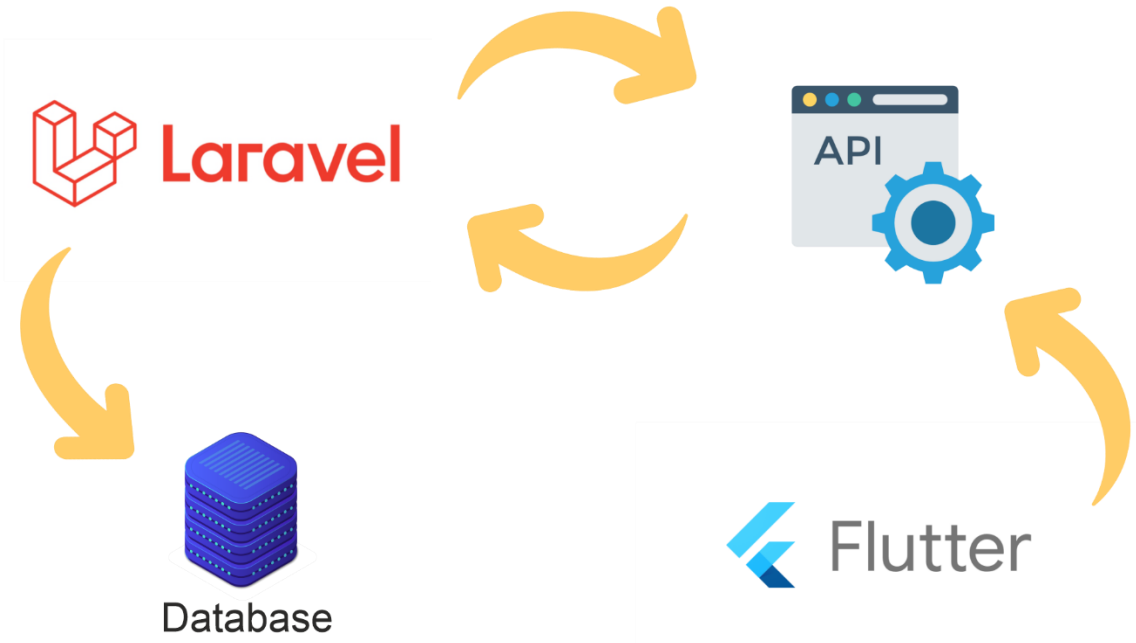
Modul CRUD Database Flutter
Menggunakan REST API Pada Laravel



Oleh :

~ Politeknik Negeri Malang ~

DAFTAR ISI

BAB I
PERSYARATAN

A. TUJUAN

Pada kesempatan kali ini, kita akan belajar cara membuat sebuah aplikasi flutter yang memiliki fungsi CRUD untuk mengatur database melalui API yang ada pada Laravel.

B. MANFAAT

Dengan mempelajari dasar bagaimana cara membuat aplikasi flutter dan penggunaan REST API akan membuat anda dapat mengembangkan kemampuan dalam membuat suatu aplikasi yang fungsional.

C. SOFTWARE

Terdapat beberapa perangkat lunak yang dibutuhkan dalam pembuatan proyek ini nantinya, diantaranya adalah :

No	Aplikasi	Link Download
1	XAMPP	https://www.apachefriends.org/download.html
2	Composer	https://getcomposer.org/download/
3	Node JS	https://nodejs.org/en/download
4	Postman	https://www.postman.com/downloads/
5	Visual Studio Code	https://code.visualstudio.com/download
6	Android Studio	https://developer.android.com/studio
7	Flutter	https://docs.flutter.dev/get-started/install/windows

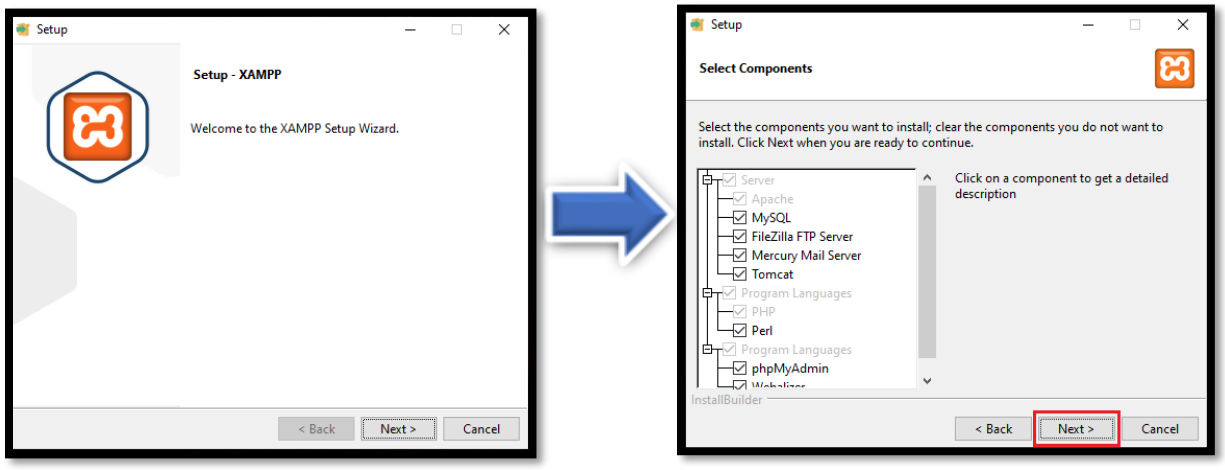
D. PERTANYAAN UMUM

NO	PERMASALAHAN
1	Mengapa harus menggunakan API ?
	Dengan menggunakan API akan membuat keamanan database tetap terjaga, karena kita hanya dapat menambah maupun mengubah isi dari database berdasarkan lingkup yang telah kita buat dalam fungsi API tersebut.
2	Mengapa terjadi error ketika saya menjalankan perintah untuk membuat proyek laravel ?
	Pastikan pemasangan composer telah berhasil dan coba buka ulang cmd anda.
3	Mengapa pengujian REST API saya di POSTMAN selalu gagal ?
	Pastikan laravel telah berjalan dan koreksi kembali code program dan url yang dituju.

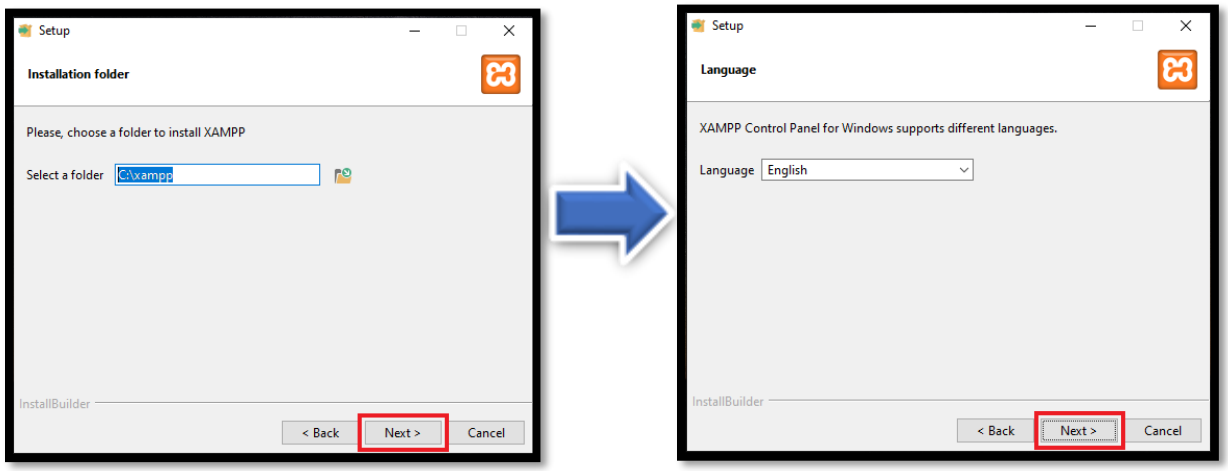
BAB II
PEMASANGAN APLIKASI

A. Instalasi XAMPP

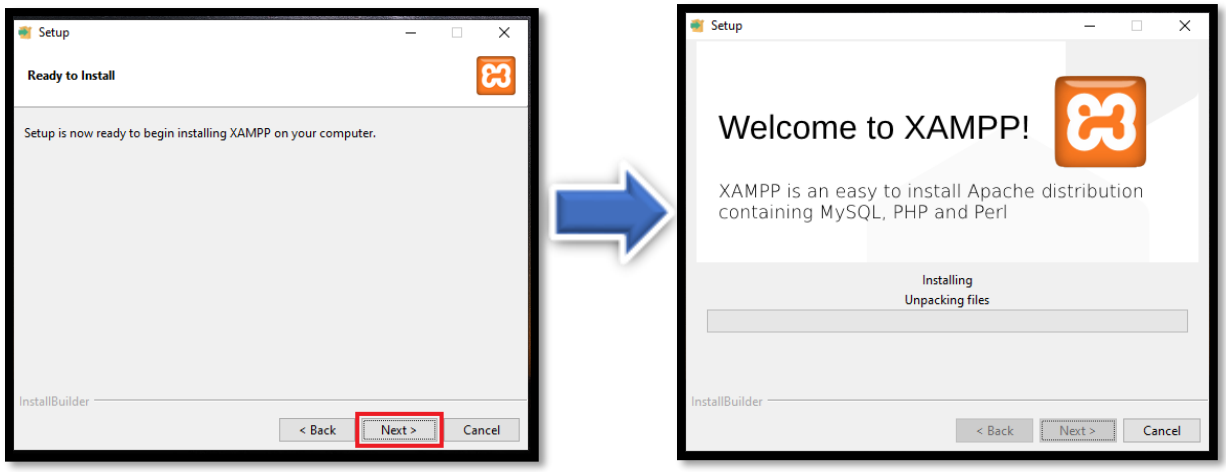
1. Buka file XAMPP yang telah didownload sebelumnya dengan cara klik 2x pada file xampp.exe kemudian klik “Next” dan pada bagian Select Component biarkan secara default dan langsung klik “Next”.



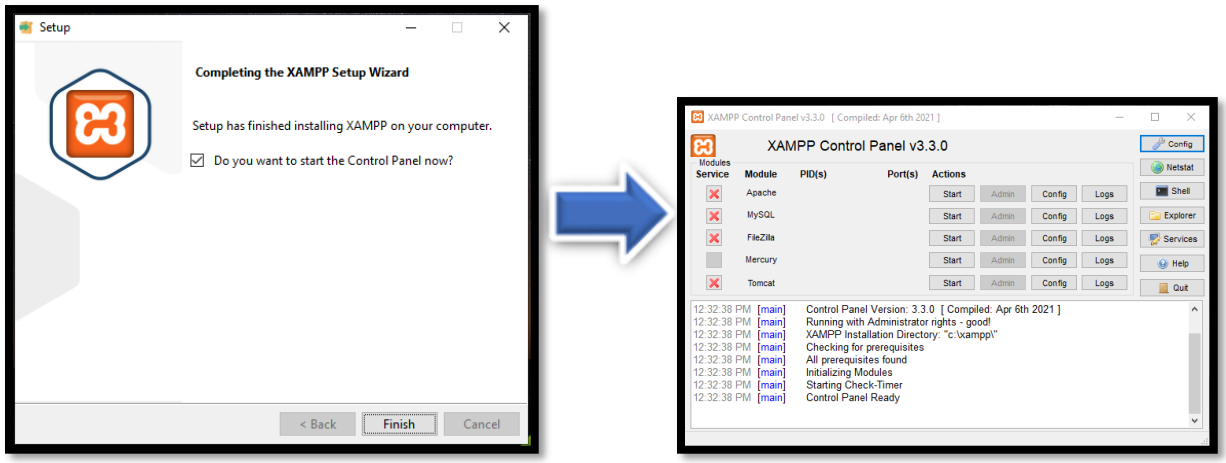
2. Biarkan lokasi folder tetap dengan pengaturan awal dan klik “Next”, pilih Bahasa yang ingin digunakan kemudian klik “Next”.



3. Klik “Next” dan tunggu hingga proses pemasangan selesai.

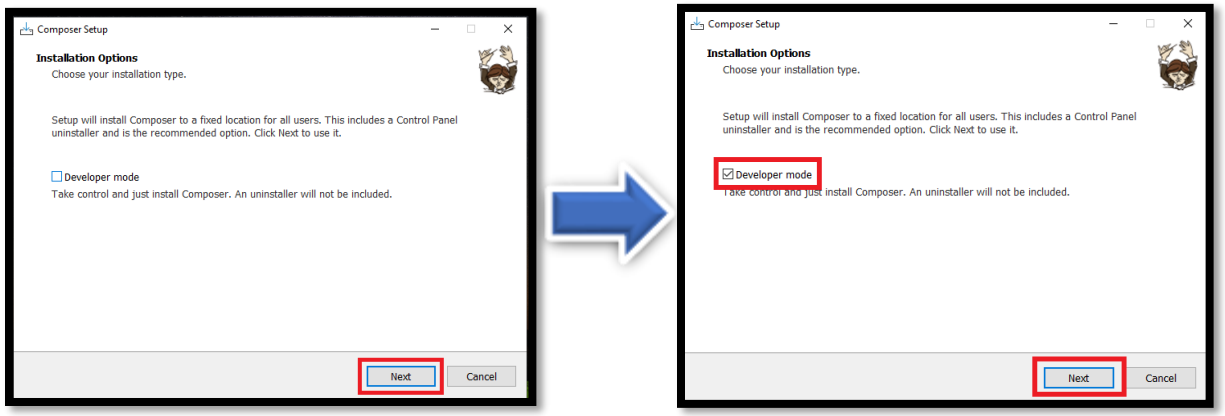


4. Setelah proses instalasi selesai klik “Finish” dan jendela panel xampp akan muncul dilayar anda, untuk dapat menjalankan project ini klik “Start” pada module Apache dan MySQL.

