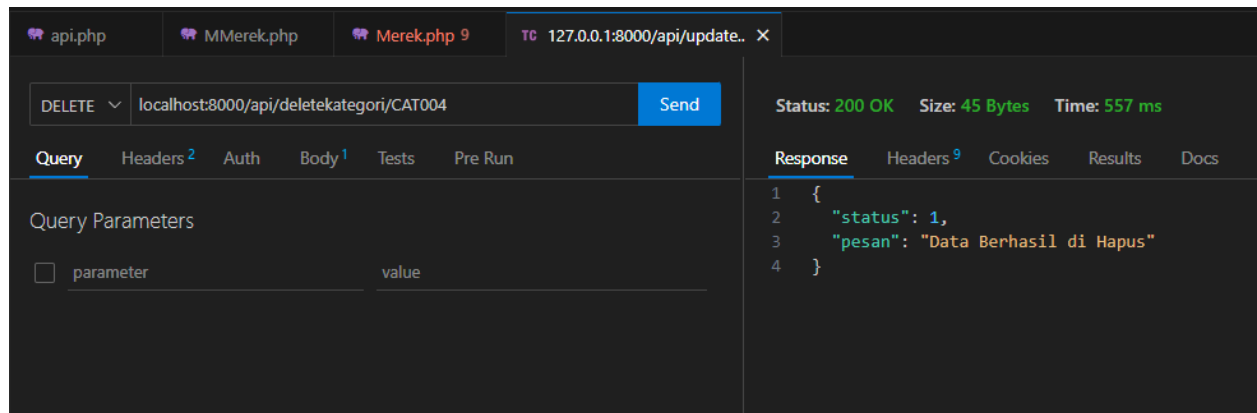
- 15) Setelah selesai kita membuat fungsi CRUD di model MKategori dan di Controller Kategori selanjutnya kita akan melakukan testing apakah api kategori yang sudah kita buat bisa tampil atau belum. Disini saya akan melakukan testing untuk tampil data kategori.



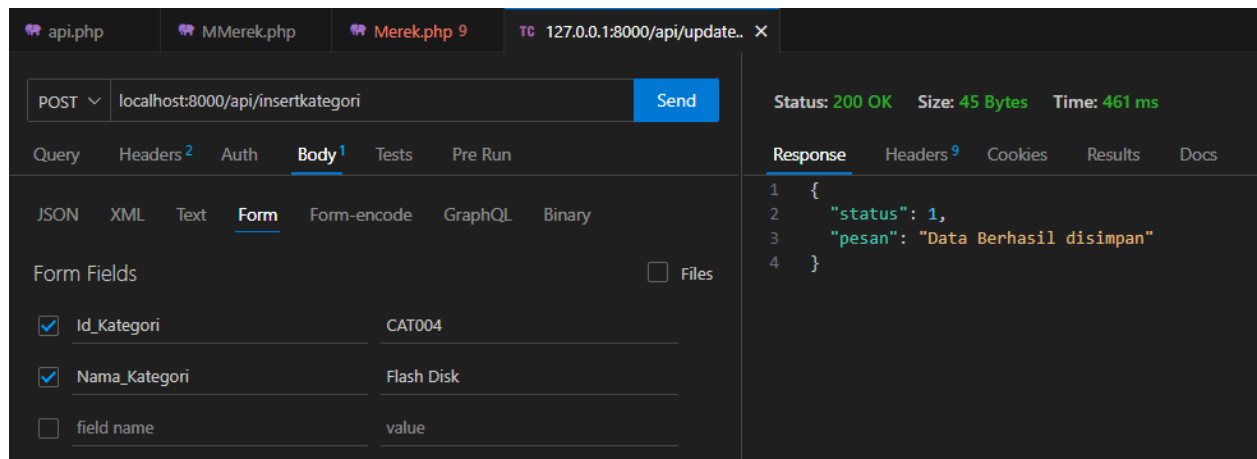
16) Testing api untuk detail pada api kategori.



