Operasi CRUD dengan API Codeigniter 4

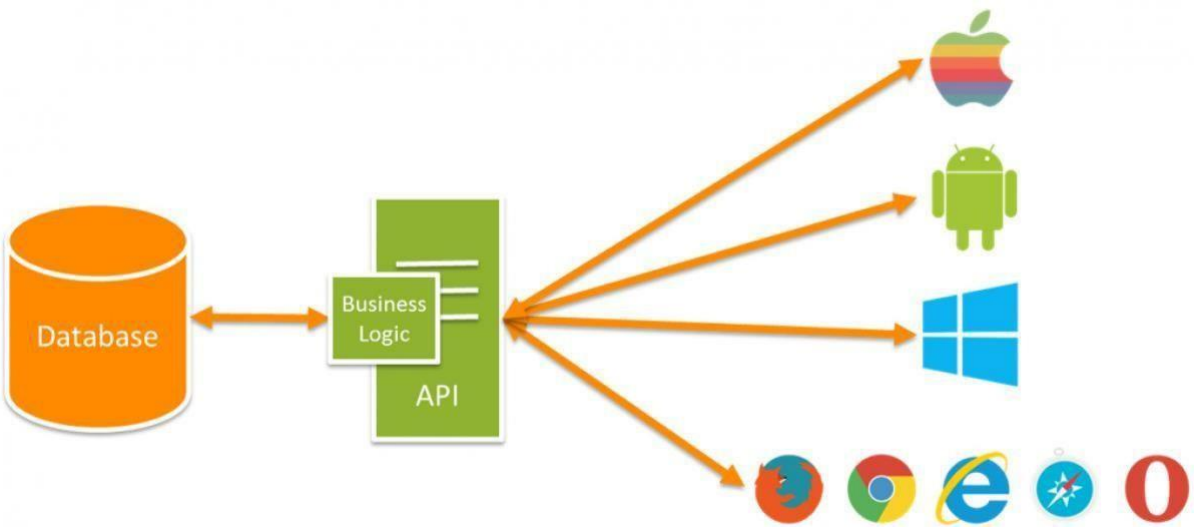
Module Mobile Programming Flutter

Membuat projek flutter yang terhubung dengan API

Apa itu API

API atau Application Programming Interface adalah sebuah interface yang dapat menghubungkan aplikasi satu dengan aplikasi lainnya. Jadi, API berperan sebagai perantara antar berbagai aplikasi berbeda, baik dalam satu platform yang sama atau lintas platform.

Perumpamaan yang bisa digunakan untuk menjelaskan API adalah seorang pelayan di restoran. Tugas pelayan tersebut adalah menghubungkan tamu restoran dengan juru masak. Tamu cukup memesan makanan sesuai daftar menu yang ada dan pelayan memberitahukannya ke juru masak. Nantinya, pelayan akan kembali ke tamu tadi dengan masakan yang sudah siap sesuai pesanan.
Itulah gambaran tugas dari API dalam pengembangan aplikasi.



Sumber : codepolitan.com

Arsitektur API

Ada tiga arsitektur API yang sering digunakan oleh developer dalam pembangunan aplikasi. Arsitektur ini berkaitan pada bentuk data yang dikirim. Adapun Arsitektur API yang sering digunakan adalah

1. RPC

RPC merupakan teknologi untuk membuat komunikasi antara client side dan server side bisa dilakukan dengan konsep sederhana.
RPC memiliki dua jenis, yaitu XML-RPC dan JSON-RPC. Sesuai namanya, XML-RPC menggunakan format XML sebagai media perpindahan data, sedangkan JSON-RPC menggunakan JSON untuk perpindahan data.

2. SOAP

Arsitektur API lainnya adalah SOAP (Simple Object Access Protocol). Arsitektur ini menggunakan XML (Extensible Markup Language) yang memungkinkan semua data disimpan dalam dokumen.

3. REST

REST atau Representational State Transfer adalah arsitektur API yang cukup populer karena kemudahan penggunaannya. Tak perlu coding yang panjang untuk menggunakannya.
REST menggunakan JSON sebagai bentuk datanya sehingga lebih ringan. Performa aplikasi pun menjadi lebih baik.

Membuat projek Toko API (Restful API)

Installasi Apache, MySql dan PHP (XAMPP)

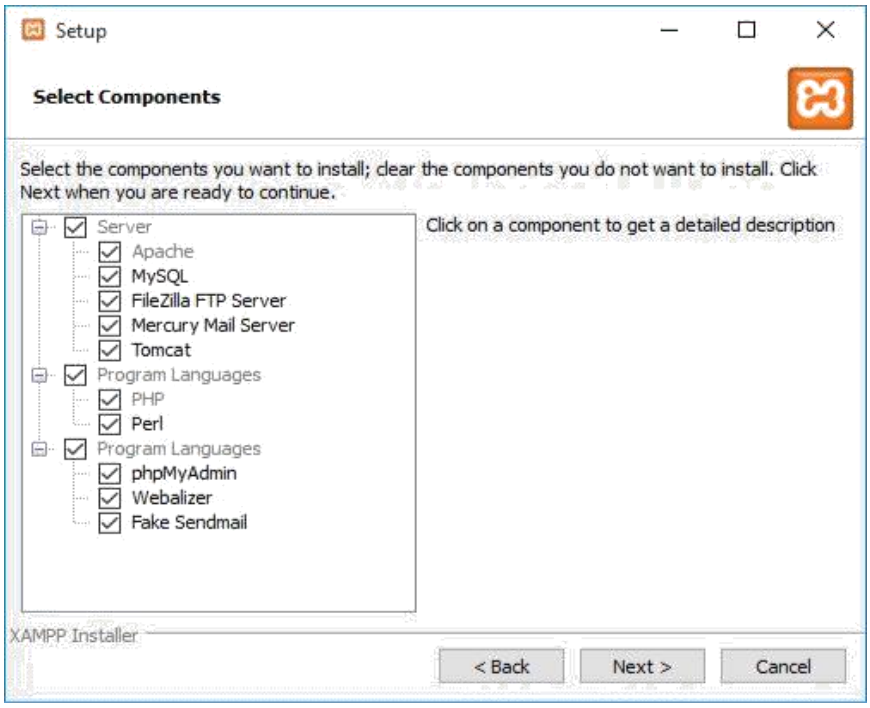
XAMPP adalah perangkat lunak untuk membuat komputer menjadi web server. XAMPP dapat di download melalui link url:

).

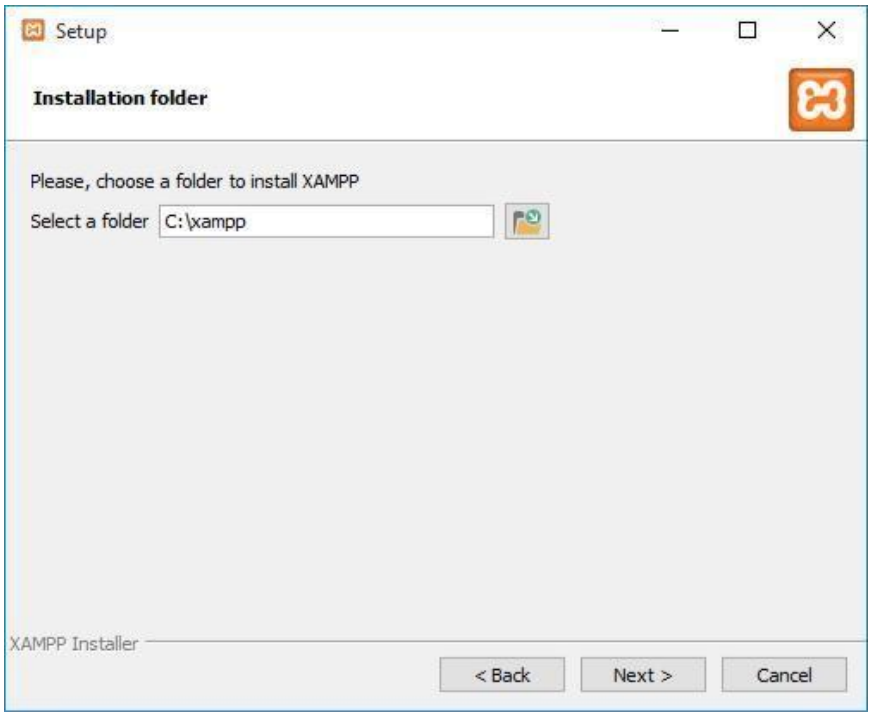
Setelah itu, double klik file xampp yang baru di download. Jika muncul pesan error seperti gambar dibawah, abaikan saja dan lanjutkan klik tombol OK.

Berikutnya akan muncul jendela Setup-XAMPP. Klik tombol Next untuk melanjutkan proses berikutnya.

Selanjutnya akan muncul jendela Select Components, yang meminta untuk memilih aplikasi yang akan diinstall. Centang saja semua kemudian klik tombol Next.



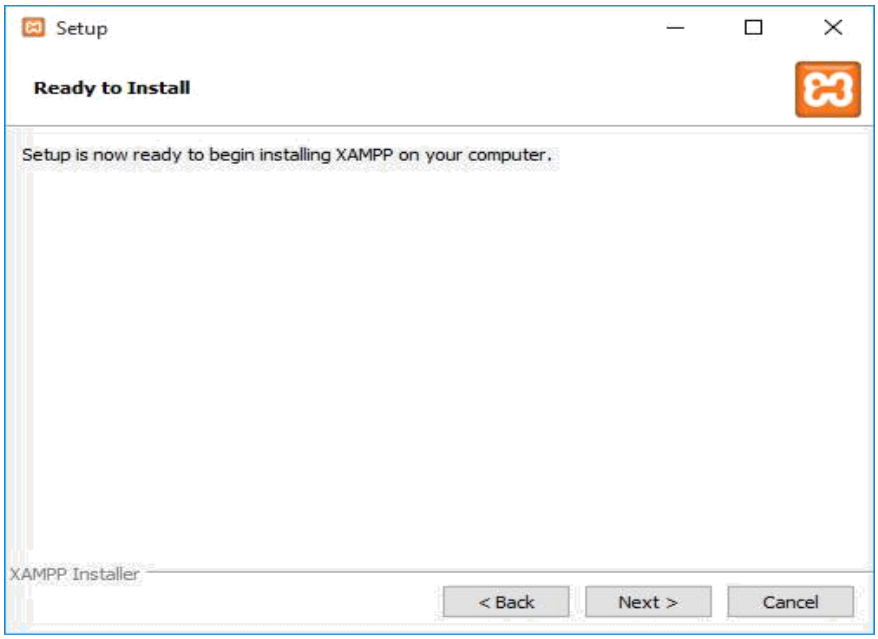
Kemudian akan muncul jendela Installation Folder, dimana anda diminta untuk menentukan lokasi penyimpanan folder xampp, secara bawaan akan diarahkan ke lokasi c:\xampp. Jika anda ingin menyimpannya di folder lain, anda dapat menekan tombol bergambar folder (Browse), kemudian klik tombol Next.