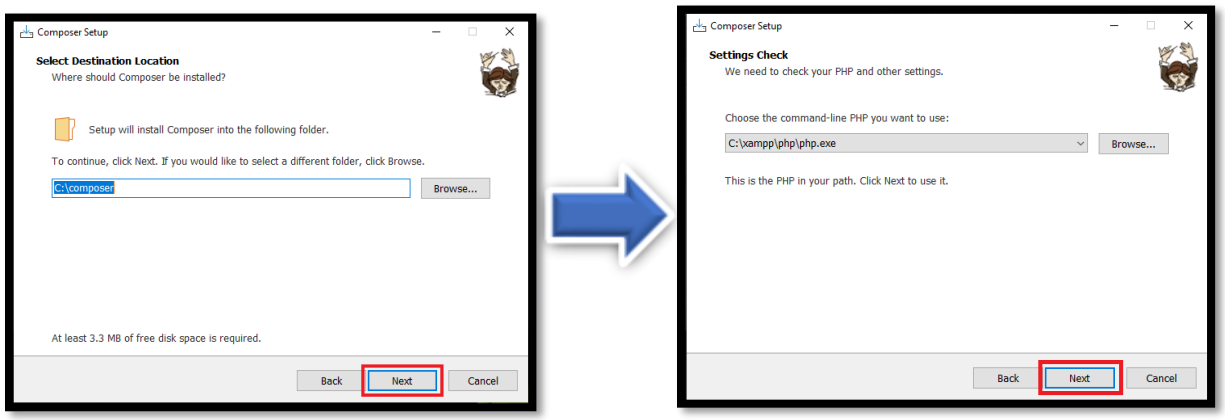


B. Instalasi Composer

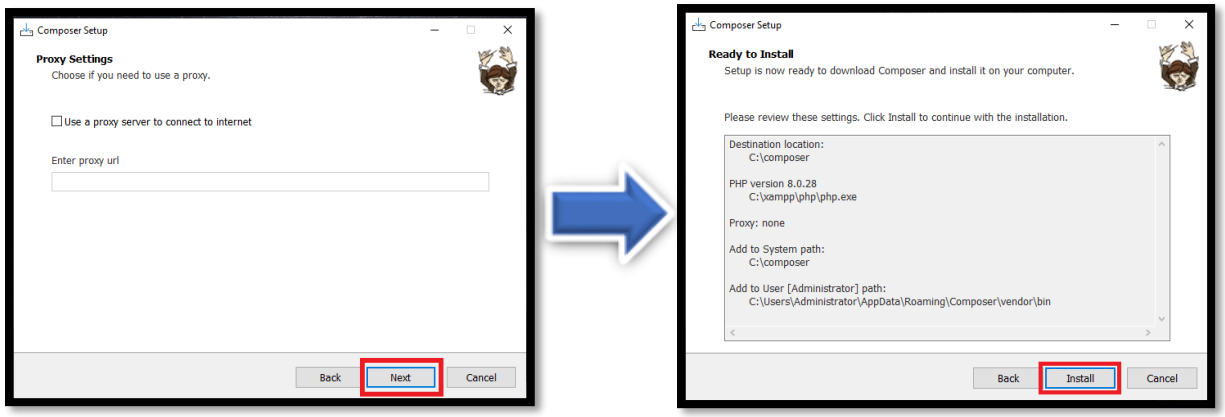
1. Buka file Composer yang telah didownload sebelumnya dengan cara klik 2x pada file composer.exe kemudian centang pada bagian “Developer Mode” dan klik “Next”.



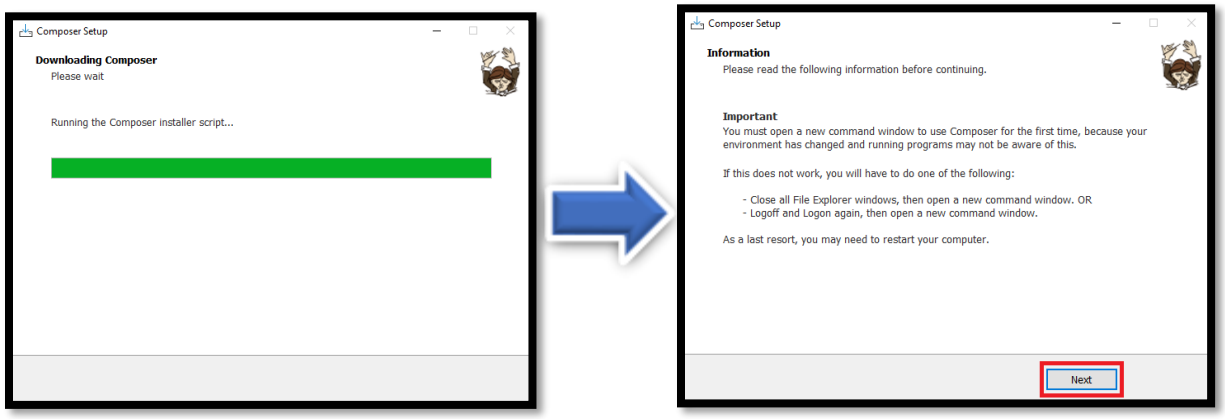
2. Biarkan lokasi folder tetap dengan pengaturan awal dan klik “Next”, lalu pada bagian Setting Check apabila file php.exe sudah terdeteksi langsung klik “Next” apabila belum silahkan klik “Browse” untuk mencari lokasi file php.exe anda.



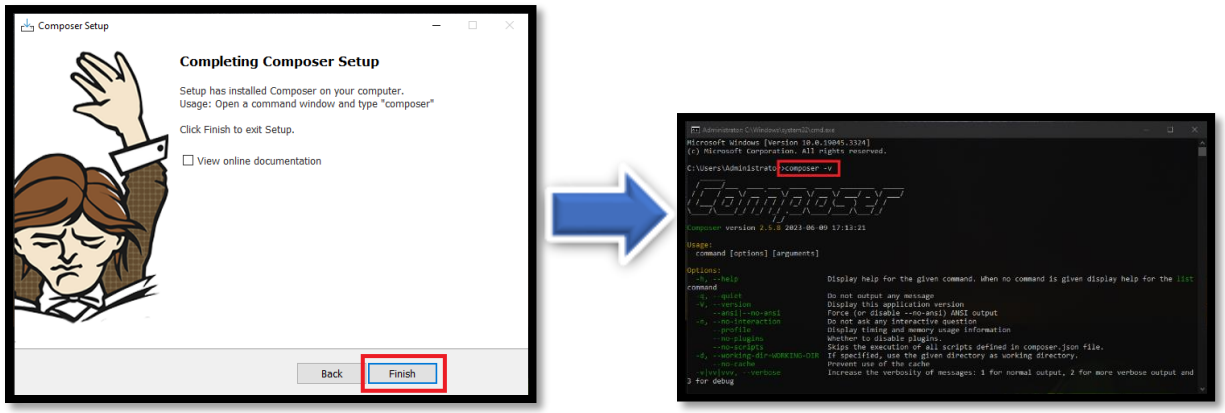
3. Klik “Next” kemudian klik “Install”.



4. Tunggu proses pemasangan selesai kemudian klik “Next”.



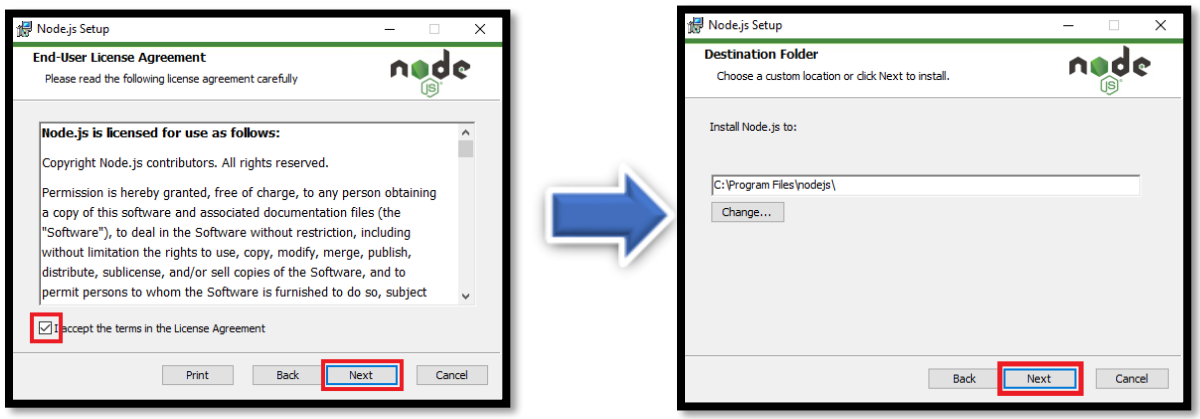
5. Klik “Finish” dan buka Terminal kemudian ketikkan perintah “composer -v” tanpa tanda kutip dan amati output yang dihasilkan, jika output yang dikeluarkan merupakan rincian versi dari composer maka pemasangan composer telah berhasil dilakukan.



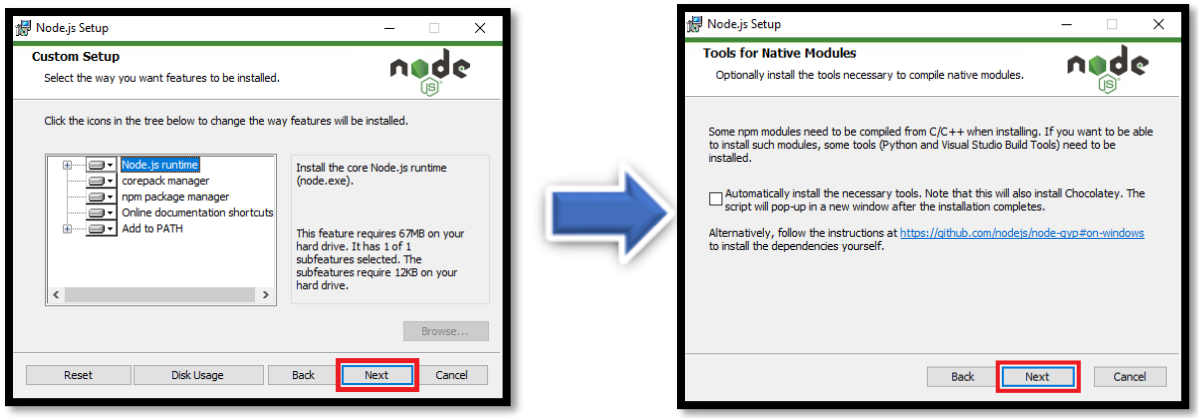
C. Instalasi Node JS

1. Buka file Node JS yang telah didownload sebelumnya dengan cara klik 2x pada file node.exe, apabila terdapat peringatan keamanan tetap jalankan dengan mengklik tombol "Run" kemudian klik "Next".

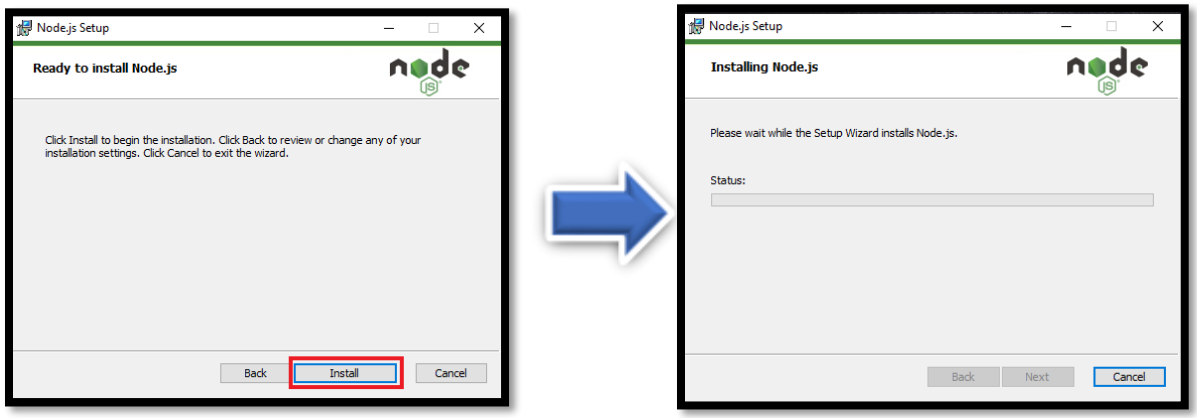
2. Centang box "I Accept …" dan klik "Next", kemudian pada bagian Destination Folder biarkan menggunakan lokasi yang telah disediakan lalu klik "Next".



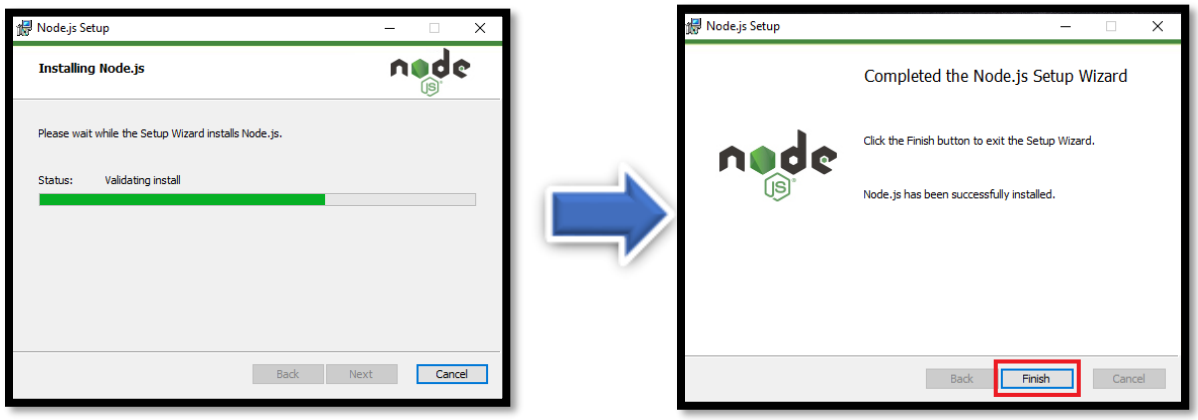
3. Gunakan pengaturan awal dan jangan diubah kemudian klik “Next” 2x.