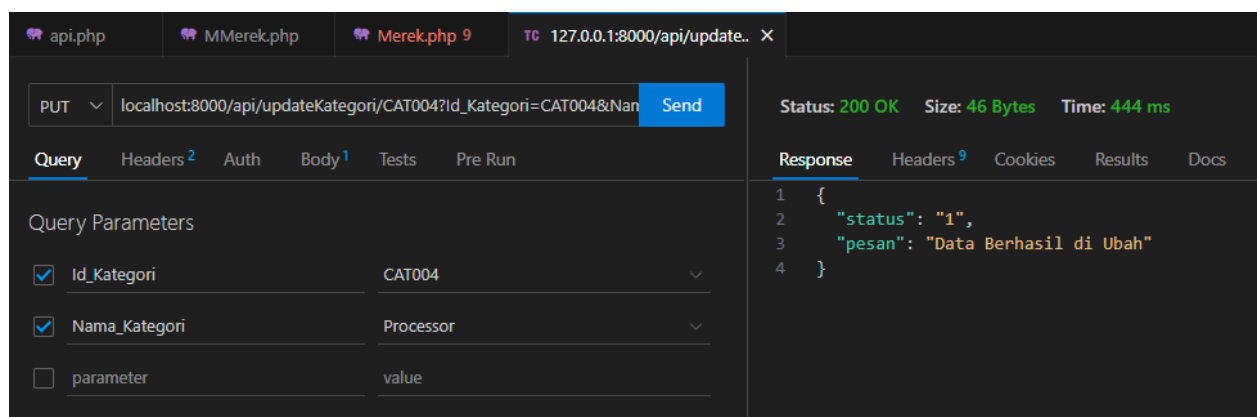
17) Testing api untuk delete pada api kategori.



18) Testing api insert kategori untuk menyimpan data pada api kategori.



19) Testing api untuk update data pada api kategori.

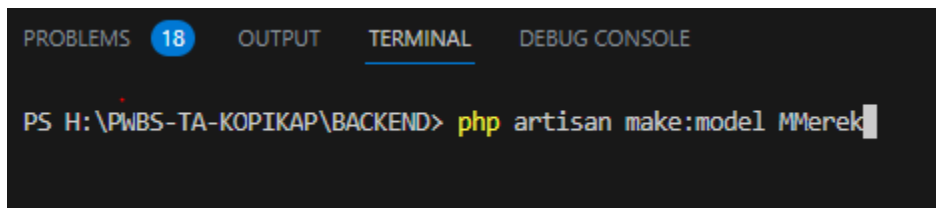


### 3. Buat Api Merek

- 1) Buka Route/api.php selanjutnya kita buat route untuk CRUD Merek pada setting route/api.php

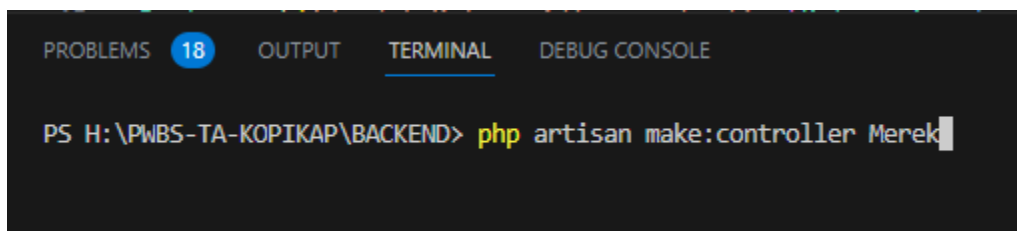
```
// Route untuk tampil data merek
Route::get('/tampilmerk', [Merek::class, 'tampilmerk']);
// Route untuk tampil data merek
Route::get('/detailmerk/{parameter}', [Merek::class, 'detailmerk']);
// Route Untuk hapus data merek
Route::delete('/deletemerek/{parameter}', [Merek::class, 'deletemerek']);
// Route Untuk tambah data merek
Route::post('/insertmerk', [Merek::class, 'insertmerk']);
// Route untuk update data kategori
Route::put('/updateMerek/{parameter}', [Merek::class, 'updateMerek']);
```

- 2) Selanjutnya kita buat model untuk api Merek.



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS' (with a blue circle containing the number 18), 'OUTPUT', 'TERMINAL' (which is selected and underlined), and 'DEBUG CONSOLE'. Below the tabs, the command prompt shows the path 'PS H:\PWBS-TA-KOPIKAP\BACKEND>' followed by the command 'php artisan make:model MMerek' with a cursor at the end.

- 3) Selanjutnya kita buat Controller untuk api Merek



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS' (with a blue circle containing the number 18), 'OUTPUT', 'TERMINAL' (which is selected and underlined), and 'DEBUG CONSOLE'. Below the tabs, the command prompt shows the path 'PS H:\PWBS-TA-KOPIKAP\BACKEND>' followed by the command 'php artisan make:controller Merek' with a cursor at the end.