Kita akan menjumpai jendela tawaran untuk mempelajari lebih lanjut tentang Bitnami. Silahkan checklist jika ingin mempelajari lebih lanjut. Bitnami adalah pustaka dari aplikasi client server yang populer seperti misalnya CMS WordPress atau Drupal, dengan penawaran kemudahan

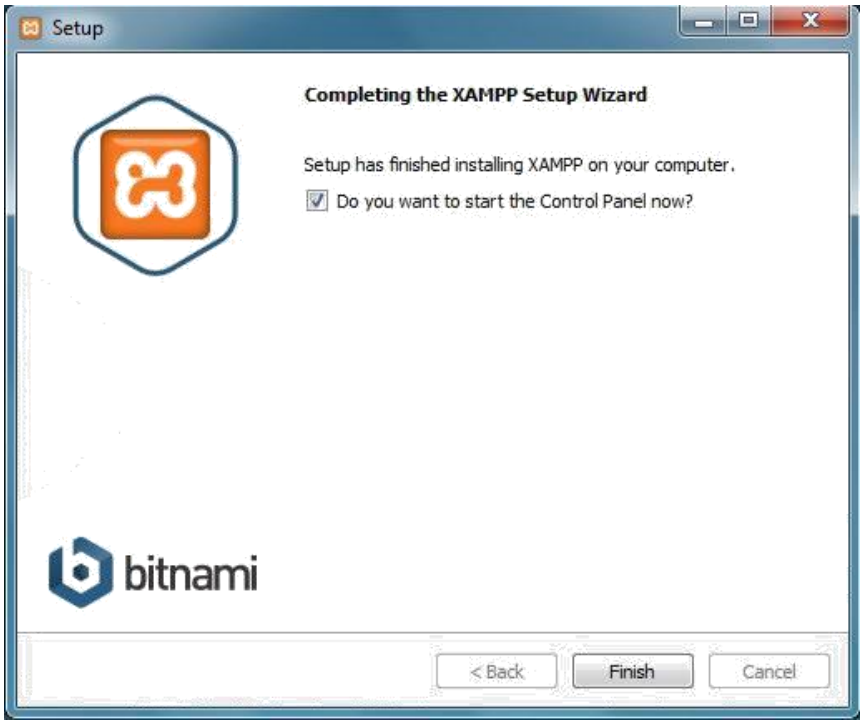
dalam installasi hanya dengan satu klik. Kemudian klik tombol Next.

Kemudian muncul jendela Ready To Install yang menunjukkan xampp sudah siap di install. Kemudian klik tombol Next.

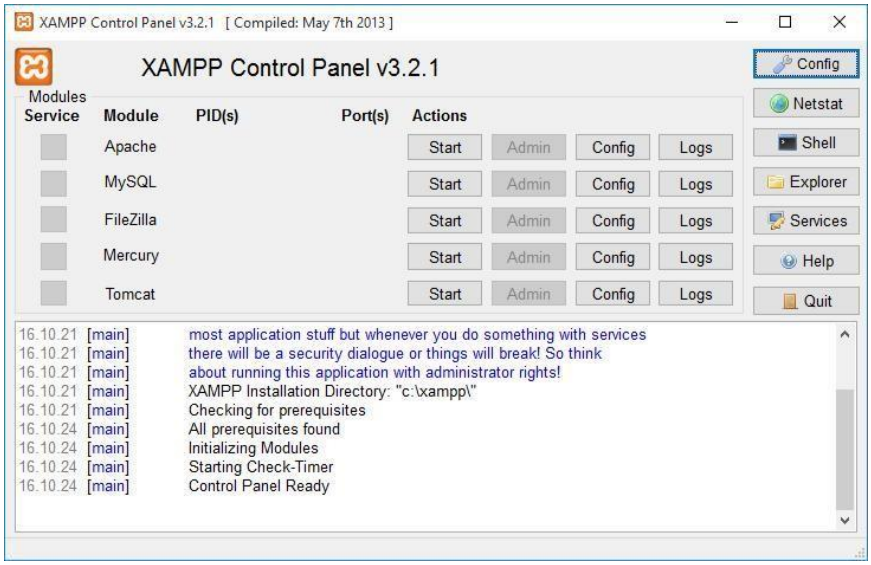


Dan proses installasi pun berjalan.

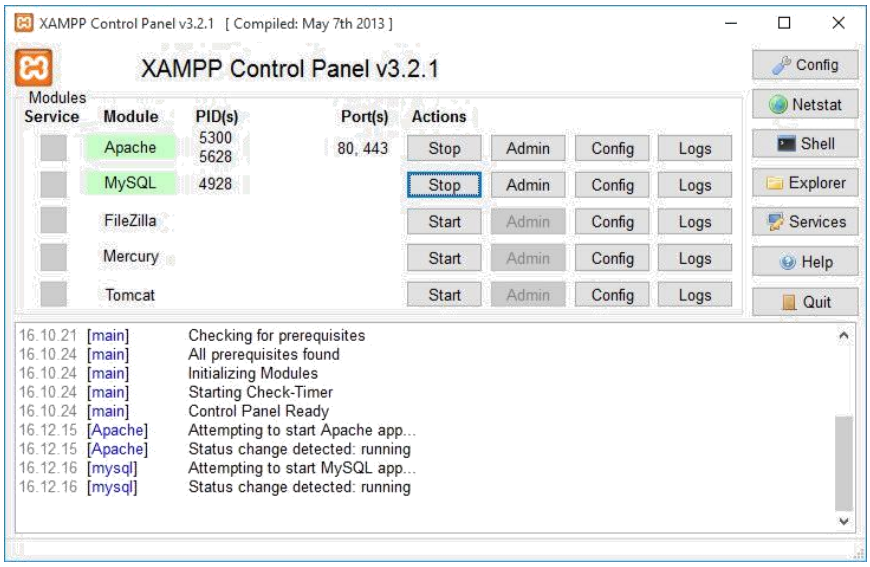
Tunggu hingga proses install selesai dan muncul jendela sebagai berikut.



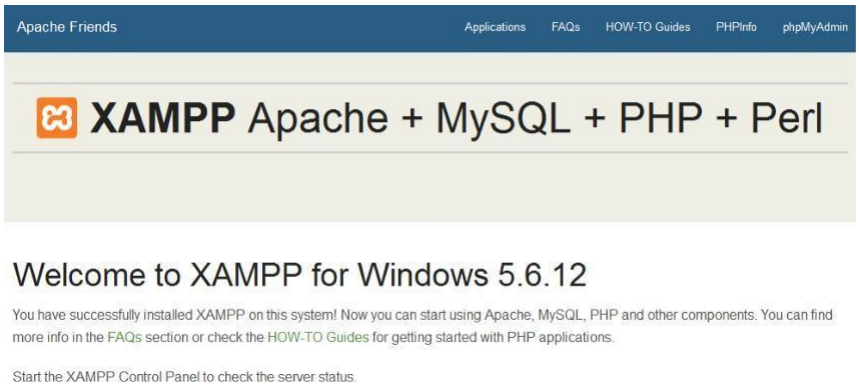
Klik tombol Finish setelah itu akan muncul jendela Xampp Control Panel yang berguna untuk menjalankan server.



Klik tombol Start pada kolom Actions untuk module Apache dan MySQL.



Untuk mengetahui apakah installasi telah berhasil atau tidak, ketikkan ‘localhost/’ pada browser, jika berhasil akan muncul halaman seperti gambar dibawah.



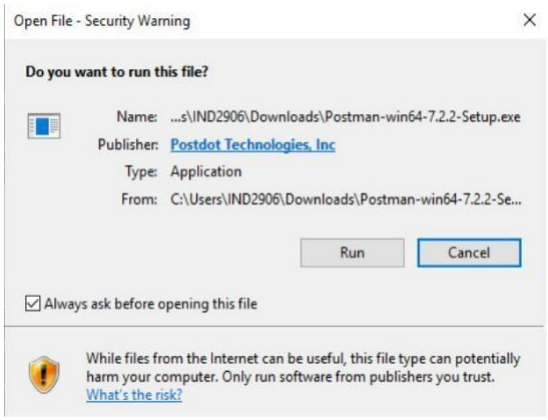
Install Postman

Postman adalah sebuah aplikasi fungsinya adalah sebagai REST Client atau istilahnya adalah aplikasi yang digunakan untuk melakukan uji coba REST API yang telah kita buat. Postman ini merupakan tools wajib bagi para developer yang bergerak pada pembuatan API, fungsi utama postman ini adalah sebagai GUI API Caller Pemanggil. namun sekarang postman juga menyediakan fitur lain yaitu Sharing Collection API for Documentation (free), Testing API (free), Realtime Collaboration Team (paid), Monitoring API (paid), Integration (paid).

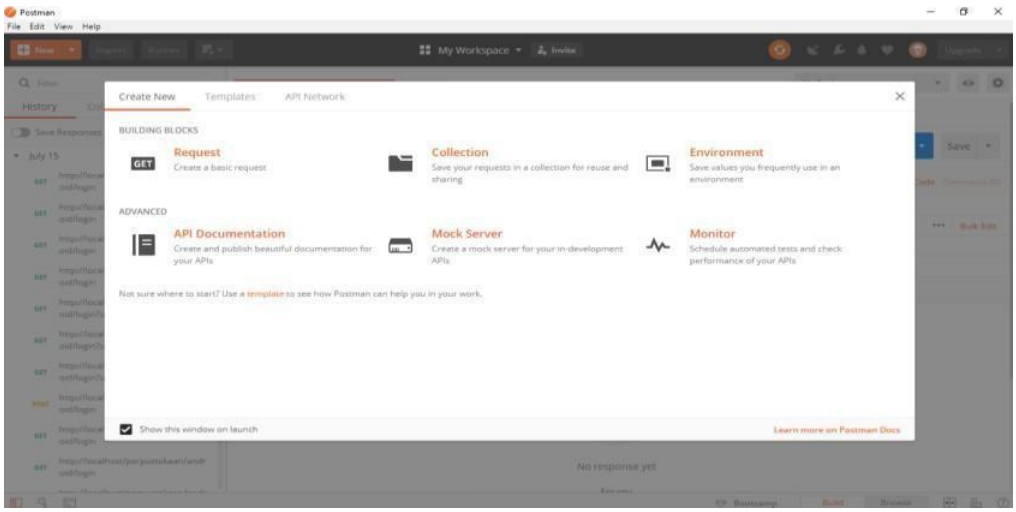
Postman tersedia sebagai aplikasi asli untuk sistem operasi macOS, Windows (32-bit dan 64-bit), dan Linux (32-bit dan 64-bit). Untuk mendapatkan aplikasi Postman, dapat diunduh

pada website resminya yanitu getpostman.com atau dapat diunduk pada halaman https://www.postman.com/downloads/

Setelah berhasil mengunduh paket instalasi postman, kemudian jalankan dengan cara klik dua kali. Pilih run jika muncul pop up seperti berikut :



Kemudian tunggu hingga proses instalasi selesai dan muncul seperti gambar berikut



API SPEC

API SPEC ini bermaksud untuk membuat standar API sebagai dokumentasi kepada pengembang baik itu frontend maupun backend

A. Registrasi

EndPoint	/registrasi
Method	POST

Header	• Content-Type: application/json

Body	{
	"nama" : "string",
	"email" : "string, unique",
	"password" : "string"
	}

Response	{
	"code" : "integer",
	"status" : "boolean",
	"data" : "string"
	}

B. Login

EndPoint	/login
Method	POST
Header	• Content-Type: application/json

Body	{
	"email" : "string"
	"password" : "string"
	}

Response	{
	"code" : "integer",
	"status" : "boolean",
	"data" : {
	"token" : "string",
	"user" : {
	"id" : "integer",
	"email" : "string",
	}
	}
	}