4. Klik “Install” dan tunggu hingga proses pemasangan selesai.

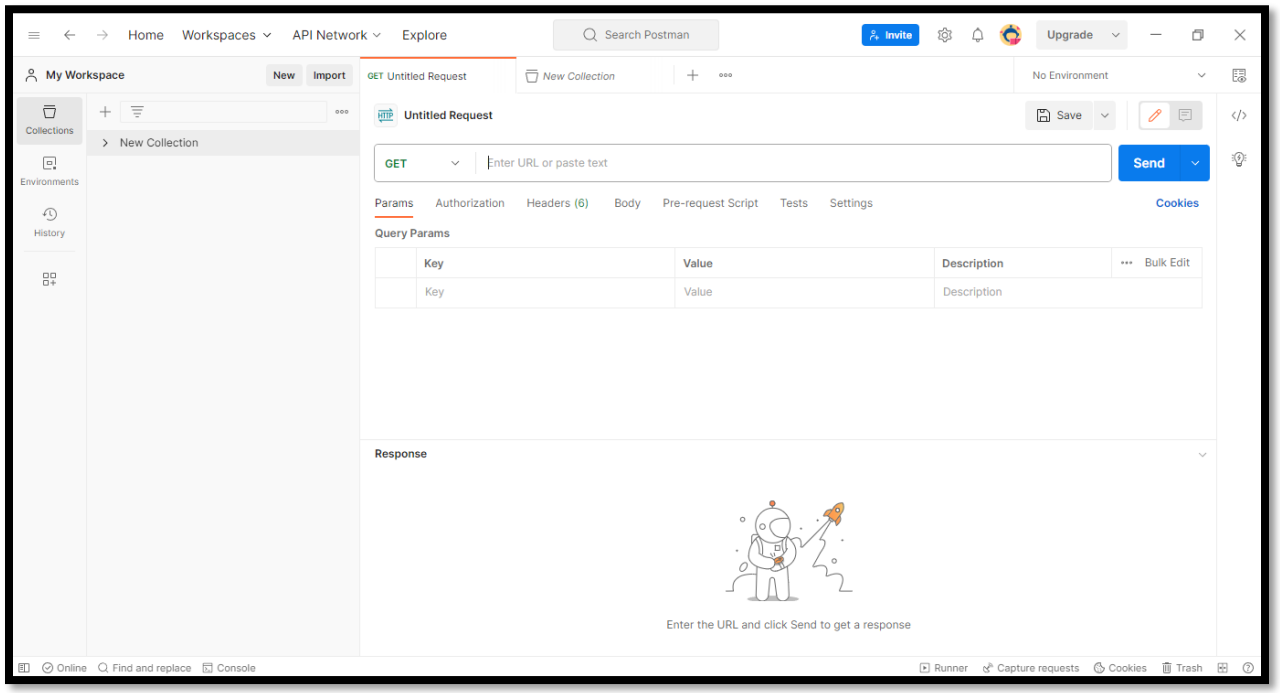


5. Setelah proses pemasangan selesai, klik “Finish”.



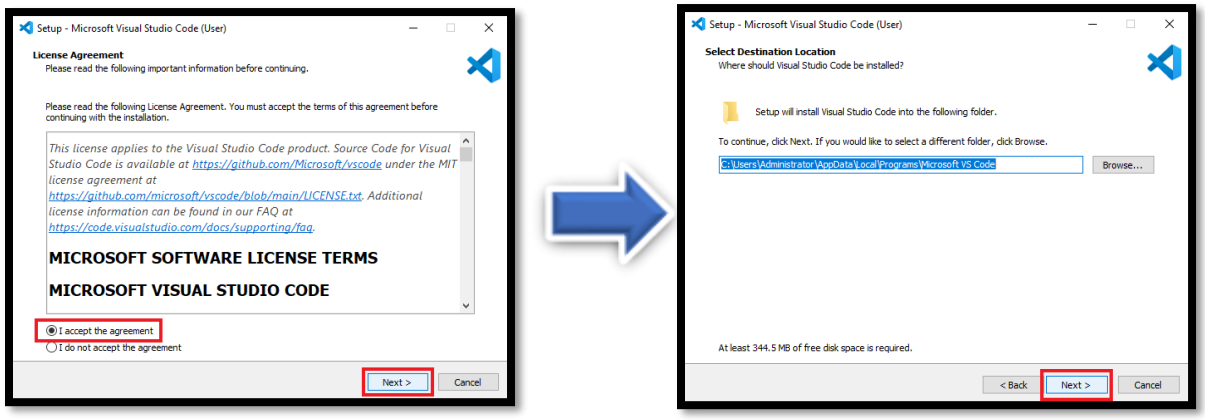
D. Instalasi Postman

1. Buka file Postman yang telah didownload sebelumnya dengan cara klik 2x pada file postman.exe, setelah itu tunggu beberapa saat hingga proses pemasangan postman selesai, apabila aplikasi telah menampilkan tampilan jendela seperti gambar dibawah ini berarti aplikasi telah siap digunakan.

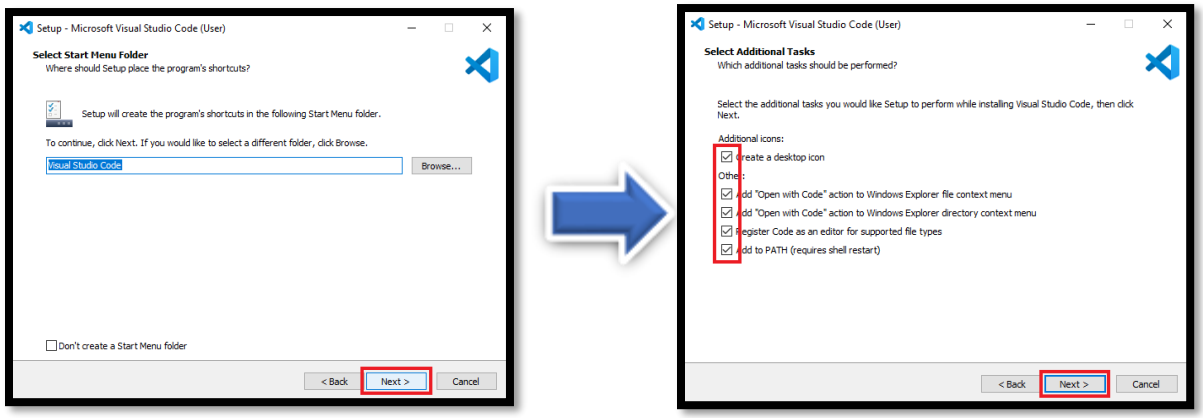


E. Instalasi Visual Studio Code

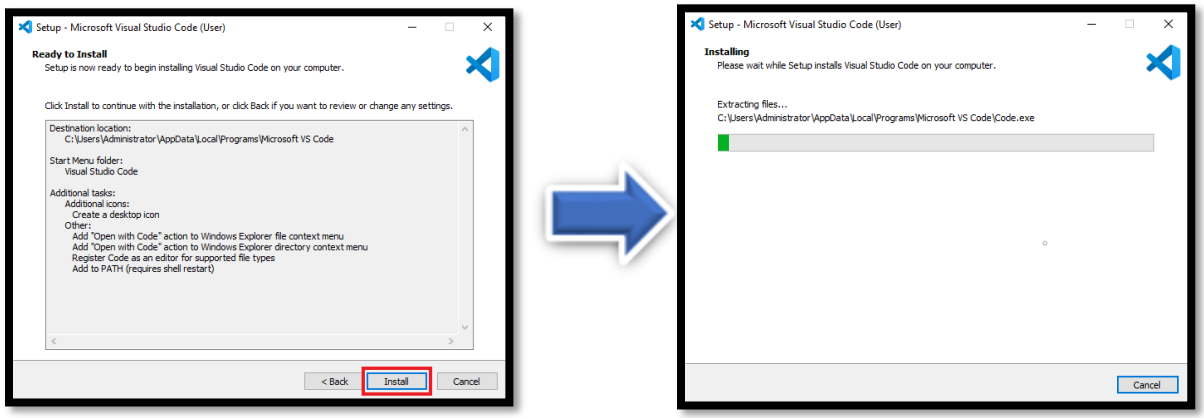
1. Buka file installer yang telah didownload sebelumnya, kemudian pilih “I Accept …” dan klik “Next”. Lalu langsung klik “Next” pada bagian Select Destination Location.



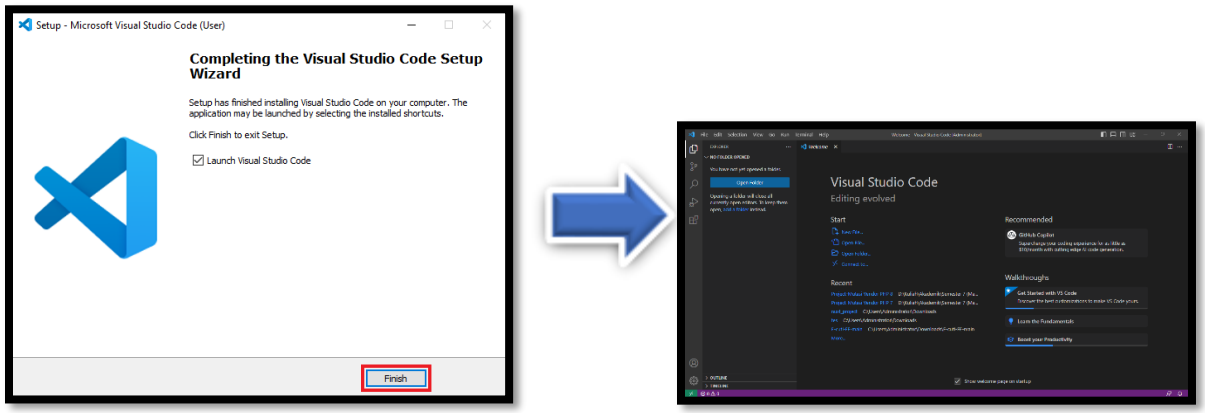
2. Klik “Next” kemudian centang semua opsi pilihan dan klik “Next” lagi.



3. Klik “Install” dan tunggu hingga proses pemasangan selesai.

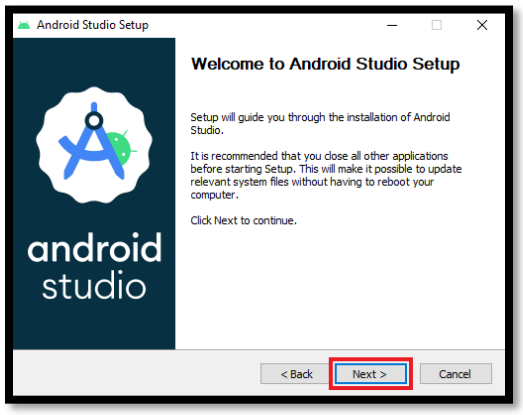


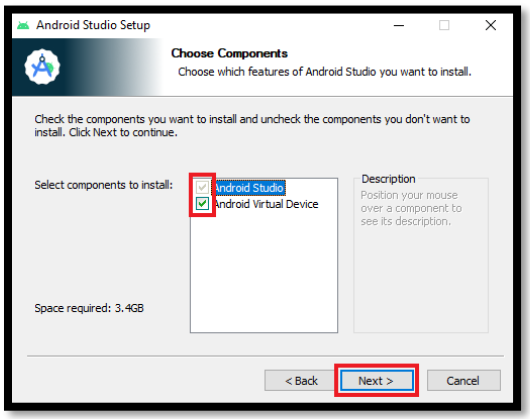
4. Setelah proses pemasangan selesai, klik “Finish” dan akan terbuka jendela baru yang merupakan tampilan dari visual studio code.



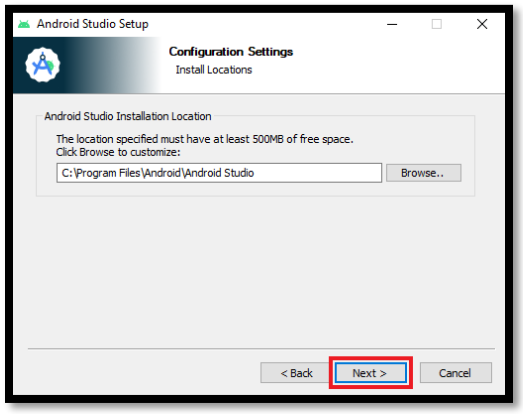
F. Instalasi Android Studio

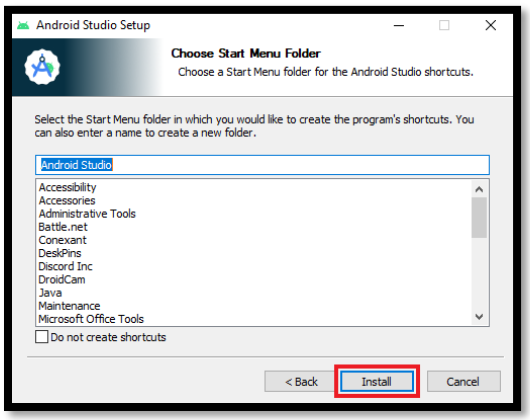
1. Buka file Android Studio yang telah didownload sebelumnya dengan cara klik 2x pada file android-studio.exe, kemudian klik “Next” dan centang pada bagian “Android Virtual Device” lalu klik “Next”.