- 4) Selanjutnya kita buat fungsi tampil datamerek pada model MMerek

```

protected $table = 'tbl_merek';

function tampilDataMerek()
{
$query = DB::table('tbl_merek')
    ->select(
        "Id_Merek",
        "Nama_Merek",
    )
    ->orderBy("Id_Merek")
    ->get();

    return $query;
}

```

- 5) Buat Fungsi detailDataMerek() pada MMerek.

```

function detailDataMerek($parameter){
    $query = DB::table('tbl_merek')
    ->select(
        "Id_Merek",
        "Nama_Merek",
    )
    ->where(DB::raw("Id_Merek"), "=", $parameter)
    ->orderBy("Id_Merek")
    ->get();

    return $query;
}

```

- 6) Buat fungsi deleteDataMerek untuk menghapus data merek.

```

// buat fungsi delete data
function deleteDataMerek($parameter)
{
    DB::table("tbl_merek")
        ->where(DB::raw("Id_Merek"), '=', $parameter)
        ->delete();
}

```

7) Buat fungsi InsertDataMerek() untuk menyimpan data.

```
// buat fungsi tambah data
function saveDataMerek($Id_Merek, $Nama_Merek)
{
    DB::table("tbl_merek")
    ->insert([
        "Id_Merek" => $Id_Merek,
        "Nama_Merek" => $Nama_Merek,
    ]);
}
```

8) Buat fungsi cek update pada model MMerek.

```
// Fungsi Untuk Cek Ubah Data
function CekUpdateMerek($Id_Merek_Lama, $Id_Merek_baru)
{
    // tampilkan data
    $query = DB::table("tbl_merek")
    ->select("Id_Merek")
    ->where("Id_Merek", "=", $Id_Merek_baru)
    ->where(DB::raw("(Id_Merek)"), "!=" , $Id_Merek_Lama)
    ->get();

    return $query;
}
```

9) Buat fungsi updatedata pada model MMerek



```
// Update Data Kamar
function updateDataMerek(
    $Id_Merek,
    $Nama_Merek,
    $Id_Merek_Lama
) {
    DB::table("tbl_merek")
        ->where(DB::raw("Id_Merek"), "=", $Id_Merek_Lama)
        ->update([
            "Id_Merek" => $Id_Merek,
            "Nama_Merek" => $Nama_Merek,
        ]);
}
```

- 10) Setelah selesai buat function CRUD di model MMerek selanjutnya kita akan membuat function untuk CRUD di controller Merek. Yang pertama kita buat fungsi untuk tampilmerek() pada controller Merek.

```
function __construct()
{
    $this->model = new MMerek();
}

// Function Untuk Tampil Data
function tampilmerek()
{
    // ambil fungsi dari viewData(dari Mkaryawan)
    $data = $this->model->tampilDataMerek();

    // tampilkan hasil dari "tbkaryawan"
    return response([
        "Merek" => $data
    ], http_response_code());
}
```

- 11) Selanjutnya buat function detailmerek() pada controller MMerek.

```

function detailmerek($parameter)
{
    // ambil fungsi dari viewData(dari Mkaryawan)
    $data = $this->model->detailDataMerek($parameter);

    // tampilkan hasil dari "tbkaryawan"
    return response([
        "Merek" => $data
    ], http_response_code());
}

```

12) Selanjutnya buat fungsi deletemerek() pada controller Merek.

```

// buat fungsi untuk delete data
function deletemerek($parameter)
{
    // cek data dari tbl_karyawan
    //(berdasarkan nik)
    $data = $this->model->detailDataMerek($parameter);

    // jika data ditemukan
    if (count($data) != 0) {
        // lakukan penghapusan data
        $data = $this->model->deleteDataMerek($parameter);
        // buat pesan dan status hasil penghapusan data
        $status = 1;
        $pesan = "Data Berhasil di Hapus";
    }
    // jika data tidak ditemukan
    else {
        // tampilkan pesan data gagal dihapus
        $status = 1;
        $pesan = "Data Gagal di Hapus ! (NIK tidak ditemukan !)";
    }

    // tampilkan hasil respon

    return response([
        "status" => $status,
        "pesan" => $pesan
    ], http_response_code());
}

```

13) Selanjutnya kita buat fungsi updateMerek() pada Controller Merek.

```
// buat function untuk insert data
function insertmerek(Request $req)
{
    // ambil data hasil input
    $data = array(
        "Id_Merek" => $req->Id_Merek,
        "Nama_Merek" => $req->Nama_Merek,
    );
    // baruu
    $parameter =($data["Id_Merek"]);
    // cek apakah data karyawan (nik) sudah pernah tersimpan/belum
    $check = $this->model->detailDataMerek($parameter);

    // jika data tidak ditemukan
    if (count($check) == 0) {
        // lakukan proses penyimpanan
        $this->model->saveDataMerek($data["Id_Merek"], $data["Nama_Merek"])
        // buat pesan dan status hasil penyimpanan data
        $status = 1;
        $pesan = "Data Berhasil disimpan";
    }
    // jika data tidak ditemukan
    else {

        // tampilkan pesan data gagal disimpan
        $status = 0;
        $pesan = "Data Gagal disimpan";
    }
    // tampilkan hasil respon
}
```

14) Selanjutnya kita buat fungsi updateMerek() pada Controller Merek.