C. Produk

1. List Produk

EndPoint	/produk
Method	GET
Header	• Content-Type: application/json
Response	{ "code" : "integer", "status" : "boolean", "data" : [

```
            {
                "id" : "integer",
                "kode_produk" : "string",
                "nama_produk" : "string",
                "harga" : "integer",
            },
            {
                "id" : "integer",
                "kode_produk" : "string",
                "nama_produk" : "string",
                "harga" : "integer",
            }
    ]
}
```

2. Create Produk

EndPoint	/produk
Method	POST
Header	• Content-Type: application/json
Body	{ "kode_produk" : "string", "nama_produk" : "string", "harga" : "integer" }
Response	{ "code" : "integer", "status" : "boolean", "data" : { "id" : "integer", "kode_produk" : "string", "nama_produk" : "string", "harga" : "integer", } }

3. Update Produk

EndPoint	/produk/{id}/update
Method	POST
Header	• Content-Type: application/json
Body	{

```
    "kode_produk"  :  "string",
    "nama_produk"  :  "string",
    "harga" : "integer"
}
```

Response	{
	"code" : "integer",
	"status" : "boolean",
	"data" : "boolean"
	}

4. Show Produk

EndPoint	/produk/{id}
Method	GET
Header	• Content-Type: application/json

Response	{
	"code" : "integer",
	"status" : "boolean",
	"data" : {
	"id" : "integer",
	"kode_produk" : "string",
	"nama_produk" : "string",
	"harga" : "integer",
	}
	}

5. Delete Produk

EndPoint	/produk/{id}
Method	DELETE
Header	• Content-Type: application/json

Response	{
	"code" : "integer",
	"status" : "boolean",
	"data" : "boolean"
	}

Pembuatan Database

Buat database dengan nama : toko_api

Kemudian buat tabel-tabel dengan perintah sebagai berikut :

1. SQL membuat tabel member

```
CREATE table member (
    id INT NOT NULL AUTO_INCREMENT,
    nama VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    PRIMARY KEY(id)
);
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	*None*		AUTO_INCREMENT
2	nama	varchar(255)	utf8mb4_general_ci		No	*None*		
3	email	varchar(255)	utf8mb4_general_ci		No	*None*		
4	password	varchar(255)	utf8mb4_general_ci		No	*None*		

2. SQL membuat tabel member_token

```
CREATE table member_token (
    id INT NOT NULL AUTO_INCREMENT,
    member_id INT NOT NULL,
    auth_key VARCHAR(255) NOT NULL,
    FOREIGN KEY (member_id) REFERENCES member(id) on update cascade on delete no action,
    PRIMARY KEY(id)
);
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	*None*		AUTO_INCREMENT
2	member_id	int(11)			No	*None*		
3	auth_key	varchar(255)	utf8mb4_general_ci		No	*None*		

3. SQL membuat tabel produk

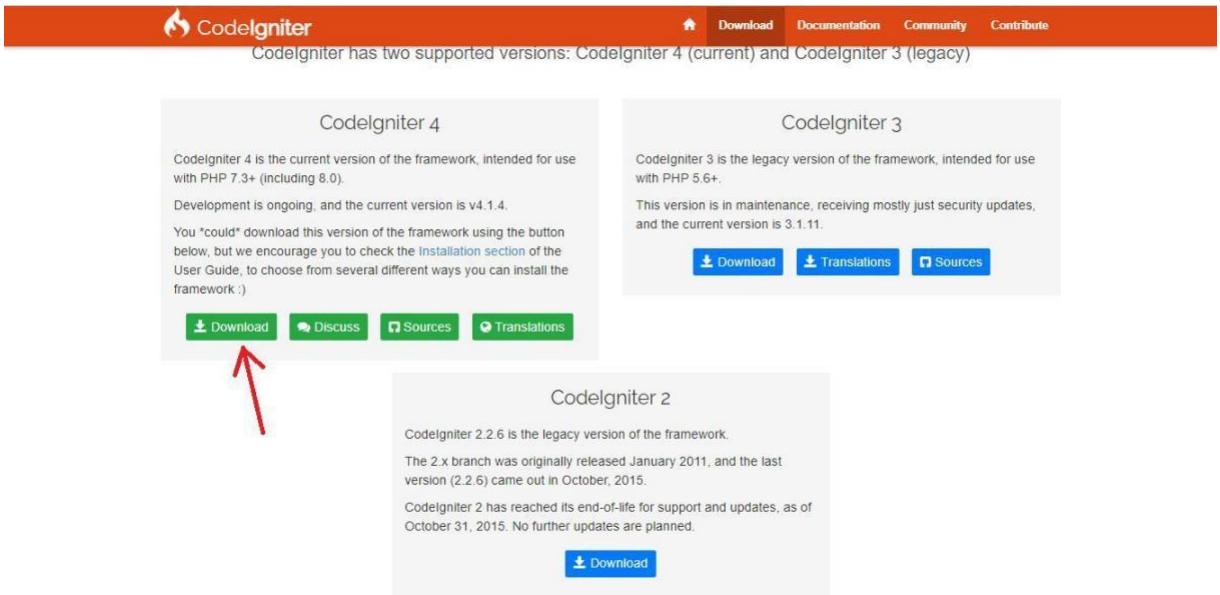
```
CREATE table produk (
    id INT NOT NULL AUTO_INCREMENT,
    kode_produk VARCHAR(255) NOT NULL,
    nama_produk VARCHAR(255) NOT NULL,
    harga INT NOT NULL,
    PRIMARY KEY(id)
);
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	kode_produk	varchar(255)	utf8mb4_general_ci		No	None		
3	nama_produk	varchar(255)	utf8mb4_general_ci		No	None		
4	harga	int(11)			No	None		

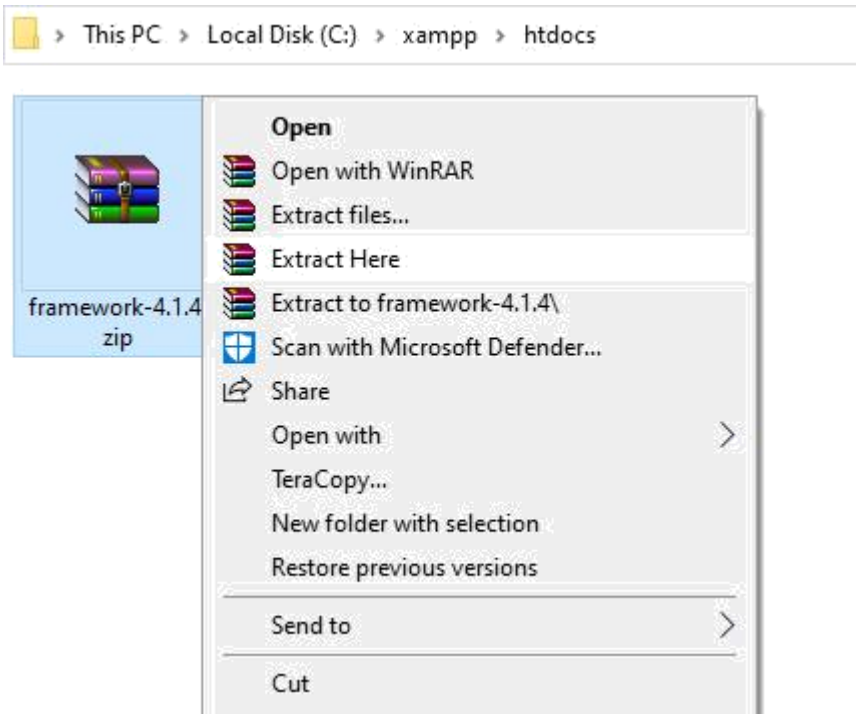
Installasi CodeIgniter 4 sebagai Restful API

Selanjutnya install projek CodeIgniter pada web server (pada folder C:\xampp\htdocs)

Framework CodeIgniter 4 dapat di unduh di https://codeigniter.com/download



Setelah di unduh ekstrak file CodeIgniter pada folder **C:\xampp\htdocs**



Kemudian ubah nama folder menjadi **toko-api.** Pada projek buka file **Database.php** yang terletak pada folder **app\config**, kemudian cari kode berikut

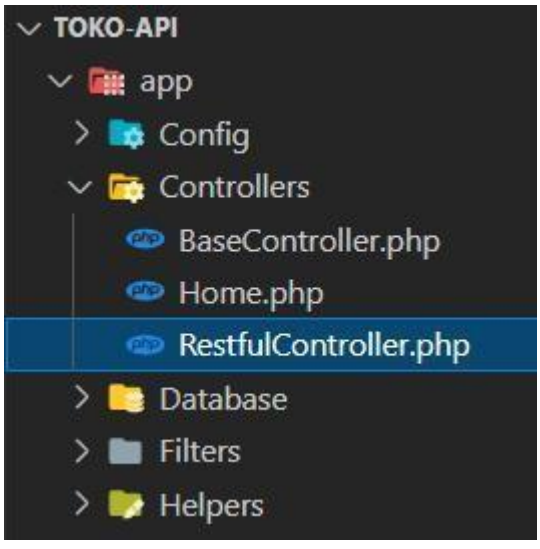
```
public $default = [
        'DSN'      => '',
        'hostname' => 'localhost',
        'username' => '',
        'password' => '',
        'database' => '',
        'DBDriver' => 'MySQLi',
        'DBPrefix' => '',
        'pConnect' => false,
        'DBDebug' => (ENVIRONMENT !== 'production'),
        'charset' => 'utf8',
        'DBCollat' => 'utf8_general_ci',
        'swapPre'  => '',
        'encrypt'  => false,
        'compress' => false,
        'strictOn' => false,
        'failover' => [],
        'port'     => 3306,
];
```

Untuk menghubungkan projek dengan database kita akan mengisikan **hostname**, **username**, **password** dan **database** menjadi

```
public $default = [
        'DSN'      => '',
        'hostname' => 'localhost',
        'username' => 'root',
        'password' => '',
        'database' => 'toko_api',
        'DBDriver' => 'MySQLi',
        'DBPrefix' => '',
        'pConnect' => false,
        'DBDebug' => (ENVIRONMENT !== 'production'),
        'charset' => 'utf8',
        'DBCollat' => 'utf8_general_ci',
        'swapPre' => '',
        'encrypt' => false,
        'compress' => false,
        'strictOn' => false,
        'failover' => [],
        'port'     => 3306,
];
```

Membuat hasil response

Buat sebuah file dengan nama RestfulControlller.php pada folder app\Controllers



Kemudian tambahkan kode pada file tersebut sehingga menjadi

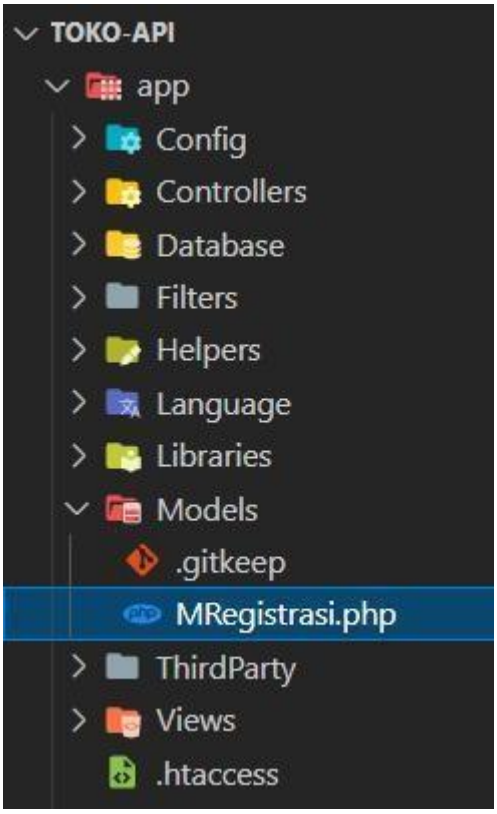
```
1.  <?php
2.
3.  namespace App\Controllers;
4.
5.  use CodeIgniter\RESTful\ResourceController;
6.
7.  class RestfulController extends ResourceController
8.  {
9.
10.     protected $format = 'json';
```