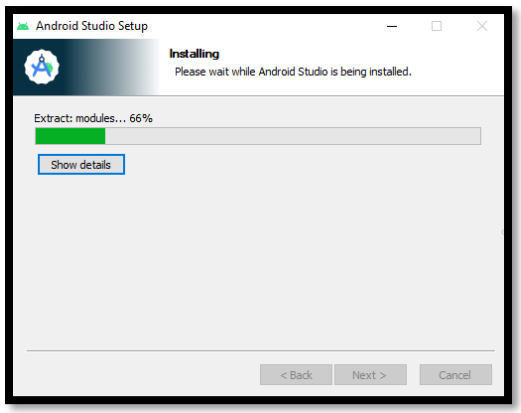


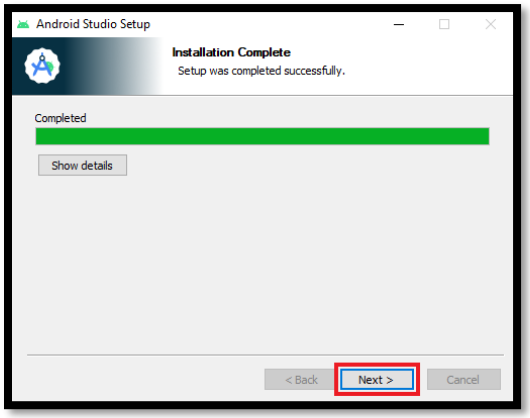
2. Pilih lokasi untuk tempat menginstall kemudian klik “Next” dan klik “Install”.



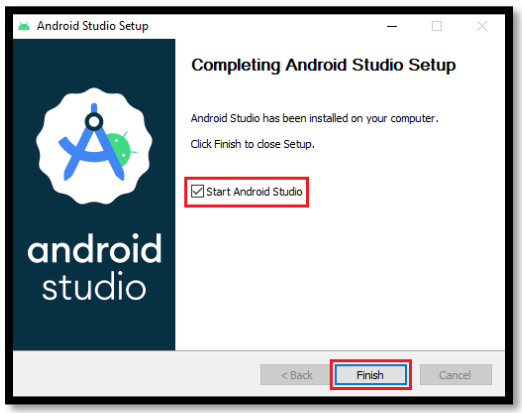


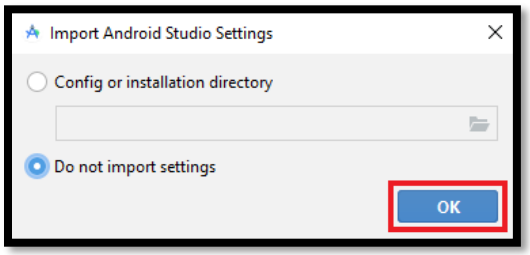
3. Tunggu hingga proses pemasangan selesai, kemudian klik “Next”.



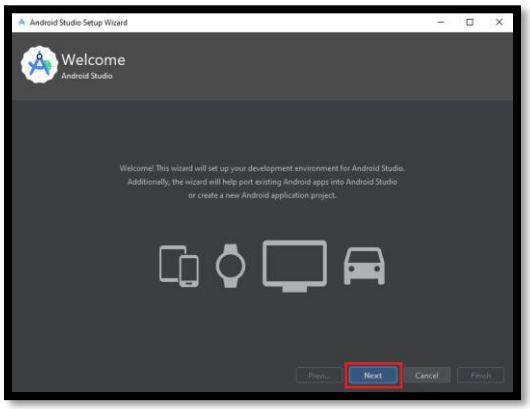


4. Centang pada bagian “Start Android Studio” dan klik “Finish”.

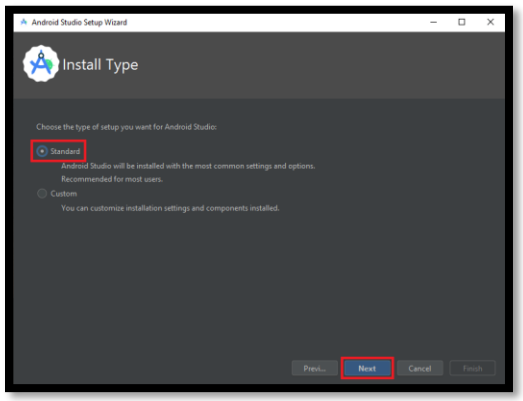


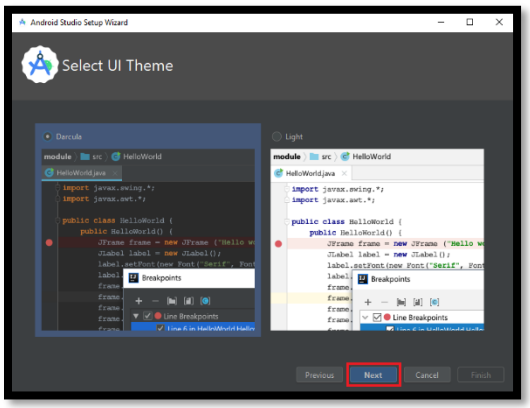


5. Setelah proses install berhasil, aplikasi android studio akan membuka jendela baru untuk pengaturan awal, klik “Next” pada jendela yang telah muncul.

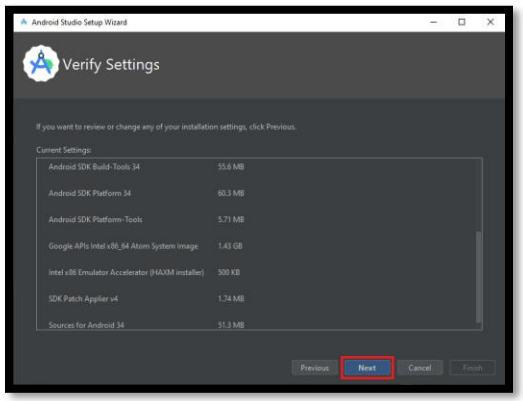


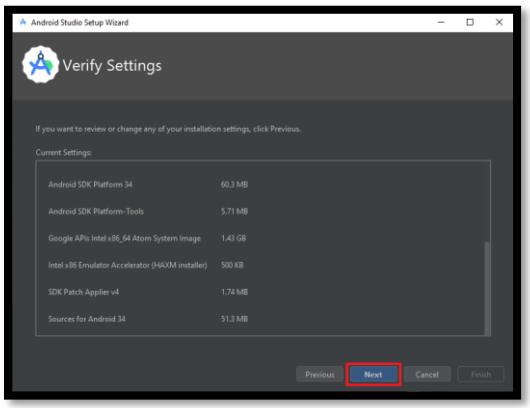
6. Pilih tipe “Standard” dan klik “Next”, kemudian pilih tema yang anda inginkan kemudian klik “Next”.



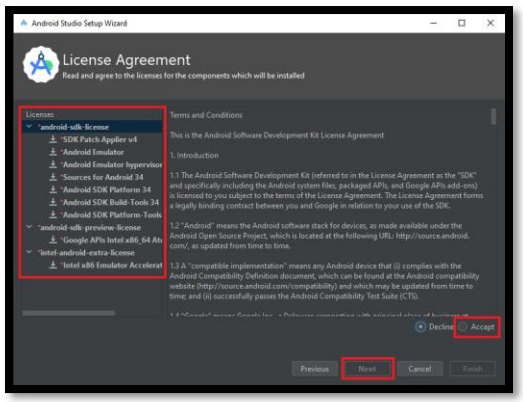


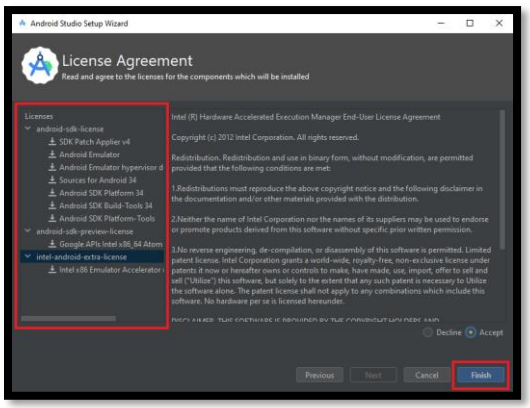
7. Terdapat rincian dari paket-paket yang akan terdownload lagi secara otomatis nantinya, klik “Next”.



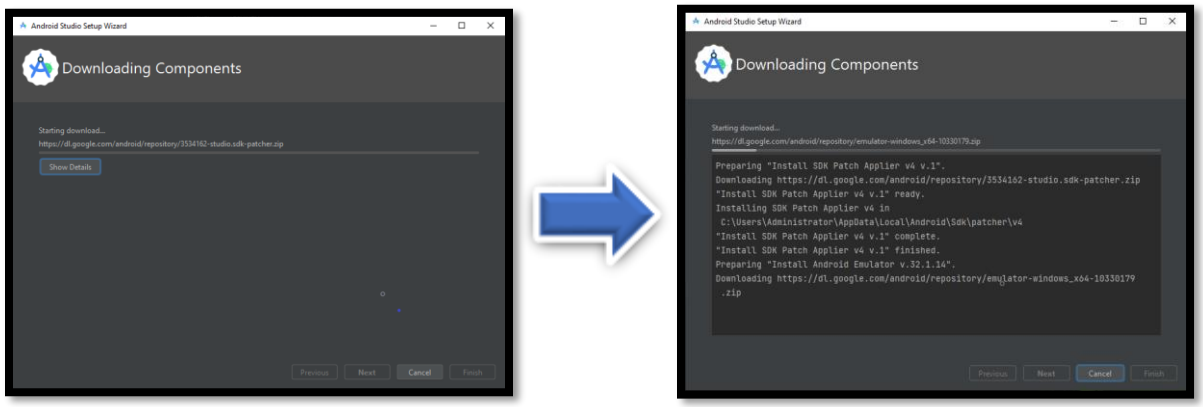


8. Pada setiap lisensi paket disebelah kiri jendela, pilih “Accept” kemudian klik “Finish”.

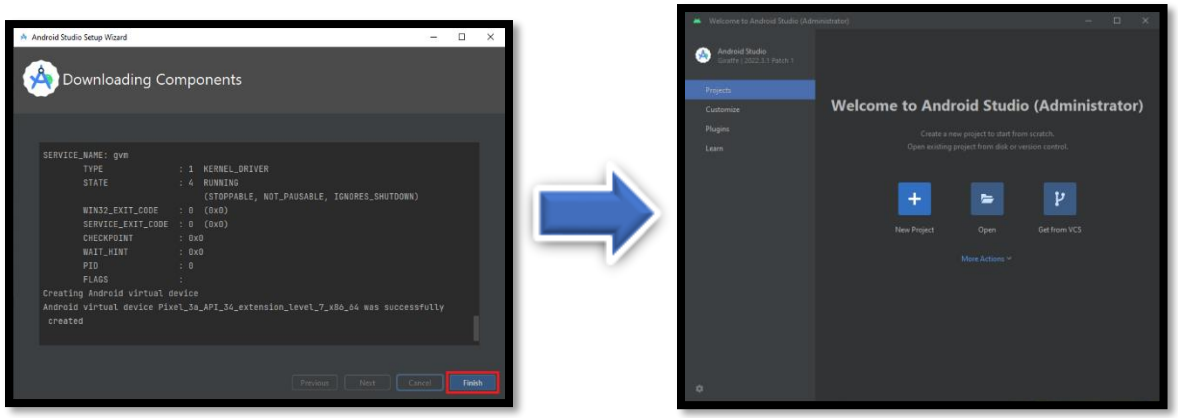




9. Tunggu hingga proses download selesai, komponen-komponen yang perlu didownload cukup besar jadi pastikan internet anda stabil dengan device yang tetap menyala.

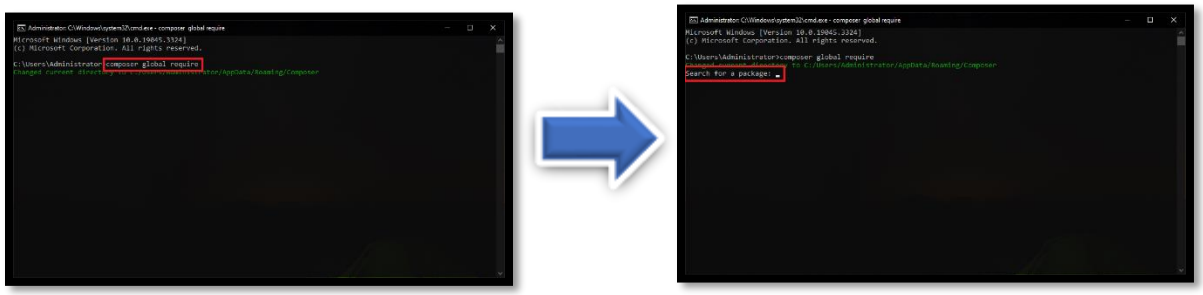


10. Setelah proses download selesai klik “Finish” dan android studio telah siap untuk digunakan.

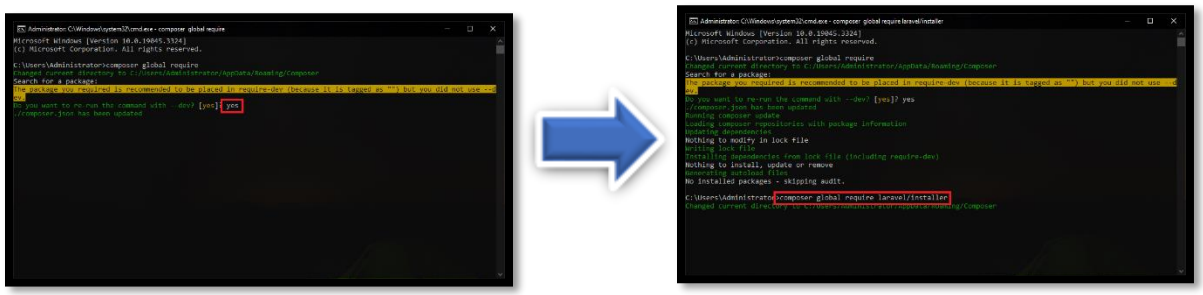


G. Instalasi Laravel

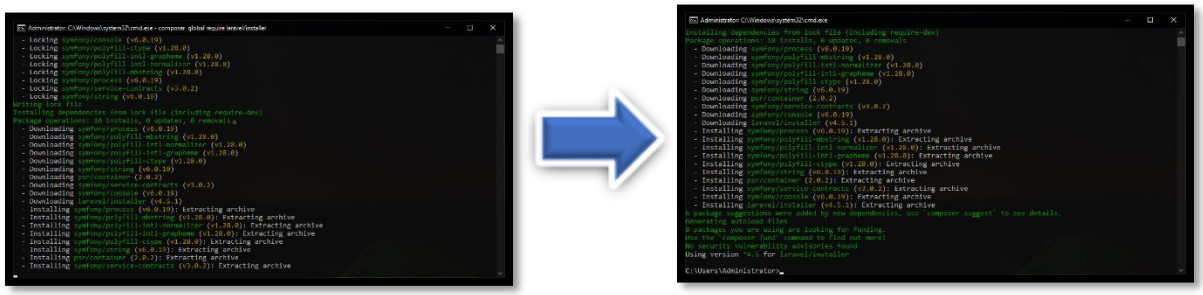
1. Buka Terminal atau Command Prompt di perangkat anda dan ketikkan perintah “composer global require” tanpa tanda kutip kemudian klik enter, apabila muncul baris yang berisi “Search for a package…” langsung saja klik enter lagi.