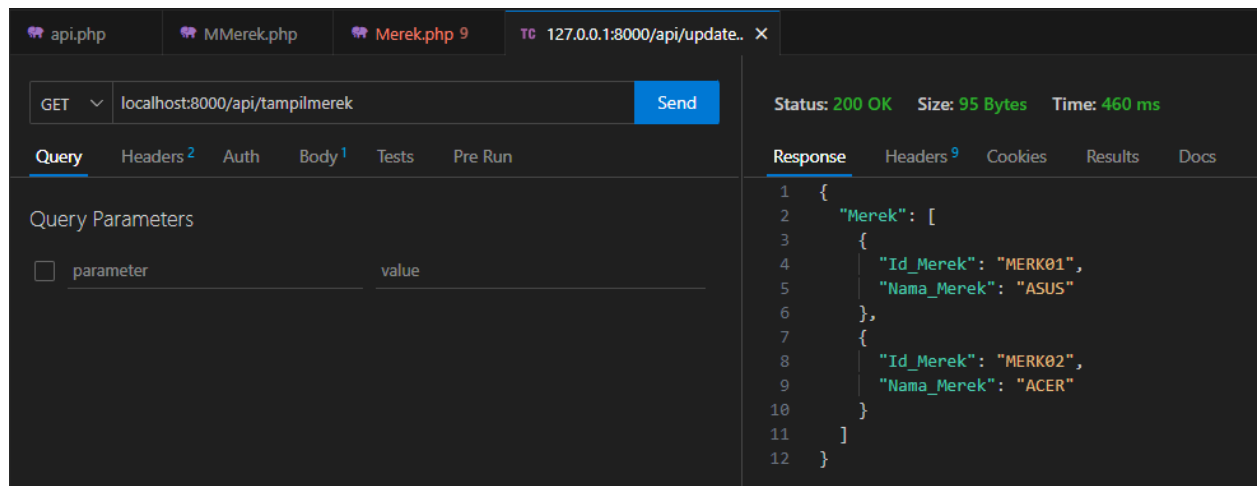
```

function updateMerek($parameter, Request $req)
{
    // Ambil data hasil input
    $data = array(
        "Id_Merek" => $req->Id_Merek,
        "Nama_Merek" => $req->Nama_Merek,
    );

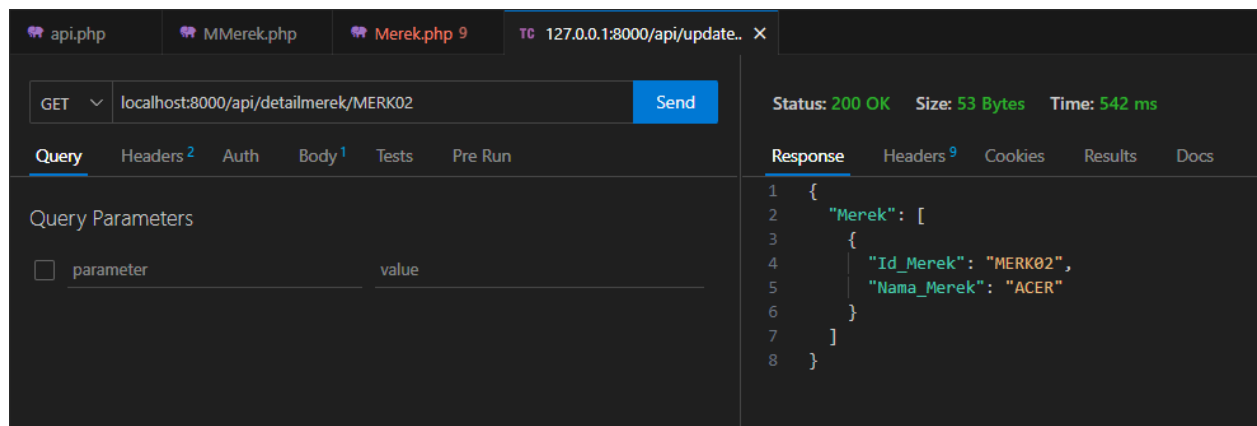
    // Cek apakah data kamar tersedia/tidak
    $cek = $this->model->CekUpdateMerek($parameter, $data["Id_Merek"]);
    // Jika data tidak ditemukan
    if (count($cek) == 0) {
        // Ubah data kamar
        $this->model->updateDataMerek(
            $data["Id_Merek"],
            $data["Nama_Merek"],
            $parameter
        );
        // tampilkan pesan
        $status = "1";
        $pesan = "Data Berhasil di Ubah";
    }
    // Jika data tidak ditemukan
    else {
        $status = 0;
        $pesan = "Data Gagal Diubah ! (Kode_Kamar Sudah Pernah Tersimpan)";
    }
    // Tampilkan pesan
    return response([
        "status" => $status,
        "pesan" => $pesan
    ], http_response_code());
}