```
11.
12.     protected function responseHasil($code, $status, $data)
13.     {
14.         return $this->respond([
15.             'code' => $code,
16.             'status' => $status,
17.             'data' => $data
18.           ]);
19.       }
20. }
```

Registrasi

Membuat model Registrasi

Buat sebuah file dengan nama MRegistrasi.php pada folder app/Models

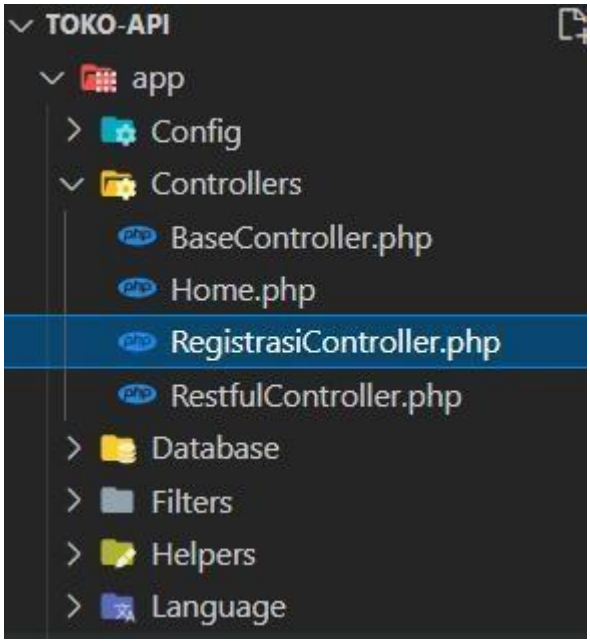


Kemudian pada file MRegistrasi.php ketikkan kode berikut

```
1. <?php
2.
3. namespace App\Models;
4.
5. use CodeIgniter\Model;
6.
7. class MRegistrasi extends Model
8. {
9.     protected $table = 'member';
10.     protected $allowedFields = ['nama', 'email', 'password'];
11. }
```

Membuat controller Registrasi

Buat sebuah file dengan nama RegistrasiController.php pada folder app\Controllers



Kemudian pada file RegistrasiController.php tersebut masukkan kode berikut

```
1.  <?php
2.
3.  namespace App\Controllers;
4.
5.  use App\Models\MRegistrasi;
6.
7.  class RegistrasiController extends RestfulController
8.  {
9.
10.     public function registrasi()
11.     {
12.         $data = [
13.             'nama' => $this->request->getVar('nama'),
14.             'email' => $this->request->getVar('email'),
15.             'password' => password_hash($this->request->getVar('password'),
    PASSWORD_DEFAULT)
16.         ];
17.
18.         $model = new MRegistrasi();
19.         $model->save($data);
20.         return $this->responseHasil(200, true, "Registrasi Berhasil");
21.     }
22. }
```

Menambah route Registrasi

Buka file Routes.php pada folder app\Config

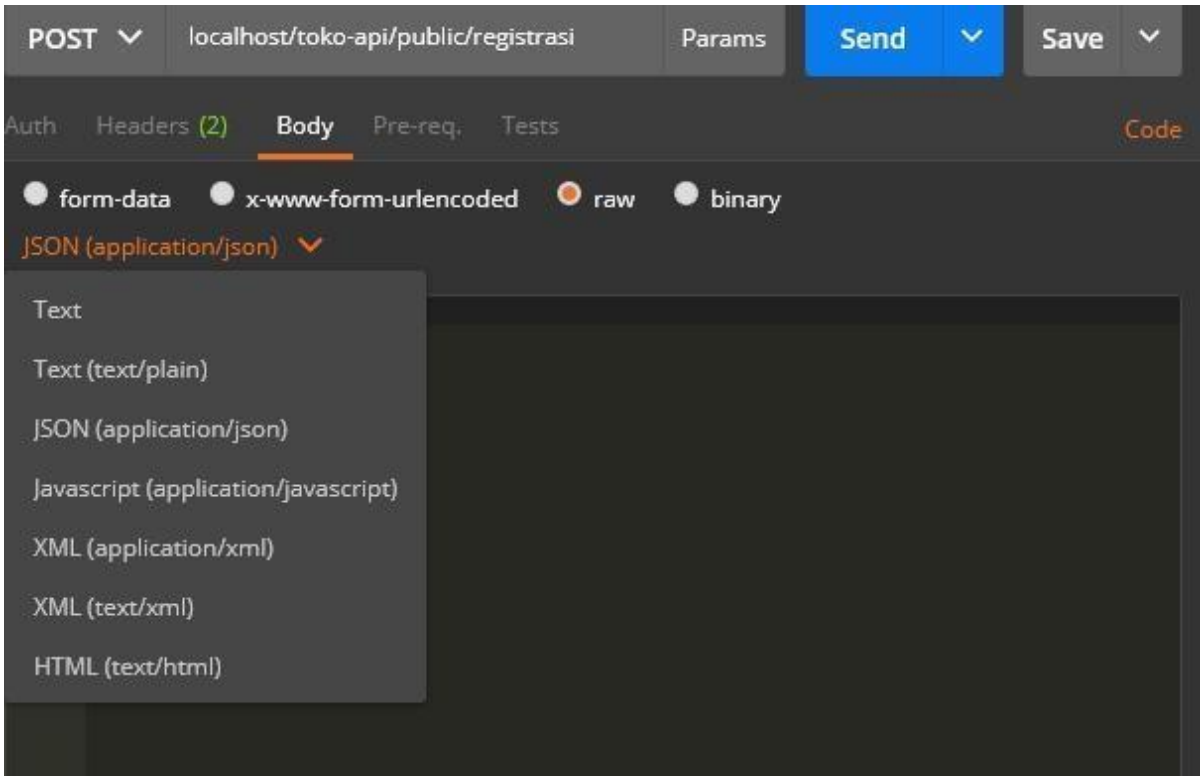
Kemudian lihat baris ke 34, telah terdapat routing dengan method get, kita akan menambahkan routing untuk registrasi sehingga kode menjadi seperti berikut

```
 /
 *
 * ____________________________________________________________
 * Route Definitions
* ____________________________________________________________
 */

// We get a performance increase by specifying the default
// route since we don't have to scan directories.
$routes->get('/', 'Home::index');
$routes->post('/registrasi', 'RegistrasiController::registrasi);
```

Pada baris terakhir kita menambahkan routing registrasi agar dapat diakses dengan method POST. Untuk mengakses registrasi, kita gunakan Postman dengan alamat url localhost/toko-api/public/registrasi dengan method POST

Adapun langkah menggunakan postman untuk menguji Rest API yang telah dibuat adalah sebagai berikut:

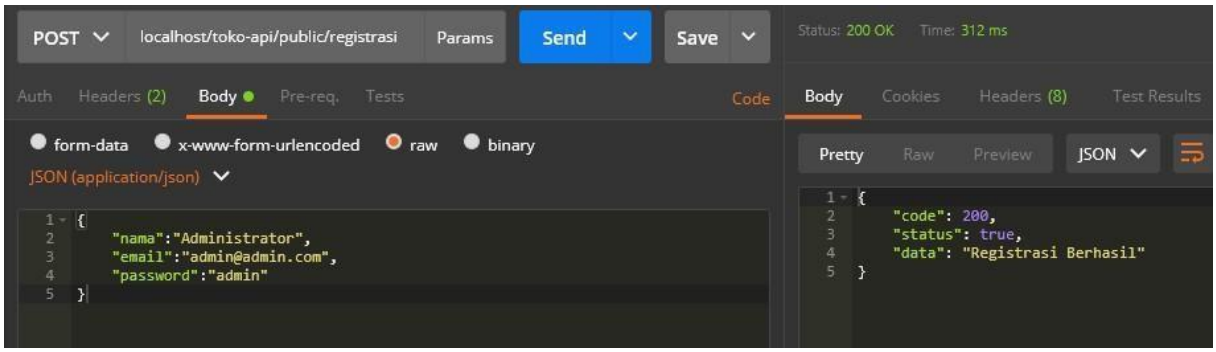
1. Buka aplikasi Postman yang telah terinstall
2. Masukkan alamat url untuk melakukan registrasi toko-api yang telah kita buat yaitu http://localhost/toko-api/public/registrasi
3. Selanjutnya pilih pengujian dengan type POST
4. Kemudian klik Body yang berada pada bagian bawah inputan url, kemudian pilih raw
5. Kemudian dibagian bawah pilihan raw, pilih JSON (application/json)



6. Kemudian isi dengan format JSON sesuai dengan field request pada RegistrasiController pada fungsi registrasi yaitu *nama*, *email*, *password* kemudian klik tombol Send

Contoh isian data

```
{
        "nama":"Administrator",
        "email":"admin@admin.com",
        "password":"admin"
}
```



Login

Membuat model Member

Buat sebuah file dengan nama MMember.php pada folder app\Models dan ketikkan kode berikut

```
1.  <?php
2.
3.  namespace App\Models;
4.
5.  use CodeIgniter\Model;
6.
7.  class MMember extends Model
8.  {
9.      protected $table = 'member';
10. }
```

Membuat model Login

Buat sebuah file dengan nama MLogin.php pada folder app/Models dan ketikkan kode berikut

```php
<?php

namespace App\Models;

use CodeIgniter\Model;

class MLogin extends Model
{
    protected $table = 'member_token';
    protected $allowedFields = ['member_id', 'auth_key'];
}
```

Membuat controller Login

Buat sebuah file dengan nama LoginController.php pada folder app/Controllers dan ketikkan kode berikut

```php
<?php

namespace App\Controllers;

use App\Models\MLogin;
use App\Models\MMember;

class LoginController extends RestfulController
{

    public function login()
    {
        $email = $this->request->getVar('email');
        $password = $this->request->getVar('password');

        $model = new MMember();
        $member = $model->where(['email' => $email])->first();
        if (!$member) {
            return $this->responseHasil(400, false, "Email tidak ditemukan");
        }
        if (!password_verify($password, $member['password'])) {
            return $this->responseHasil(400, false, "Password tidak valid");
        }

        $login = new MLogin();
        $auth_key = $this->RandomString();
        $login->save([
            'member_id' => $member['id'],
            'auth_key' => $auth_key
        ]);
        $data = [
            'token' => $auth_key,
            'user' => [
                'id' => $member['id'],
                'email' => $member['email'],
            ]
        ];
        return $this->responseHasil(200, true, $data);
    }

    private function RandomString($length = 100)
    {
        $karakkter = '012345678dssd9abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
        $panjang_karakter = strlen($karakkter);
        $str = '';
```