2. Ketika terdapat pertanyaan untuk menjalankan perintah ketik “yes” lalu enter. Setelah proses sebelumnya selesai, ketik “composer global require laravel/installer” tanpa tanda kutip lalu enter.

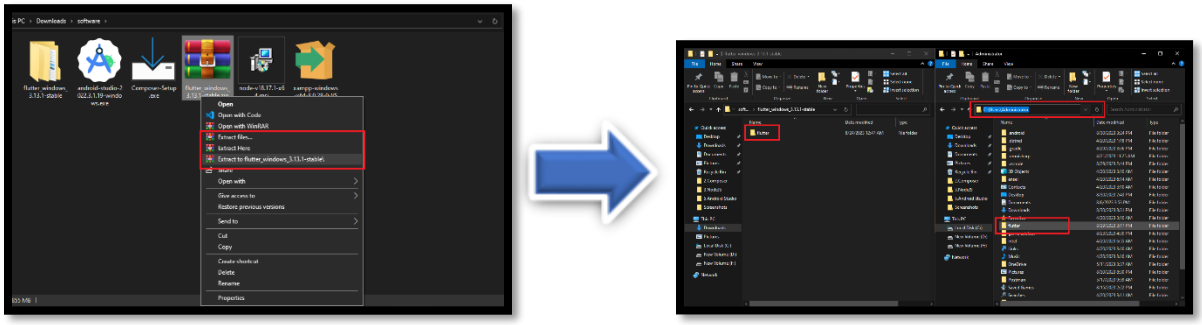


3. Pastikan koneksi internet anda lancer dan stabil, tunggu hingga proses pemasangan selesai.

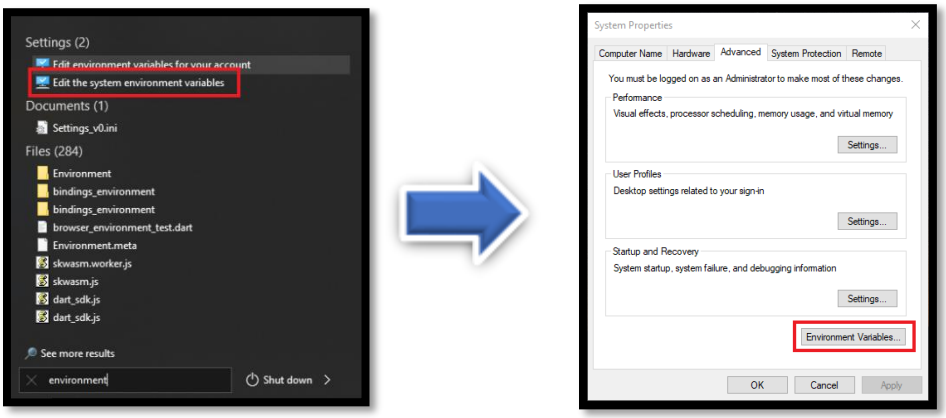


H. Instalasi Flutter

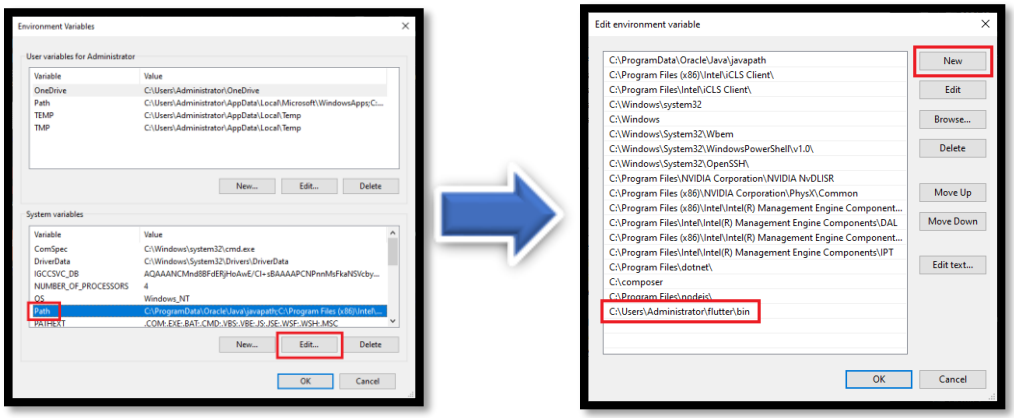
1. Klik kanan pada file flutter yang telah didownload sebelumnya dan extrak file tersebut terlebih dahulu. Kemudian pindahkan folder bernama “flutter” hasil dari extrak tadi ke direktori “C:\Users\(nama_user)”.



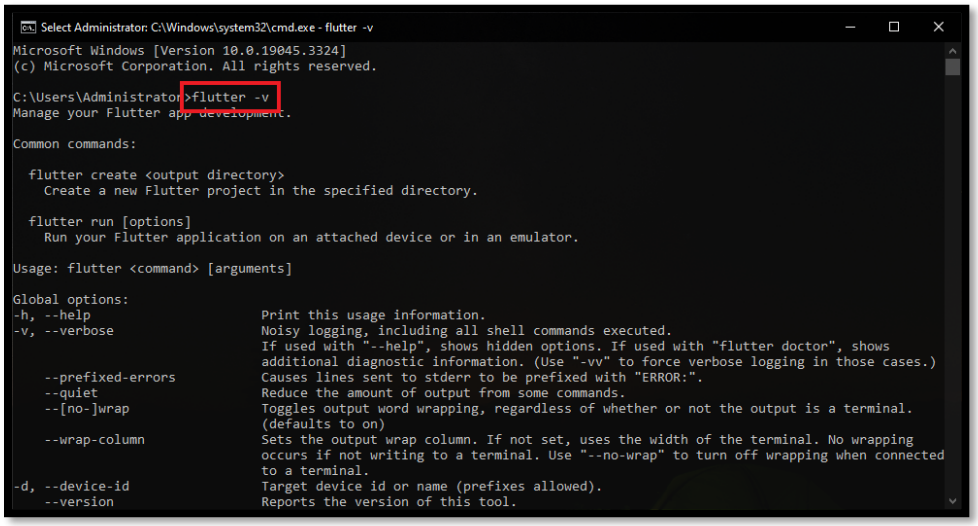
2. Setelah itu cari pengaturan “environment” pada fitur pencarian di perangkat anda. Lalu klik “Environtment Variables…”.



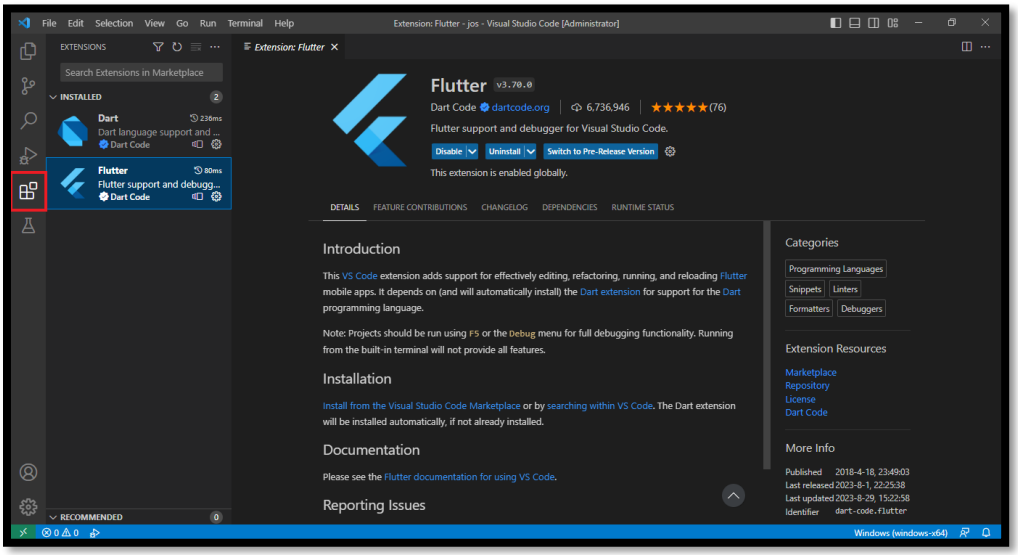
3. Pilih “Path” atau “PATH” pada bagian System Variables dan klik “Edit”, kemudian pilih “New” dan isikan lokasi dari folder flutter/bin. (example : C:\Users\Administrator\flutter\bin)



4. Buka terminal atau command prompt dan ketikkan perintah “flutter -v” tanpa tanda kutip dan jika hasil output sama seperti gambar dibawah ini maka flutter telah berhasil dipasang.



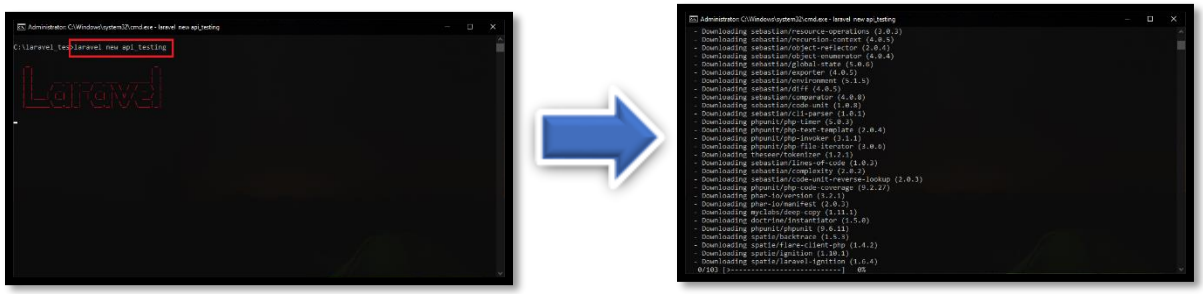
5. Buka aplikasi Visual Studio Code dan pilih menu “Extensions” pada sebelah kiri jendela aplikasi, kemudian cari dan install ekstensi Flutter dan Dart seperti gambar dibawah ini. (note: untuk ekstensi Dart seharusnya sudah terinclude dalam ekstensi Flutter)



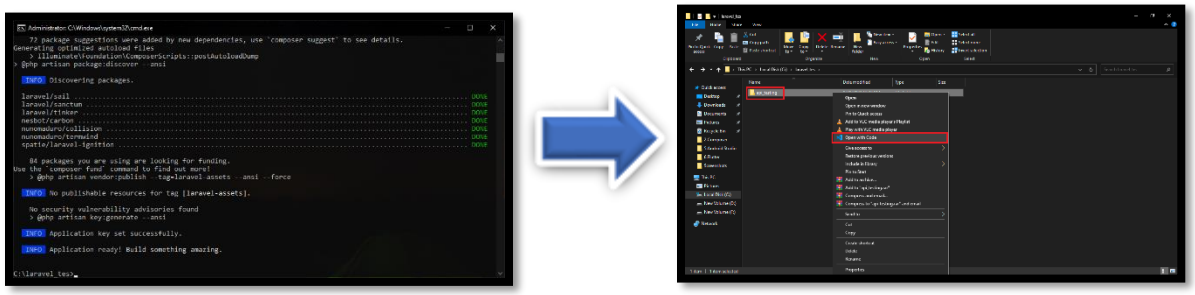
BAB III
PEMBUATAN API LARAVEL DAN FLUTTER

A. Pembuatan Framework Laravel

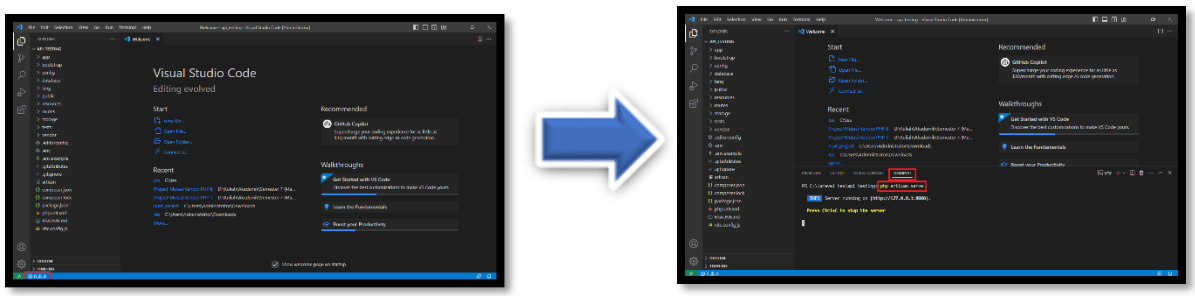
1. Buka Terminal atau Command Prompt di perangkat anda dan ketikkan perintah “laravel new api_testing” kemudian klik enter dan tunggu hingga proses pembuatan proyek laravel selesai.