```

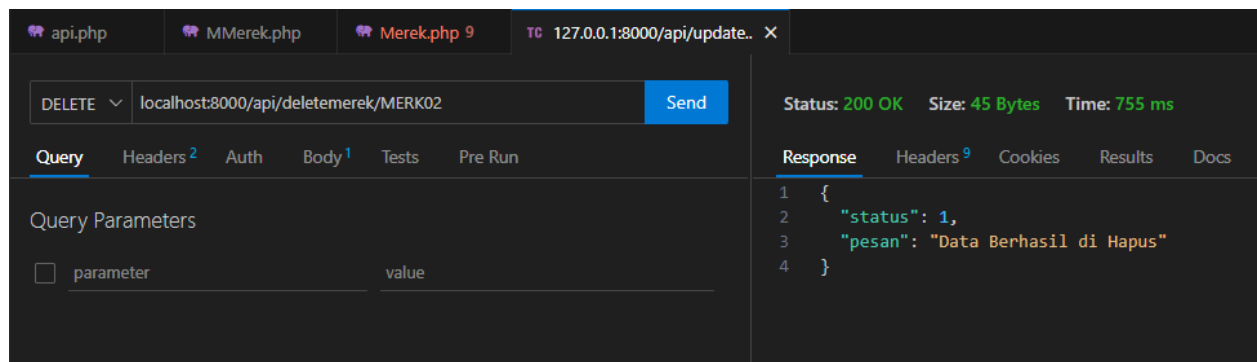
- 15) Setelah selesai kita membuat function CRUD di model MMerek dan di Controller Merek selanjutnya kita akan melakukan testing apakah api merek yang kita sudah buat sudah bisa atau belum. Disini saya akan coba testing untuk menampilkan data pada api merek.



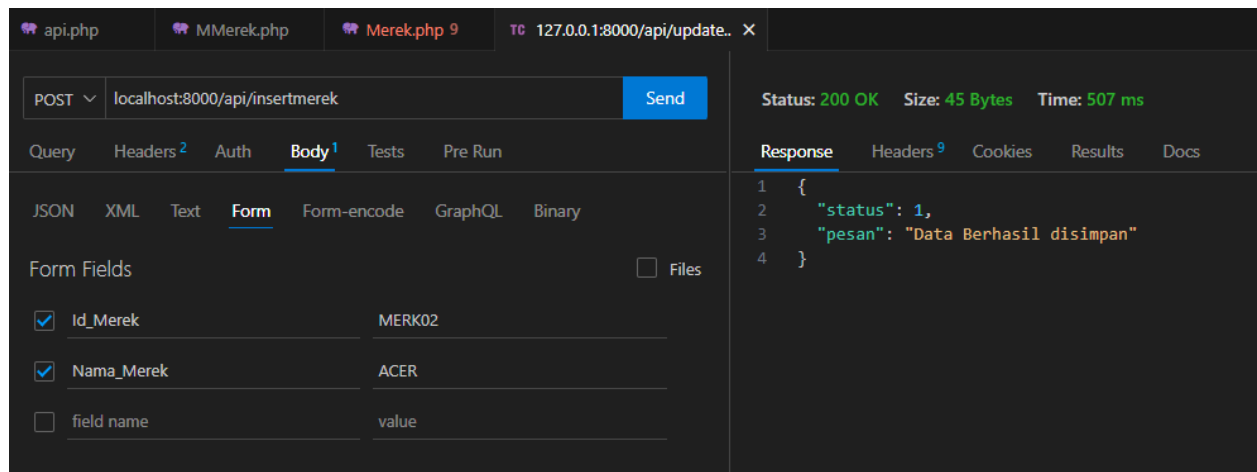
16) Testing api untuk melihat detail data pada api merek.



17) Testing api untuk menghapus data pada api merek.



18) Testing api untuk menyimpan data pada api merek.



19) Testing api untuk mengubah data pada api merek,