```
        for ($i = 0; $i < $length; $i++) {
            $str .= $karakkter[rand(0, $panjang_karakter - 1)];
        }
        return $str;
    }
}
```

Menambahkan route Login

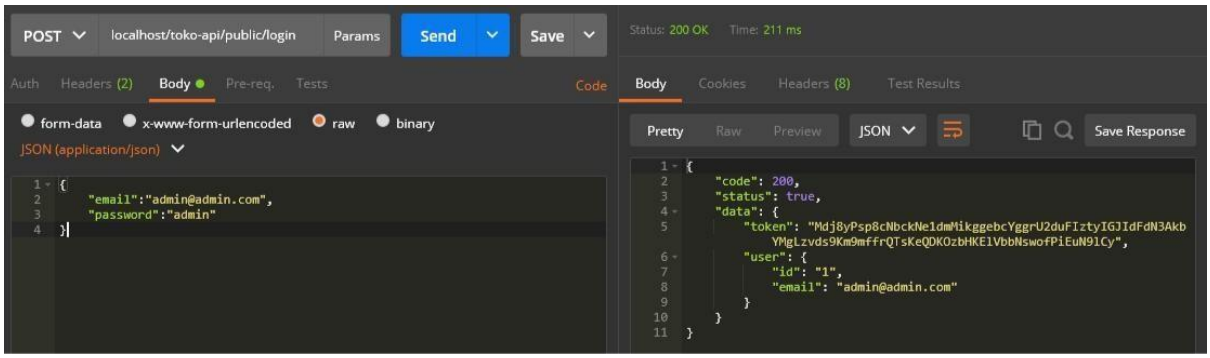
Pada route tambahkan kode untuk mengakses login sehingga menjadi sebagai berikut

```
$routes->post('/registrasi', 'RegistrasiController::registrasi');
$routes->post('/login', 'LoginController::login');
```

Mencoba Rest

Silahkan coba Rest API dengan memasukkan url http://localhost/toko-api/public/login

dengan method POST



CRUD Produk

Membuat model Produk

Buat sebuah file dengan nama MProduk.php pada folder app/Models dan ketikkan kode

berikut

```
<?php

namespace App\Models;

use CodeIgniter\Model;

class MProduk extends Model
{
    protected $table = 'produk';
    protected $primaryKey = 'id';
    protected $allowedFields = ['kode_produk', 'nama_produk', 'harga'];
}
```

Membuat controller produk

Buat sebuah file dengan nama ProdukController pada folder app/Controllers dan ketikkan

kode berikut

```
1. <?php
2.
3. namespace App\Controllers;
4.
5. use App\Models\MProduk;
6.
7.  class ProdukController extends RestfulController
8. {
9.
10. }
```

Selanjutnya pada class ProdukController kita akan menambahkan fungsi (function) CRUD produk, yaitu:

Membuat fungsi create produk

```
public function create()
{
        $data = [
            'kode_produk'   =>   $this->request->getVar('kode_produk'),
            'nama_produk'   =>   $this->request->getVar('nama_produk'),
            'harga' => $this->request->getVar('harga')
        ];

        $model = new MProduk();
        $model->insert($data);
        $produk = $model->find($model->getInsertID());
}       return $this->responseHasil( 200, true , $produk);
```

Membuat fungsi list produk

```
public function list()
{
     $model = new MProduk();
     $produk = $model->findAll();
     return $this->responseHasil(200, true,
$produk); }
```

Membuat fungsi tampil produk

```
public function detail($id)
{
     $model = new MProduk();
     $produk = $model->find($id);
     return $this->responseHasil(200,  true, $produk);
}
```

Membuat fungsi update produk

```
public function ubah($id)
{
    $data = [
        'kode_produk'   =>   $this->request->getVar('kode_produk'),
        'nama_produk'   =>   $this->request->getVar('nama_produk'),
        'harga' => $this->request->getVar('harga')
    ];

    $model = new MProduk();
    $model->update($id, $data);
    $produk = $model->find($id);

    return $this->responseHasil(200,  true,
$produk); }
```

Membuat fungsi delete produk

```
public function hapus($id)
{
    $model = new MProduk();
    $produk = $model->delete($id);

    return $this->responseHasil(200, true,
$produk); }
```

Sehingga adapun keseluruhan kode ProdukController.php adalah sebagai berikut

```
1.  <?php
2.
3.  namespace App\Controllers;
4.
5.  use App\Models\MProduk;
6.
7.  class ProdukController extends RestfulController
8.  {
9.      public function create()
10.     {
11.         $data = [
12.             'kode_produk' => $this->request->getVar('kode_produk'),
13.             'nama_produk' => $this->request->getVar('nama_produk'),
14.         'harga' => $this->request->getVar('harga')
15.         ];
16.
17.         $model = new MProduk();
18.         $model->insert($data);
19.         $produk = $model->find($model->getInsertID());
20.         return $this->responseHasil(200, true, $produk);
21.     }
22.
23.     public function list()
24.     {
25.         $model = new MProduk();
26.         $produk = $model->findAll();
27.         return $this->responseHasil(200, true, $produk);
28.     }
29.
30.     public function detail($id)
31.     {
32.         $model = new MProduk();
33.         $produk = $model->find($id);
34.         return $this->responseHasil(200, true, $produk);
35.     }
36.
37.     public function ubah($id)
38.     {
39.         $data = [
40.             'kode_produk' => $this->request->getVar('kode_produk'),
41.             'nama_produk' => $this->request->getVar('nama_produk'),
42.         'harga' => $this->request->getVar('harga')
43.         ];
44.
45.         $model = new MProduk();
46.         $model->update($id, $data);
47.         $produk = $model->find($id);
48.
49.         return $this->responseHasil(200, true, $produk);
50.     }
```