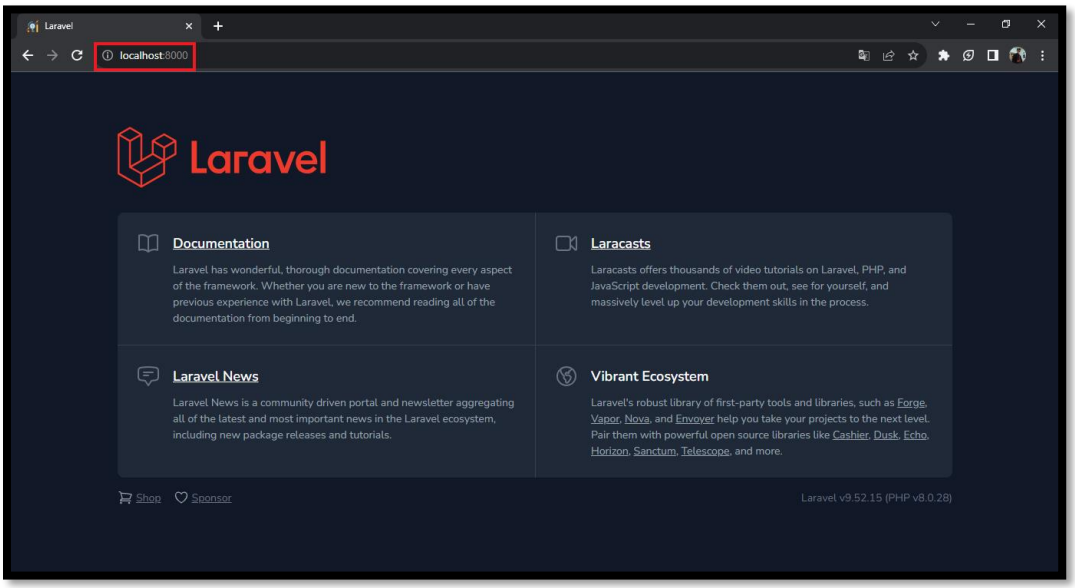
2. Setelah proses loading selesai, buka Windows Explore di perangkat anda dan cari folder dengan nama “api_testing”, kemudian klik kanan pada folder tersebut dan pilih “Open With Code” untuk membuka proyek tersebut dengan aplikasi Visual Studio Code.



3. Kemudian klik bagian kiri bawah visual studio code seperti pada gambar dibawah untuk membuka terminal pada Visual Studio Code, atau anda dapat memilih menu “Terminal pada bagian atas jendela Visual Studio Code dan pilih sub menu “New Terminal”. Kemudian ketikkan perintah “php artisan serve” lalu klik enter.

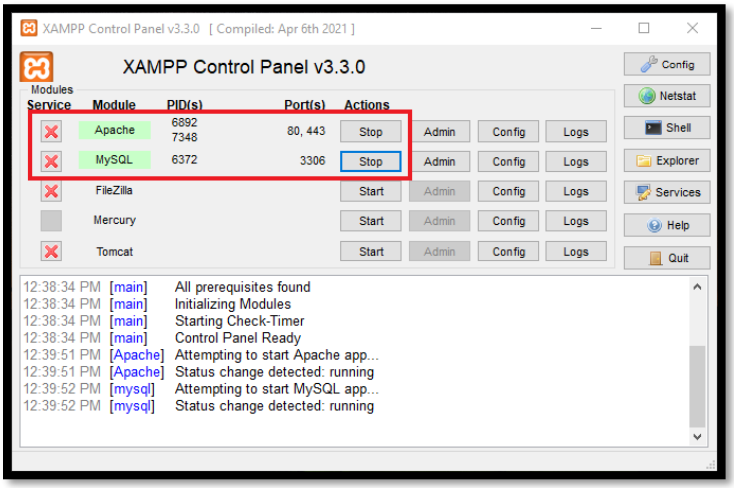


4. Buka browser di perangkat anda dan tuliskan url “localhost:8000” tanpa tanda kutip di browser anda. Apabila tampilan yang dihasilkan sama seperti gambar dibawah ini maka pembuatan proyek laravel baru telah berhasil.

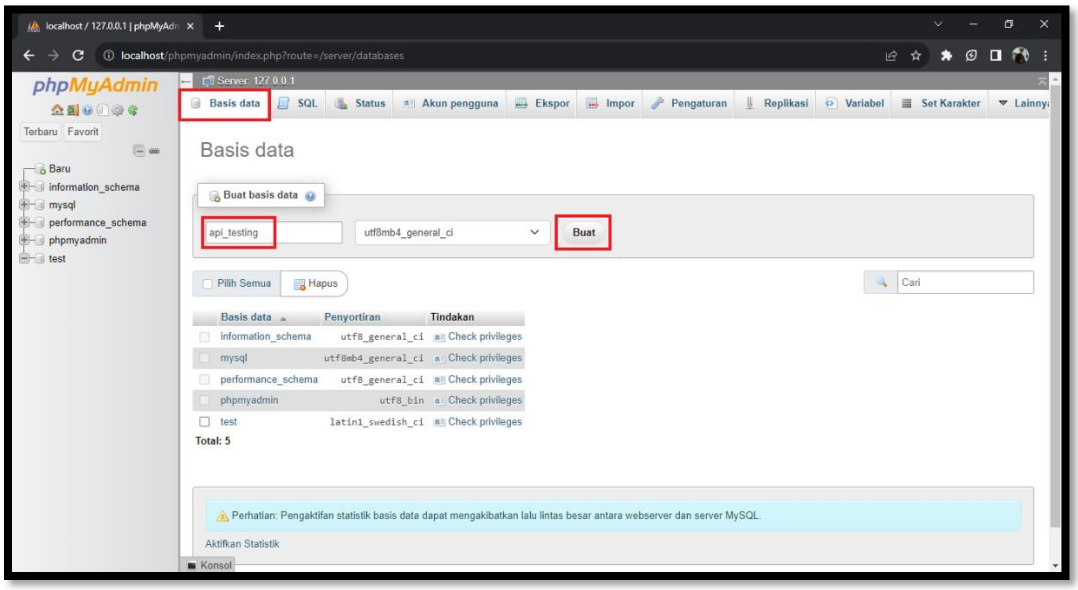


B. Pembuatan REST API di Laravel

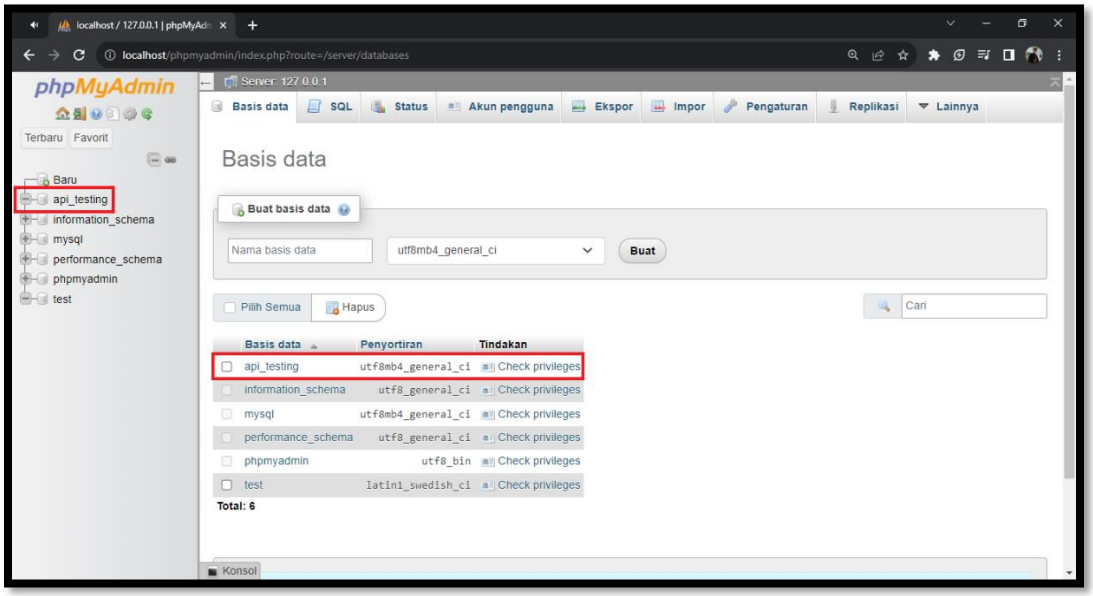
1. Pertama-tama buka aplikasi XAMPP di perangkat anda dan pastikan bahwa module “Apache” dan “MySQL” telah aktif.



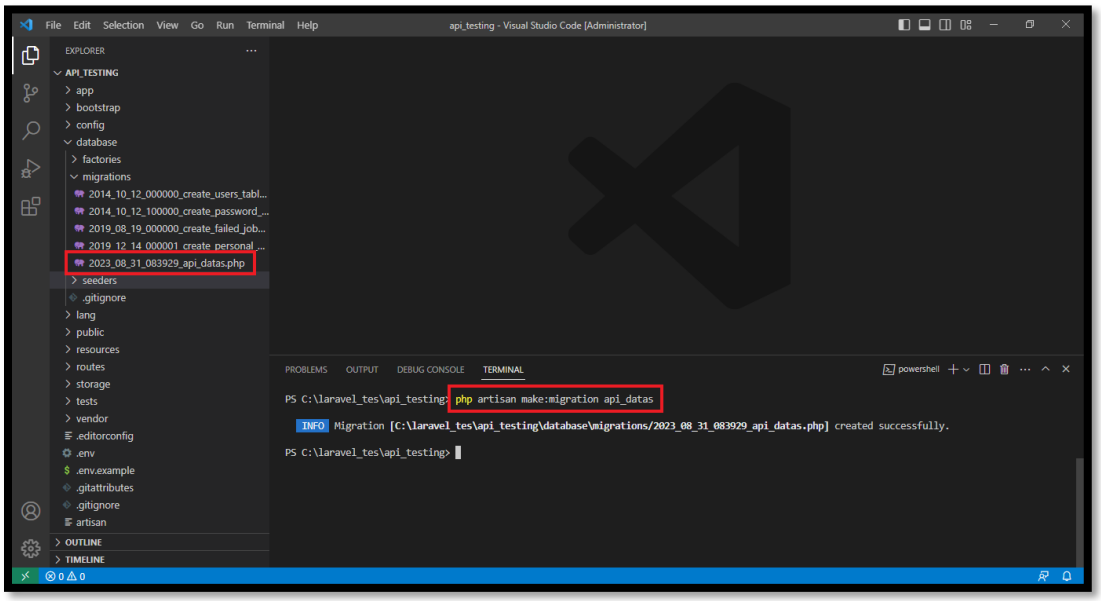
2. Kemudian buka browser anda dan akses Alamat “localhost/phpmyadmin” tanpa tanda kutip. Lalu pilih menu “Basis Data” dan buat database baru dengan nama “api_testing” tanpa tanda kutip.



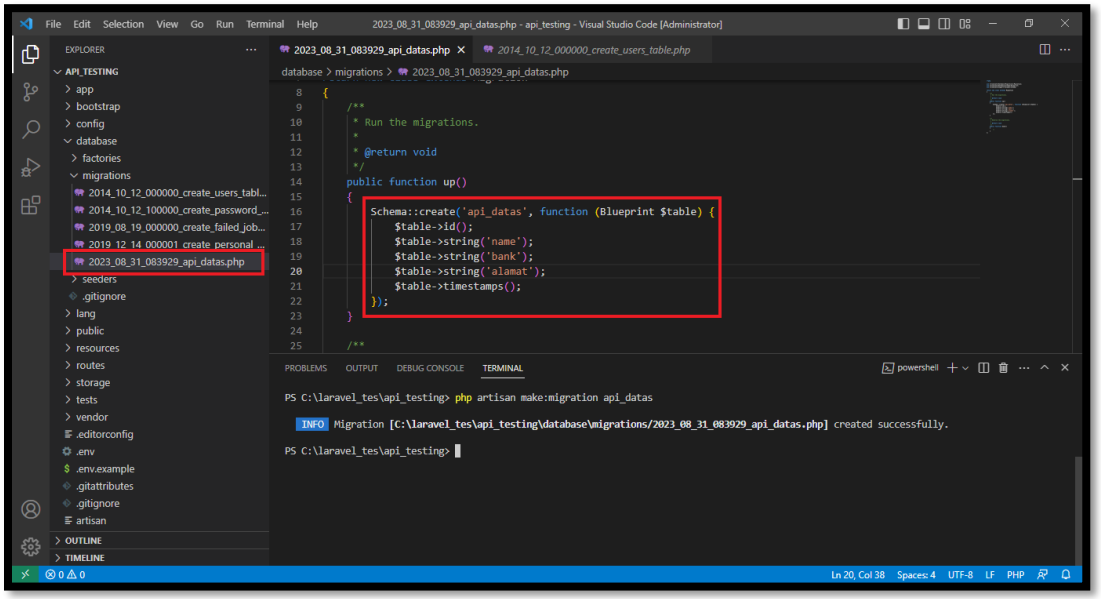
3. Setelah pembuatan database berhasil maka database dengan nama “api_testing” akan terlihat pada browser anda.



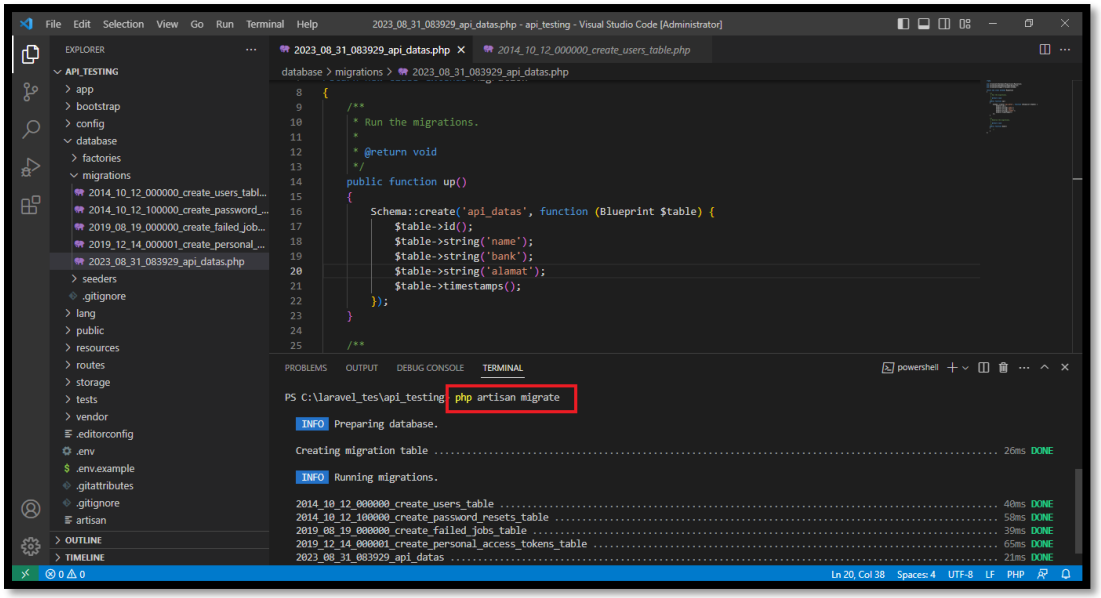
4. Kembali ke terminal yang ada di Visual Studio Code dan ketikka perintah "php artisan make:migration api_datas" tanpa tanda kutip dan klik enter untuk membuat file migration yang berfungsi untuk mengatur tabel dalam database.