```
51.
52.     public function hapus($id)
53.     {
54.         $model = new MProduk();
55.         $produk = $model->delete($id);
56.
57.         return $this->responseHasil(200, true, $produk);
58.     }
59. }
```

Menambahkan route produk

Agar ProdukController dapat diakses, selanjutnya tambah route untuk mengakses produk

pada file app/Config/Routes.php

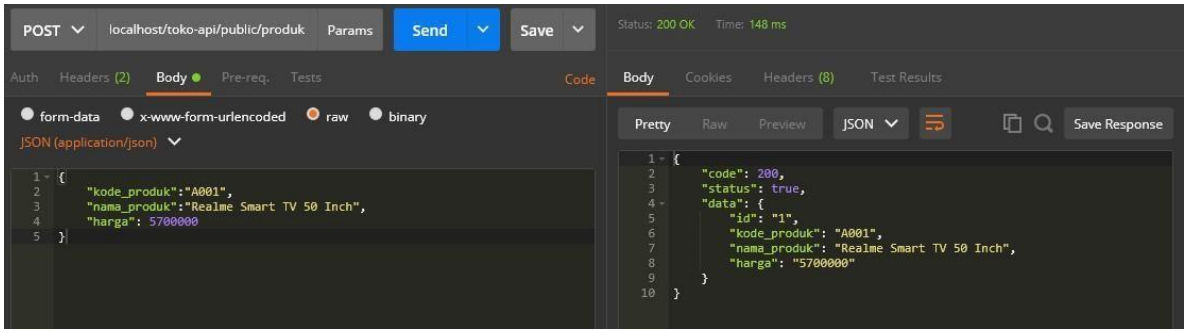
```
$routes->group('produk', function ($routes) {
    $routes->post('/', 'ProdukController::create');
    $routes->get('/', 'ProdukController::list');
    $routes->get('(:segment)', 'ProdukController::detail/$1');
    $routes->put('(:segment)', 'ProdukController::ubah/$1');
    $routes->delete('(:segment)', 'ProdukController::hapus/$1');
});
```

Adapun keseluruhan kode pada app/Config/Routes.php adalah sebagai berikut

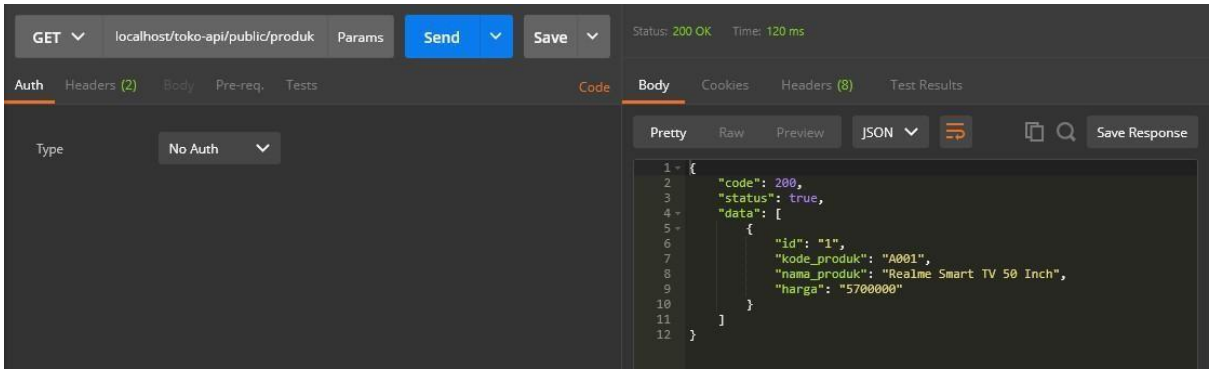
```
$routes->get('/', 'Home::index');
$routes->post('/registrasi', 'RegistrasiController::registrasi');
$routes->post('/login', 'LoginController::login');
$routes->group('produk', function ($routes) {
    $routes->post('/', 'ProdukController::create');
    $routes->get('/', 'ProdukController::list');
    $routes->get('(:segment)', 'ProdukController::detail/$1');
    $routes->put('(:segment)', 'ProdukController::ubah/$1');
    $routes->delete('(:segment)', 'ProdukController::hapus/$1');
});
```

Mencoba Rest

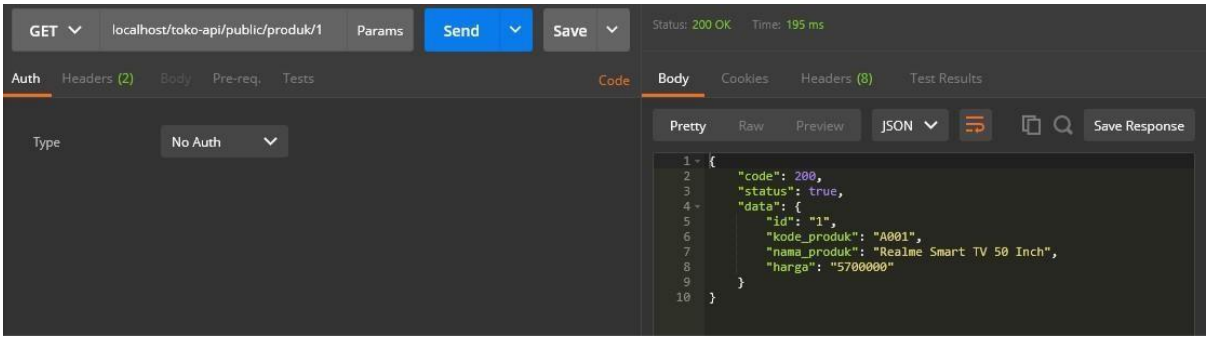
Create Produk (localhost/toko-api/public/produk) dengan method POST



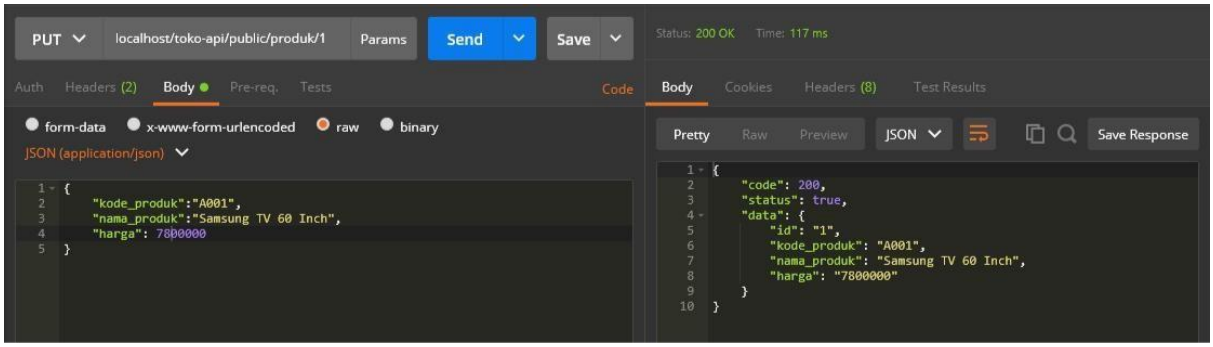
List Produk (localhost/toko-api/public/produk) dengan method GET



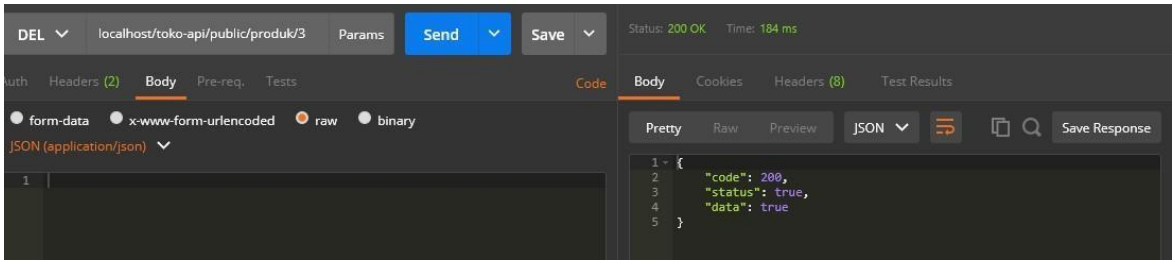
Show Produk (localhost/toko-api/public/produk/{id}) dengan method GET



Update Produk (localhost/toko-api/public/produk/{id}) dengan method PUT

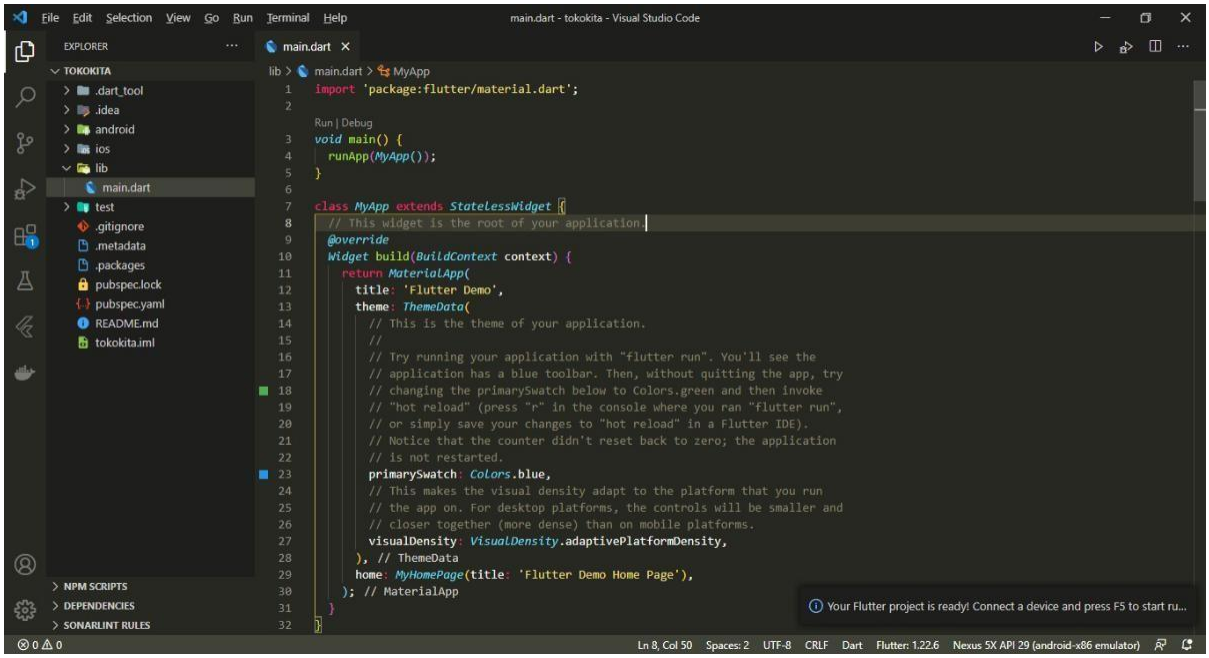


Delete Produk (localhost/toko-api/public/produk) dengan method DELETE



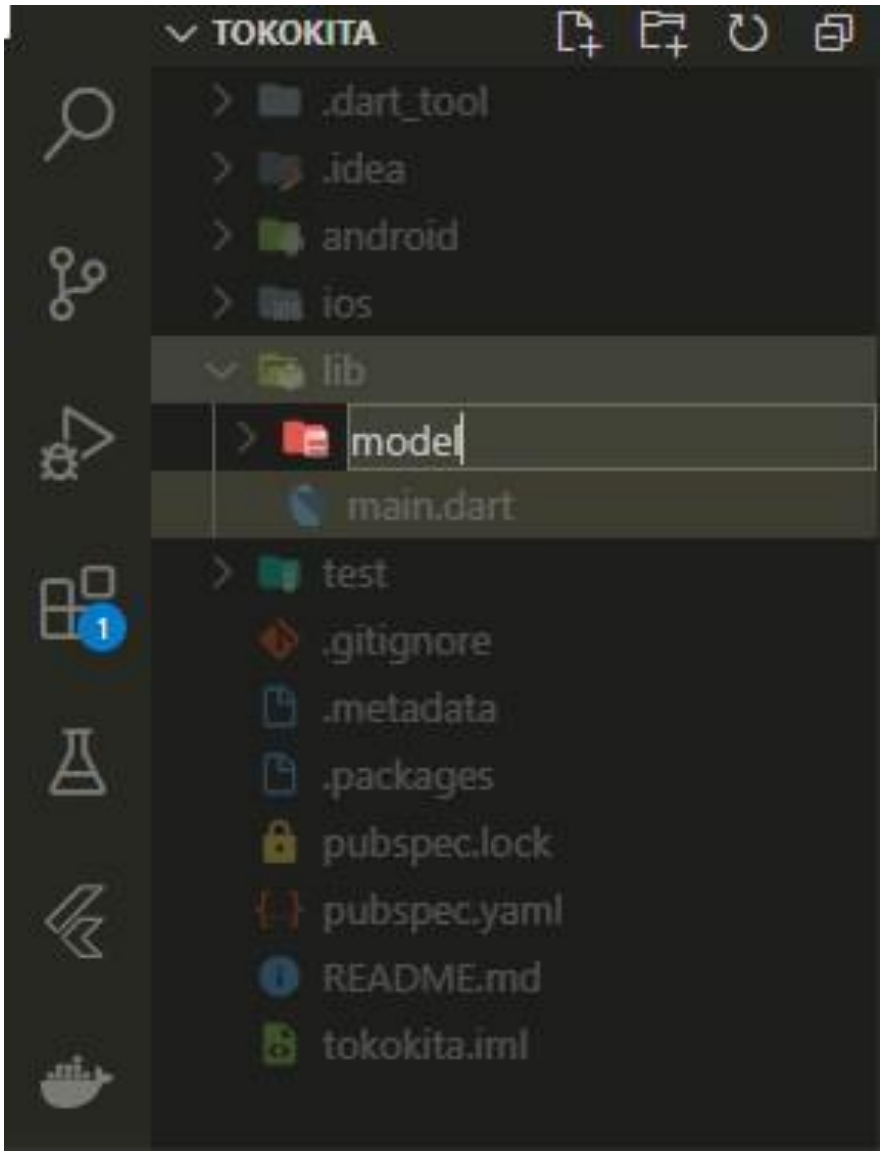
Membuat projek flutter tokokita

Buat sebuah projek flutter dengan nama tokokita



Membuat Model

Buat folder dengan nama model pada folder lib



Login

Buat sebuah file dengan nama login.dart pada folder model. Kemudian masukkan kode berikut

```
class Login {
  int? code;
  bool? status;
  String? token;
  int? userID;
  String? userEmail;
  Login({this.code, this.status, this.token, this.userID, this.userEmail});
  factory Login.fromJson(Map<String, dynamic> obj) {
    if (obj['code'] == 200) {
      return Login(
          code: obj['code'],
          status: obj['status'],
          token: obj['data']['token'],
          userID: int.parse(obj['data']['user']['id']),
```

```
          userEmail: obj['data']['user']['email']);
    } else {
      return Login(
        code: obj['code'],
        status: obj['status'],
      );
    }
  }
}
```

Registrasi

Buat sebuah file dengan nama registrasi.dart pada folder model. Kemudian masukkan kode berikut

```
class Registrasi {
  int? code;
  bool? status;
  String? data;
  Registrasi({this.code, this.status, this.data});
  factory Registrasi.fromJson(Map<String, dynamic> obj) {
    return Registrasi(
        code: obj['code'], status: obj['status'], data: obj['data']);
  }
}
```

Produk

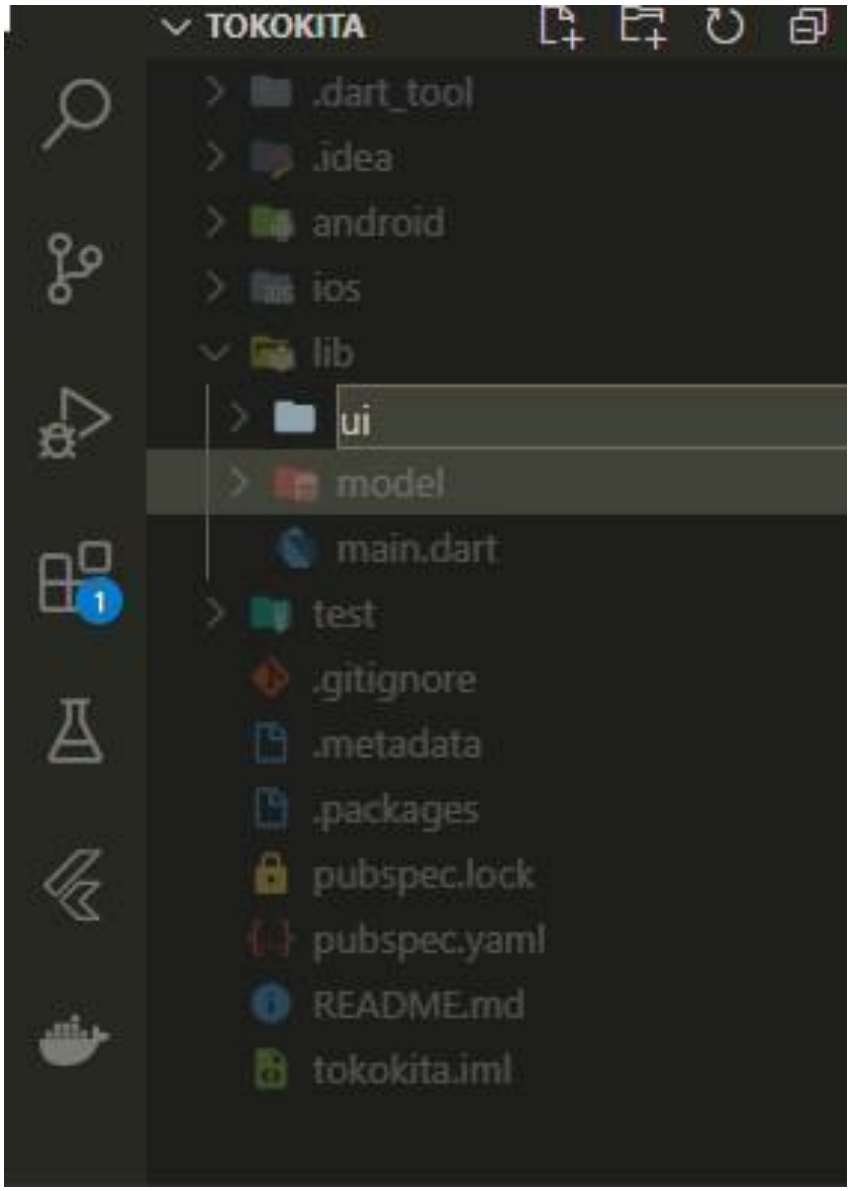
Buat sebuah file dengan nama produk.dart pada folder model. Kemudian ketikkan kode berikut

```
class Produk {
  String? id;
  String? kodeProduk;
  String? namaProduk;
  var hargaProduk;
  Produk({this.id, this.kodeProduk, this.namaProduk, this.hargaProduk});
  factory Produk.fromJson(Map<String, dynamic> obj) {
    return Produk(
        id: obj['id'],
        kodeProduk: obj['kode_produk'],
        namaProduk: obj['nama_produk'],
        hargaProduk: obj['harga']);
```

```
  }
}
```

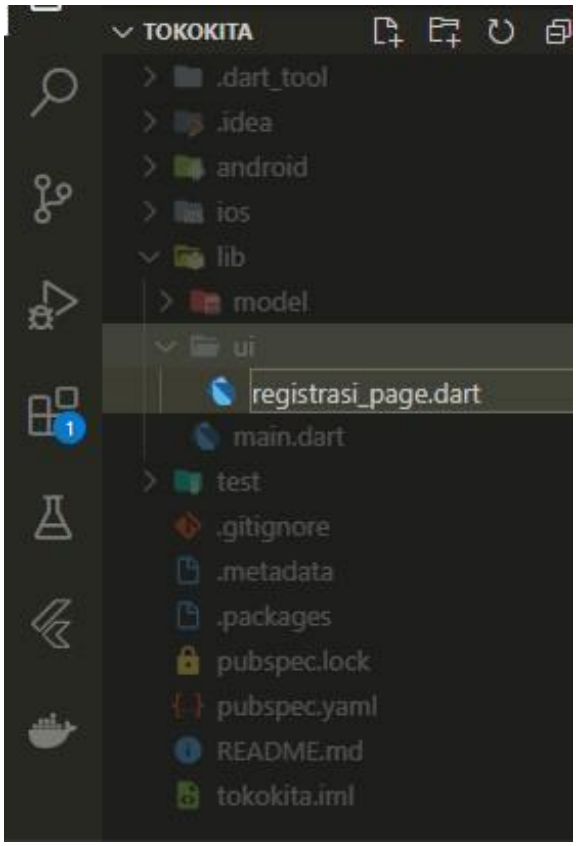
Membuat Halaman

Pertama kita akan memecah bagian-bagian kode menjadi beberapa bagian, adapun untuk tampilan, dikelompokkan kedalam folder ui.



Registrasi

Buat sebuah file dengan nama registrasi_page.dart pada folder ui.



Pada file registrasi_page.dart ketikkan kode berikut

```
import 'package:flutter/material.dart';

class RegistrasiPage extends StatefulWidget {
  const RegistrasiPage({Key? key}) : super(key: key);

  @override
  _RegistrasiPageState createState() => _RegistrasiPageState();
}

class _RegistrasiPageState extends State<RegistrasiPage> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;

  final _namaTextboxController = TextEditingController();
  final _emailTextboxController = TextEditingController();
  final _passwordTextboxController = TextEditingController();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Registrasi"),
```

```
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                _namaTextField(),
                _emailTextField(),
                _passwordTextField(),
                _passwordKonfirmasiTextField(),
                _buttonRegistrasi()
              ],
            ),
          ),
        ),
      ),
    );
  }

//Membuat Textbox Nama
  Widget _namaTextField() {
    return TextFormField(
      decoration: const InputDecoration(labelText: "Nama"),
      keyboardType: TextInputType.text,
      controller: _namaTextboxController,
      validator: (value) {
        if (value!.length < 3) {
          return "Nama harus diisi minimal 3 karakter";
        }
        return null;
      },
    );
  }

  //Membuat Textbox email
  Widget _emailTextField() {
    return TextFormField(
      decoration: const InputDecoration(labelText: "Email"),
      keyboardType: TextInputType.emailAddress,
```

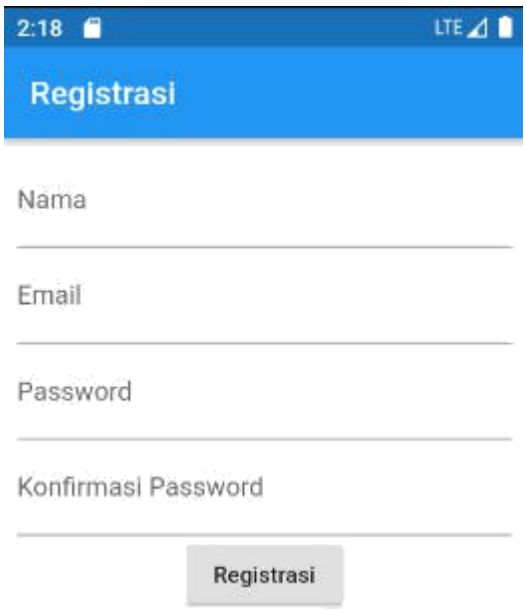
```
      controller: _emailTextboxController,
      validator: (value) {
        //validasi harus diisi
        if (value!.isEmpty) {
          return 'Email harus diisi';
        }
        //validasi email
        Pattern pattern =
            r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$';
        RegExp regex = RegExp(pattern.toString());
        if (!regex.hasMatch(value)) {
          return "Email tidak valid";
        }
        return null;
      },
    );
  }

//Membuat Textbox password
  Widget _passwordTextField() {
    return TextFormField(
      decoration: const InputDecoration(labelText: "Password"),
      keyboardType: TextInputType.text,
      obscureText: true,
      controller: _passwordTextboxController,
      validator: (value) {
        //jika karakter yang dimasukkan kurang dari 6 karakter
        if (value!.length < 6) {
          return "Password harus diisi minimal 6 karakter";
        }
        return null;
      },
    );
  }

//membuat textbox Konfirmasi Password
  Widget _passwordKonfirmasiTextField() {
    return TextFormField(
      decoration: const InputDecoration(labelText: "Konfirmasi Password"),
      keyboardType: TextInputType.text,
      obscureText: true,
```

```
      validator: (value) {
        //jika inputan tidak sama dengan password
        if (value != _passwordTextboxController.text) {
          return "Konfirmasi Password tidak sama";
        }
        return null;
      },
    );
  }

//Membuat Tombol Registrasi
  Widget _buttonRegistrasi() {
    return ElevatedButton(
        child: const Text("Registrasi"),
        onPressed: () {
          var validate = _formKey.currentState!.validate();

        });
  }
    setState(() {
      _isLoading = false;
    });
  }
}
```

Untuk mencoba halaman registrasi_page, buka file main.dart kemudian ubah kode menjadi seperti berikut ini

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/registrasi_page.dart';
3.
4. void main() {
5.   runApp(const MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   const MyApp({Key? key}) : super(key: key);
10.
11.  @override
12.  Widget build(BuildContext context) {
13.    return const MaterialApp(
14.      title: 'Toko Kita',
15.      debugShowCheckedModeBanner: false,
16.      home: RegistrasiPage(),
17.    );
18.  }
19. }
```

Dan hasilnya seperti berikut



Login

Buat sebuah file dengan nama login_page.dart pada folder ui dengan kode berikut

```
import 'package:flutter/material.dart';
import 'package:tokokita/ui/registrasi_page.dart';

class LoginPage extends StatefulWidget {
  const LoginPage({Key? key}) : super(key: key);
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  final _emailTextboxController = TextEditingController();
  final _passwordTextboxController = TextEditingController();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Login'),
      ),
      body: SingleChildScrollView(
```

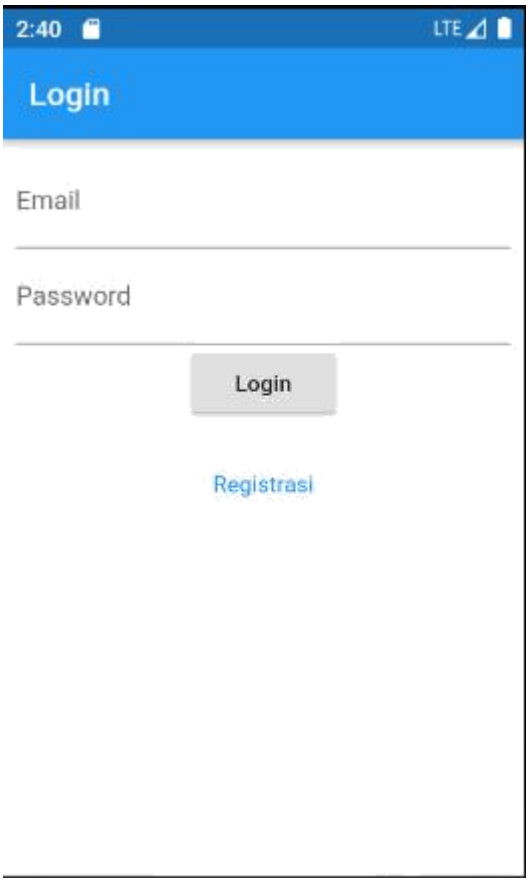
```
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              _emailTextField(),
              _passwordTextField(),
              _buttonLogin(),
              const SizedBox(
                height: 30,
              ),
              _menuRegistrasi()
            ],
          ),
        ),
      ),
    ),
  );
}

//Membuat Textbox email
Widget _emailTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Email"),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    validator: (value) {
      //validasi harus diisi
      if (value!.isEmpty) {
        return 'Email harus diisi';
      }
      return null;
    },
  );
}

//Membuat Textbox password
Widget _passwordTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Password"),
    keyboardType: TextInputType.text,
```

```
      obscureText: true,
      controller: _passwordTextboxController,
      validator: (value) {
        //jika karakter yang dimasukkan kurang dari 6 karakter
        if (value!.isEmpty) {
          return "Password harus diisi";
        }
        return null;
      },
    );
  }

  //Membuat Tombol Login
  //Membuat Tombol Login
  Widget _buttonLogin() {
    return ElevatedButton(
        child: const Text("Login"),
        onPressed: () {
          var validate = _formKey.currentState!.validate();

        });
  }

  // Membuat menu untuk membuka halaman registrasi
  Widget _menuRegistrasi() {
    return Center(
      child: InkWell(
        child: const Text(
          "Registrasi",
          style: TextStyle(color: Colors.blue),
        ),
        onTap: () {
          Navigator.push(context,
              MaterialPageRoute(builder: (context) => const RegistrasiPage()));
        },
      ),
    );
  }
}
```