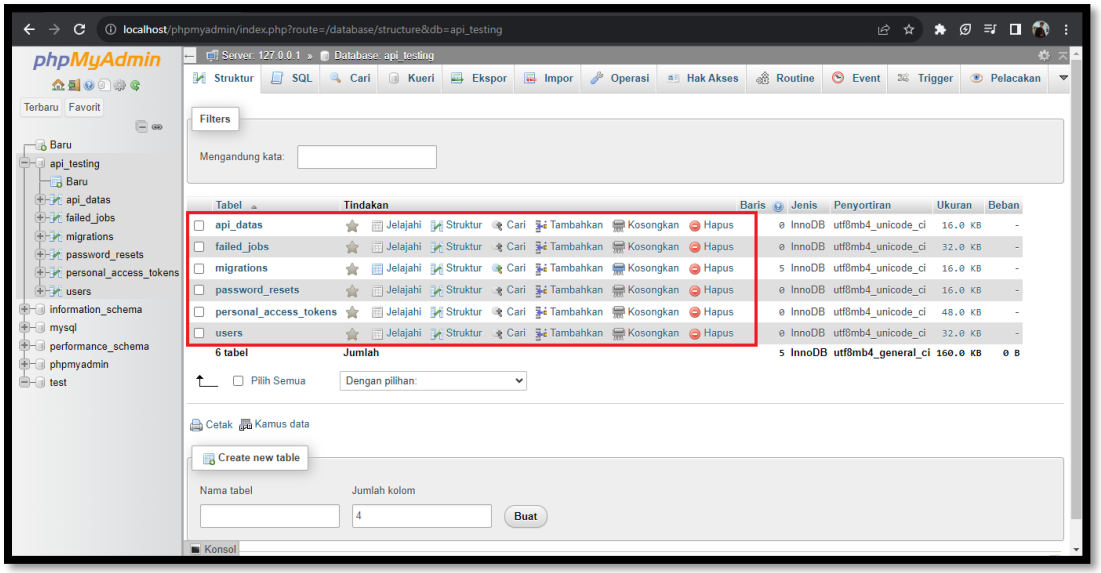
5. Buka file migration yang baru saja dibuat dengan nama **api_datas** yang berada pada folder "**database/migrations/**" pada proyek laravel anda. Kemudian ubah kode program pada fungsi up() menjadi seperti pada gambar dibawah ini.



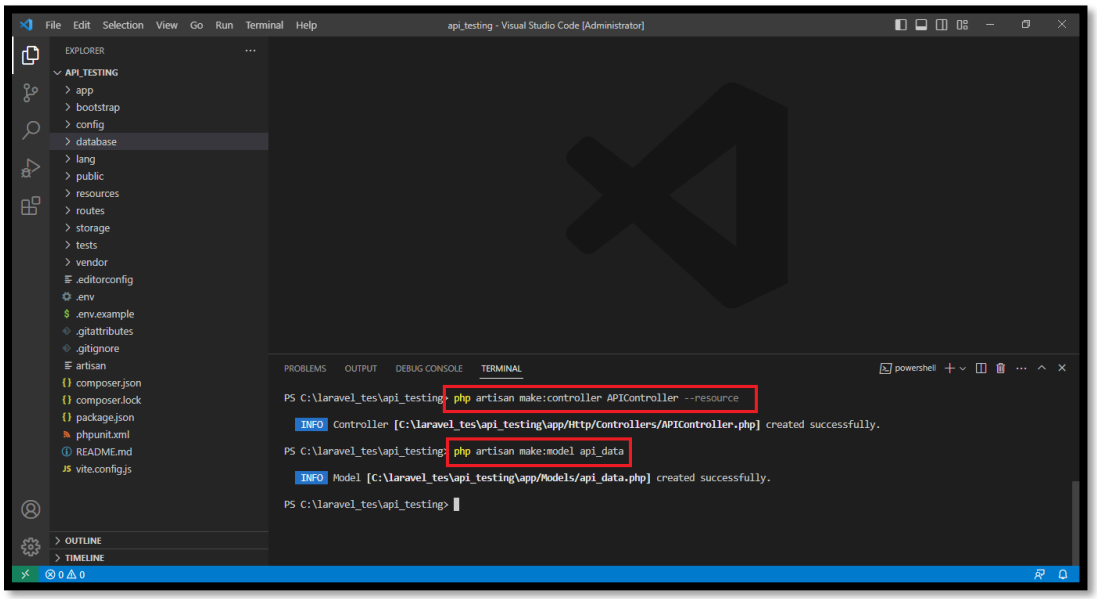
6. Ketik “php artisan migrate” pada terminal di Visual Studio Code dan klik enter.



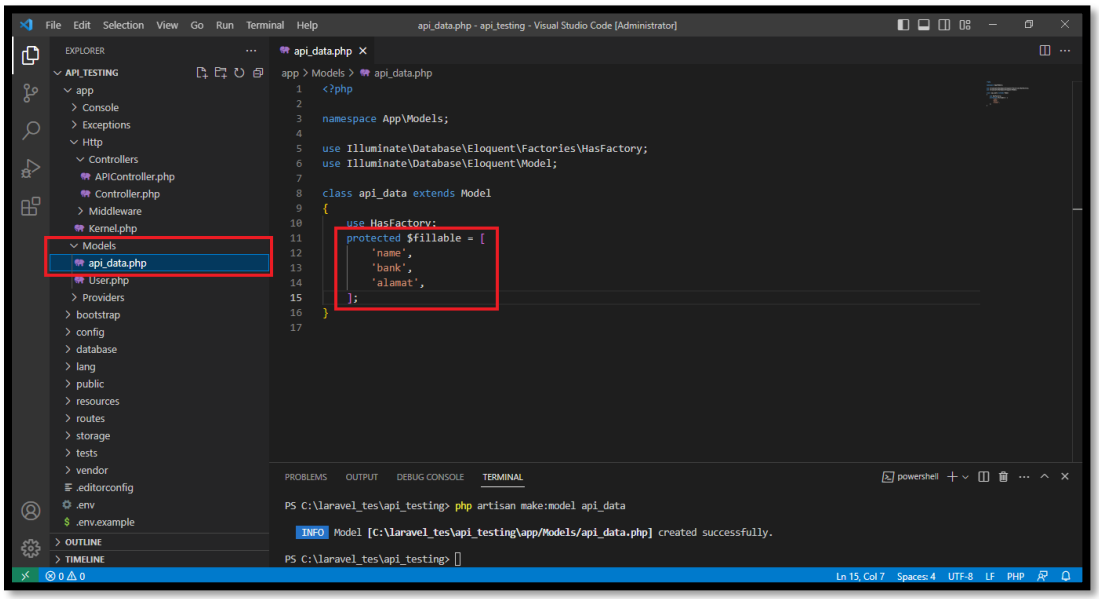
7. Buka kembali database pada browser anda dan pastikan bahwa tabel dengan nama “api_datas” telah berada di dalam database api_testing.



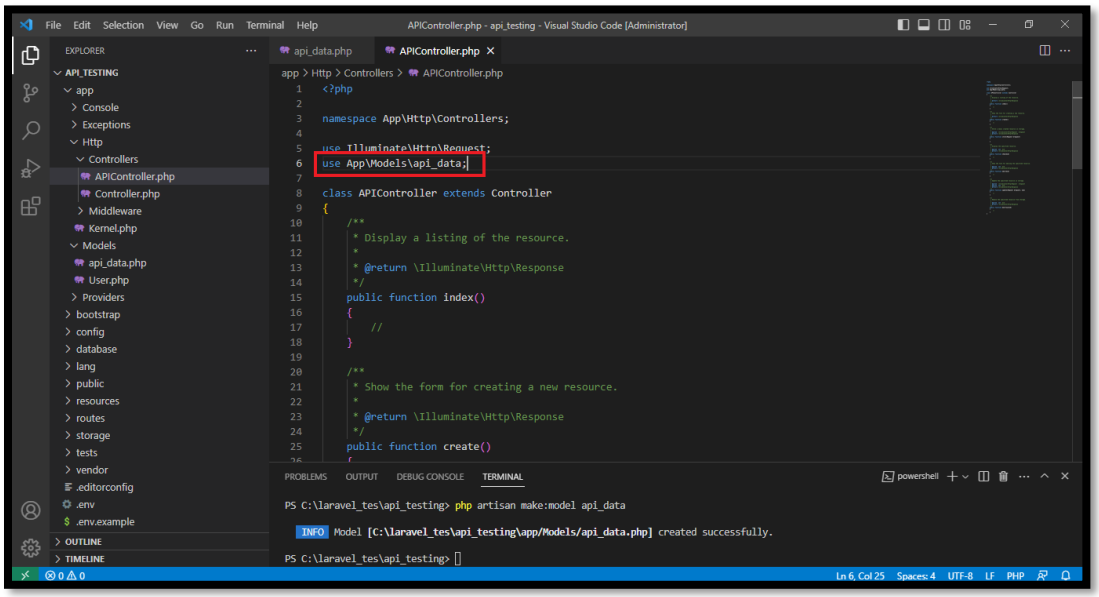
8. Ketikkan perintah "**php artisan make:controller APIController --resource**" dan klik enter untuk membuat sebuah controller. Kemudian tambahkan juga perintah "**php artisan make:model api_data**" untuk membuat sebuah model di laravel.



9. Buka file model dengan nama api_data yang baru saja dibuat pada folder "**app/Models/**" dan tambahkan kode program seperti gambar dibawah ini.



10. Kemudian buka file controller yang tadi telah dibuat pada folder "**app/Http/Controllers/**" dan tambahkan kode "**use App\Models\api_data;**" tanpa tanda kutip untuk mendeklarasikan model yang tadi telah kita buat.



11. Lalu ubah juga kode program pada fungsi index, show, store, edit dan destroy yang berada juga pada file APIController sebelumnya menjadi seperti potongan kode program dibawah ini.

```php
public function index()
{
    $data = api_data::all();
    return $data;
}
```

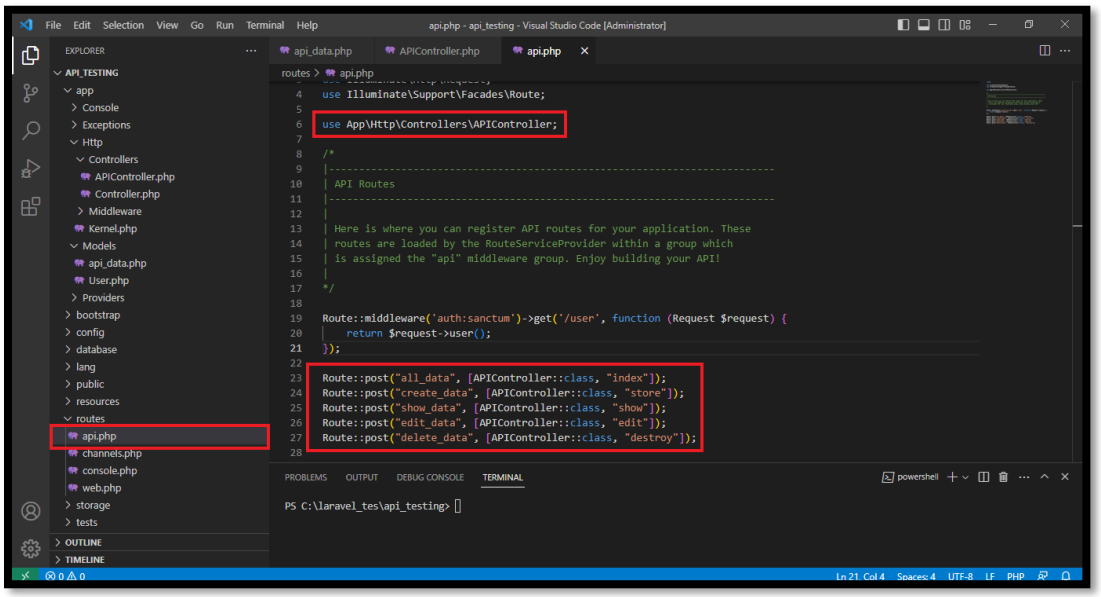
```php
public function show(Request $request)
{
    $data = api_data::all()->where('id', $request->id)->first();
    return $data;
}
```

```php
public function store(Request $request)
{
    $save = new api_data;
    $save->name = $request->name;
    $save->bank = $request->bank;
    $save->alamat = $request->alamat;
    $save->save();

    return "Berhasil Menyimpan Data";
}
```

```php
public function edit(Request $request)
{
    $data = api_data::all()->where('id', $request->id)->first();
    $data->name = $request->name;
    $data->bank = $request->bank;
    $data->alamat = $request->alamat;
    $data->save();
    return "Berhasil Mengubah Data";
}
```