Untuk mencobanya modifikasi file main.dart dimana pada bagian home akan memanggil LoginPage()

```dart
import 'package:flutter/material.dart';
import 'package:tokokita/ui/login_page.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: 'Toko Kita',
      debugShowCheckedModeBanner: false,
      home: LoginPage(),
    );
  }
}
```

Pada saat dijalankan akan terdapat link untuk membuka halaman registrasi pada bagian bawah form



Form Produk

Form produk yang akan kita buat berikut ini memiliki 2 fungsi yaitu untuk menambah data produk dan mengubah data produk

Buat sebuah file dengan nama produk_form.dart pada folder ui dengan kode berikut

```dart
import 'package:flutter/material.dart';
import 'package:tokokita/model/produk.dart';

// ignore: must_be_immutable
class ProdukForm extends StatefulWidget {
  Produk? produk;
  ProdukForm({Key? key, this.produk}) : super(key: key);
  @override
  _ProdukFormState createState() => _ProdukFormState();
}

class _ProdukFormState extends State<ProdukForm> {
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  String judul = "TAMBAH PRODUK";
  String tombolSubmit = "SIMPAN";
  final _kodeProdukTextboxController = TextEditingController();
  final _namaProdukTextboxController = TextEditingController();
```

```dart
final _hargaProdukTextboxController = TextEditingController();
@override
void initState() {
  super.initState();
  isUpdate();
}

isUpdate() {
  if (widget.produk != null) {
    setState(() {
      judul = "UBAH PRODUK";
      tombolSubmit = "UBAH";
      _kodeProdukTextboxController.text = widget.produk!.kodeProduk!;
      _namaProdukTextboxController.text = widget.produk!.namaProduk!;
      _hargaProdukTextboxController.text =
          widget.produk!.hargaProduk.toString();
    });
  } else {
    judul = "TAMBAH PRODUK";
    tombolSubmit = "SIMPAN";
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text(judul)),
    body: SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              _kodeProdukTextField(),
              _namaProdukTextField(),
              _hargaProdukTextField(),
              _buttonSubmit()
            ],
          ),
        ),
      ),
```

```
    ),
  );
 }

 //Membuat Textbox Kode Produk
 Widget _kodeProdukTextField() {
   return TextFormField(
     decoration: const InputDecoration(labelText: "Kode Produk"),
     keyboardType: TextInputType.text,
     controller: _kodeProdukTextboxController,
     validator: (value) {
       if (value!.isEmpty) {
         return "Kode Produk harus diisi";
       }
       return null;
     },
   );
 }

 //Membuat Textbox Nama Produk
 Widget _namaProdukTextField() {
   return TextFormField(
     decoration: const InputDecoration(labelText: "Nama Produk"),
     keyboardType: TextInputType.text,
     controller: _namaProdukTextboxController,
     validator: (value) {
       if (value!.isEmpty) {
         return "Nama Produk harus diisi";
       }
       return null;
     },
   );
 }

//Membuat Textbox Harga Produk
 Widget _hargaProdukTextField() {
   return TextFormField(
     decoration: const InputDecoration(labelText: "Harga"),
     keyboardType: TextInputType.number,
     controller: _hargaProdukTextboxController,
     validator: (value) {
       if (value!.isEmpty) {
```

```
          return "Harga harus diisi";
        }
        return null;
      },
    );
  }

  //Membuat Tombol Simpan/Ubah
  Widget _buttonSubmit() {
    return OutlinedButton(
        child: Text(tombolSubmit),
        onPressed: () {
          var validate = _formKey.currentState!.validate();

        });
  }
```

Detail Produk

Buat sebuah file dengan nama produk_detail.dart pada folder ui dengan kode berikut

```
import 'package:flutter/material.dart';
import 'package:tokokita/model/produk.dart';
import 'package:tokokita/ui/produk_form.dart';

// ignore: must_be_immutable
class ProdukDetail extends StatefulWidget {
  Produk? produk;

  ProdukDetail({Key? key, this.produk}) : super(key: key);

  @override
  _ProdukDetailState createState() => _ProdukDetailState();
}

class _ProdukDetailState extends State<ProdukDetail> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Detail Produk'),
      ),
      body: Center(
        child: Column(
```

```
        children: [
          Text(
            "Kode : ${widget.produk!.kodeProduk}",
            style: const TextStyle(fontSize: 20.0),
          ),
          Text(
            "Nama : ${widget.produk!.namaProduk}",
            style: const TextStyle(fontSize: 18.0),
          ),
          Text(
            "Harga : Rp. ${widget.produk!.hargaProduk.toString()}",
            style: const TextStyle(fontSize: 18.0),
          ),
          _tombolHapusEdit()
        ],
      ),
    ),
  );
}

Widget _tombolHapusEdit() {
  return Row(
    mainAxisSize: MainAxisSize.min,
    children: [
      // Tombol Edit
      OutlinedButton(
        child: const Text("EDIT"),
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => ProdukForm(
                produk: widget.produk!,
              ),
            ),
          );
        },
      ),
      // Tombol Hapus
      OutlinedButton(
        child: const Text("DELETE"),
        onPressed: () => confirmHapus(),
```

```
        ),
      ],
    );
  }

  void confirmHapus() {
    AlertDialog alertDialog = AlertDialog(
      content: const Text("Yakin ingin menghapus data ini?"),
      actions: [
        //tombol hapus
        OutlinedButton(
          child: const Text("Ya"),
          onPressed: () {
            ProdukBloc.deleteProduk(id: int.parse(widget.produk!.id!)).then(
                (value) => {
                      Navigator.of(context).push(MaterialPageRoute(
                          builder: (context) => const ProdukPage()))
                    }, onError: (error) {
              showDialog(
                  context: context,
                  builder: (BuildContext context) => const WarningDialog(
                        description: "Hapus gagal, silahkan coba lagi",
                      ));
            });
          },
        ),
        //tombol batal
        OutlinedButton(
          child: const Text("Batal"),
          onPressed: () => Navigator.pop(context),
        )
      ],
    );

    showDialog(builder: (context) => alertDialog, context: context);
  }
}
```

Tampil List Produk

Buat sebuah file dengan nama produk_page.dart pada folder ui dengan kode berikut

```
import 'package:flutter/material.dart';
```

```dart
import 'package:tokokita/model/produk.dart';
import 'package:tokokita/ui/produk_detail.dart';
import 'package:tokokita/ui/produk_form.dart';

class ProdukPage extends StatefulWidget {
  const ProdukPage({Key? key}) : super(key: key);

  @override
  _ProdukPageState createState() => _ProdukPageState();
}

class _ProdukPageState extends State<ProdukPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('List Produk'),
        actions: [
          Padding(
              padding: const EdgeInsets.only(right: 20.0),
              child: GestureDetector(
                child: const Icon(Icons.add, size: 26.0),
                onTap: () async {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (context) => ProdukForm()));
                },
              ))
        ],
      ),
      drawer: Drawer(
        child: ListView(
          children: [
            ListTile(
              title: const Text('Logout'),
              trailing: const Icon(Icons.logout),
              onTap: () async {
              },
            )
          ],
        ),
      ),
      body: ListView(
```

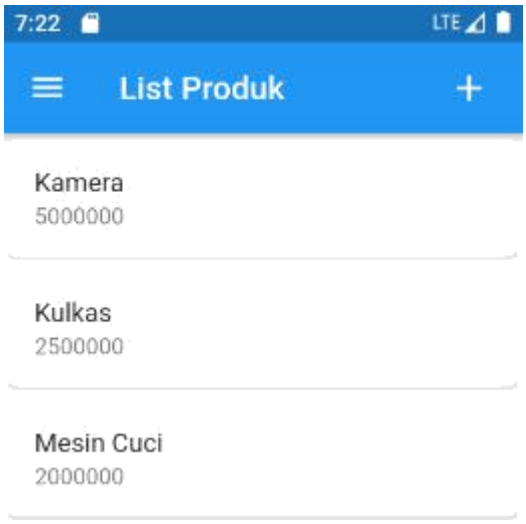
```
        children: [
          ItemProduk(
            produk: Produk(
              id: 1,
              kodeProduk: 'A001',
              namaProduk: 'Kamera',
              hargaProduk: 5000000,
            ),
          ),
          ItemProduk(
            produk: Produk(
              id: 2,
              kodeProduk: 'A002',
              namaProduk: 'Kulkas',
              hargaProduk: 2500000,
            ),
          ),
          ItemProduk(
            produk: Produk(
              id: 3,
              kodeProduk: 'A003',
              namaProduk: 'Mesin Cuci',
              hargaProduk: 2000000,
            ),
          ),
        ],
      ),
    );
  }
}
class ItemProduk extends StatelessWidget {
  final Produk produk;

  const ItemProduk({Key? key, required this.produk}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        Navigator.push(
            context,
            MaterialPageRoute(
```

```
                builder: (context) => ProdukDetail(
                      produk: produk,
                    )));
      },
      child: Card(
        child: ListTile(
          title: Text(produk.namaProduk!),
          subtitle: Text(produk.hargaProduk.toString()),
        ),
      ),
    );
  }
}
```

Untuk mencobanya modifikasi file main.dart menjadi seperti berikut

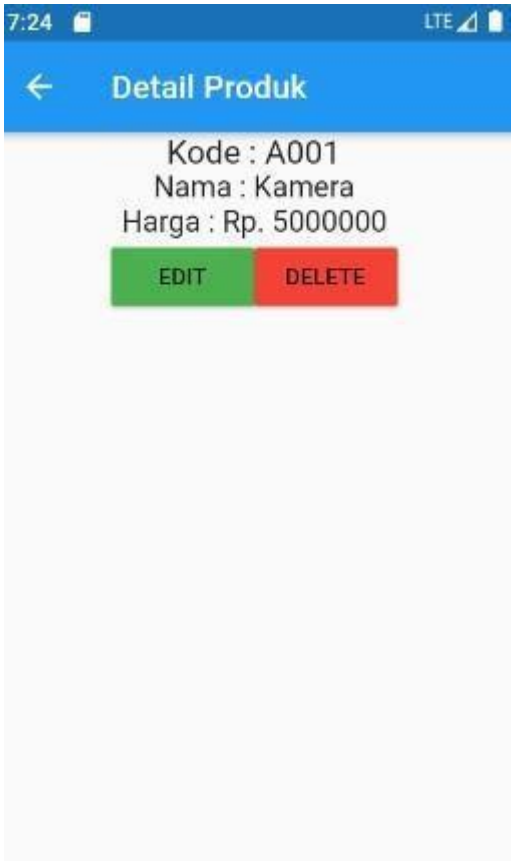
```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/produk_page.dart';
3.
4. void main() {
5.   runApp(const MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   const MyApp({Key? key}) : super(key: key);
10.
11.  @override
12.  Widget build(BuildContext context) {
13.    return const MaterialApp(
14.      title: 'Toko Kita',
15.      debugShowCheckedModeBanner: false,
16.      home: ProdukPage(),
17.    );
18.  }
19. }
```

Maka tampilannya akan menjadi seperti berikut



Pada saat tombol tambah diklik maka akan muncul form produk seperti berikut

Jika salah satu data produk diklik maka akan muncul detail produk



Ketika tombol EDIT diklik maka akan muncul form produk untuk mengubah data produk

Pada materi selanjutnya akan ada modifikasi pada tampil produk agar dapat menampilkan data dari Rest API serta modifikasi pada Form Produk sehingga dapat berfungsi untuk menyimpan ataupun mengubah data pada Rest API