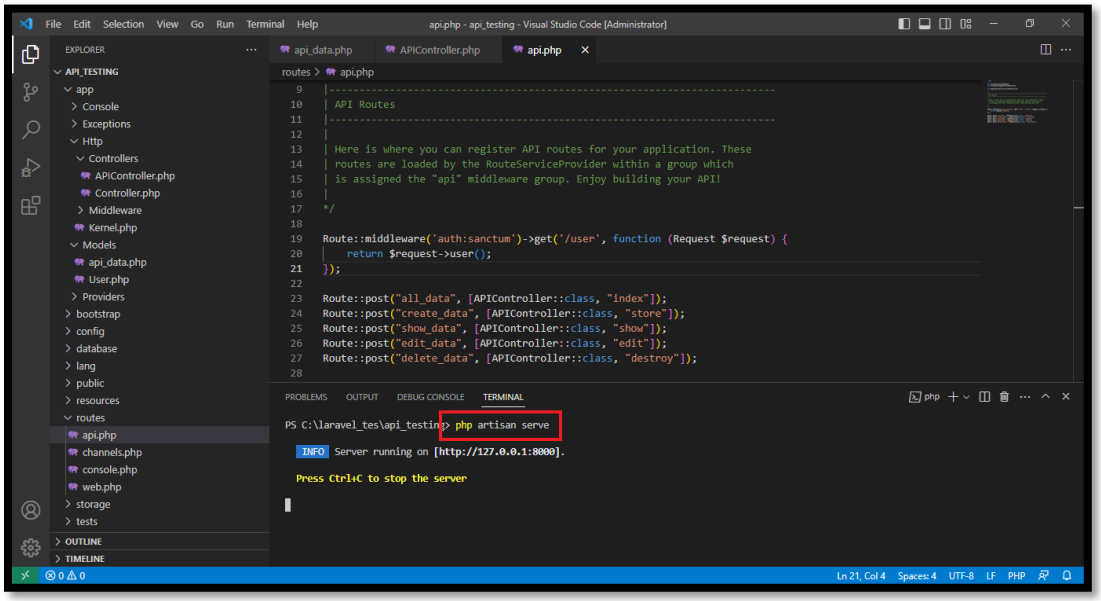
```php
public function destroy(Request $request)
{
    $del = api_data::all()->where('id', $request->id)->first();
    $del->delete();
    return "Berhasil Menghapus Data";
}
```

12. Buka file **api.php** yang ada dalam folder **routes** dan tambahkan kode program seperti yang berada dalam kotak merah pada gambar dibawah ini.

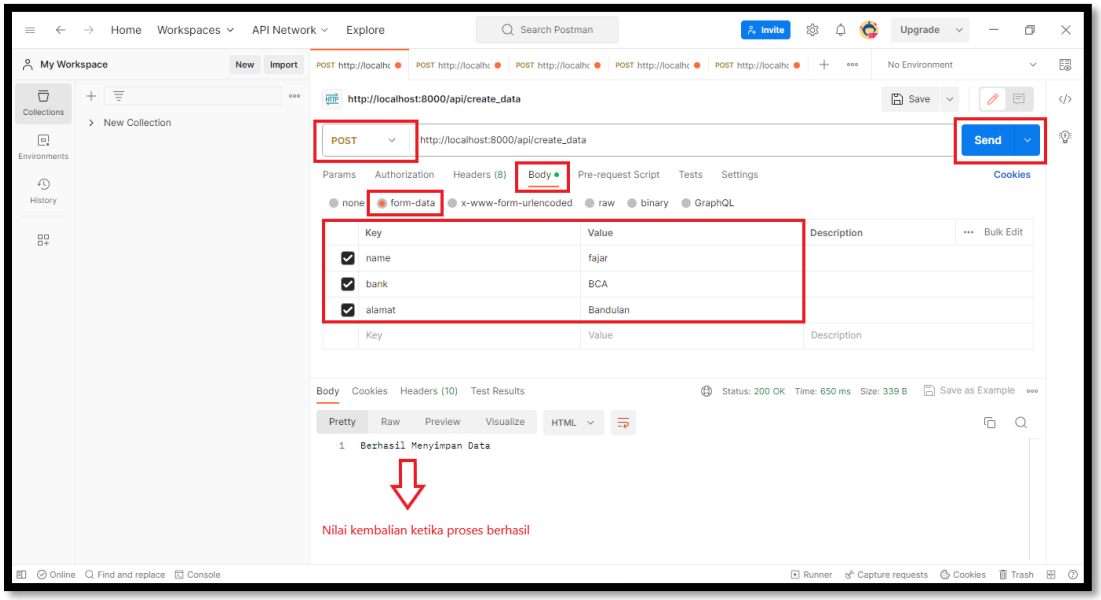


C. Pengujian REST API

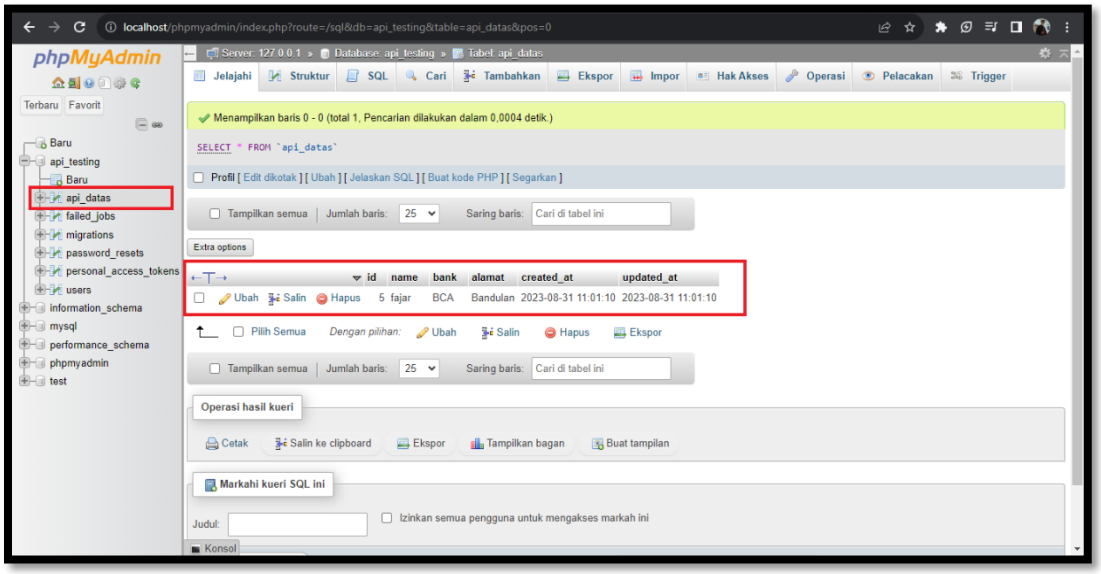
1. Jalankan perintah "php artisan serve" pada terminal Visual Studio Code untuk menjalankan proyek laravel anda. Pastikan laravel telah berjalan dengan benar.



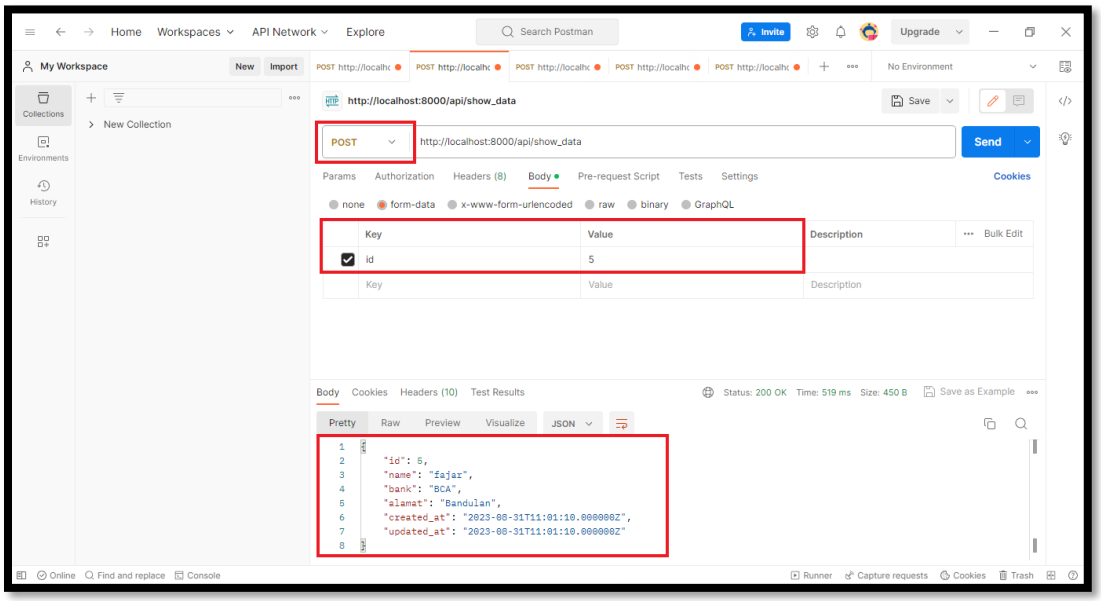
2. Kemudian buka aplikasi **postman** yang akan kita gunakan untuk menguji REST API pada laravel kita. Ubah metode pengiriman menjadi "POST" dan isikan url "**localhost:8000/api/create_data**" pada postman. Pada bagian menu yang ada dibawah box url pilih bagian **Body** dan pilih sub menu **form-data**. Lalu isikan **key** seperti pada gambar dibawah ini dan klik "Send". (Pengujian Create Data)



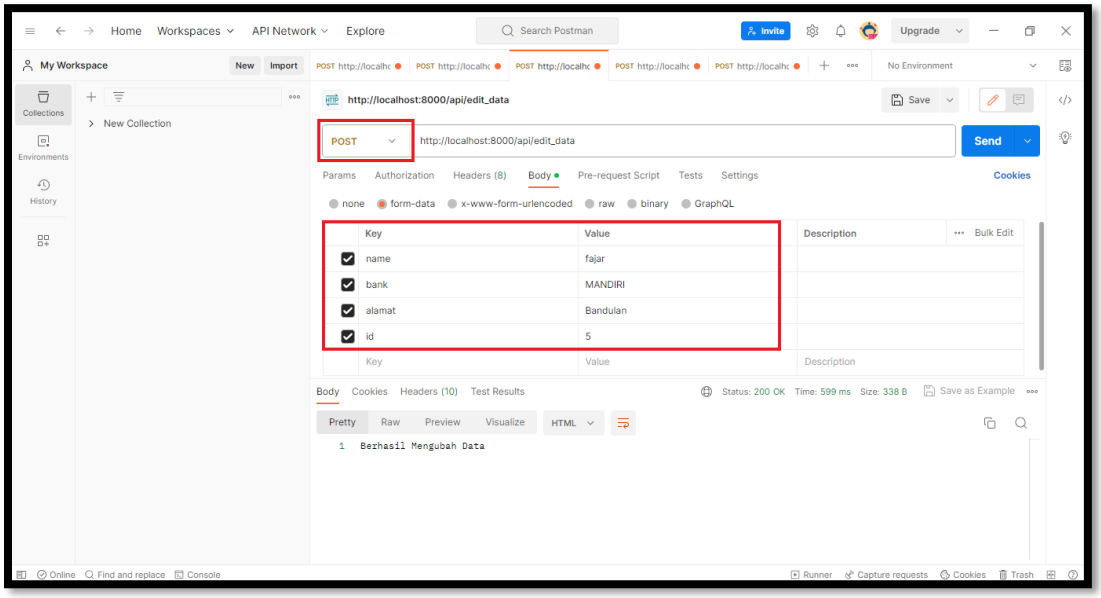
3. Buka browser anda dan lihat isi data pada tabel api_datas sebelumnya. Setelah anda mengakses REST API pada langkah sebelumnya harusnya sekarang pada tabel tersebut berisikan data yang telah anda masukkan sebelumnya.



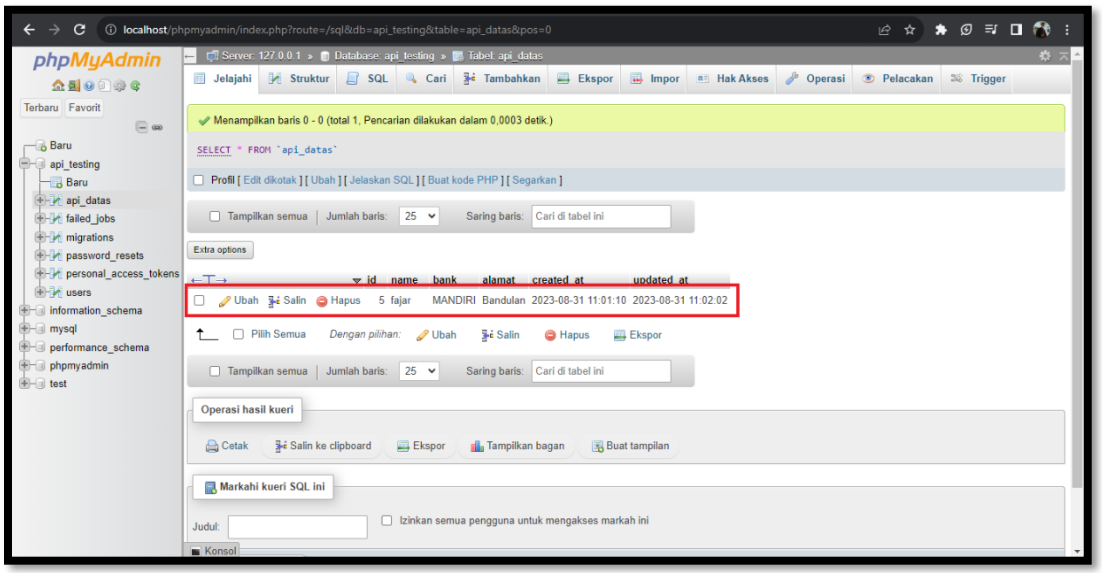
4. Ubah url pada postman menjadi "**localhost:8000/api/show_data**" dengan **form-data** yang berisi key **id** saja. Isikan value **id** dengan **id** data pada database anda lalu klik "Send". (Pengujian Read Data)



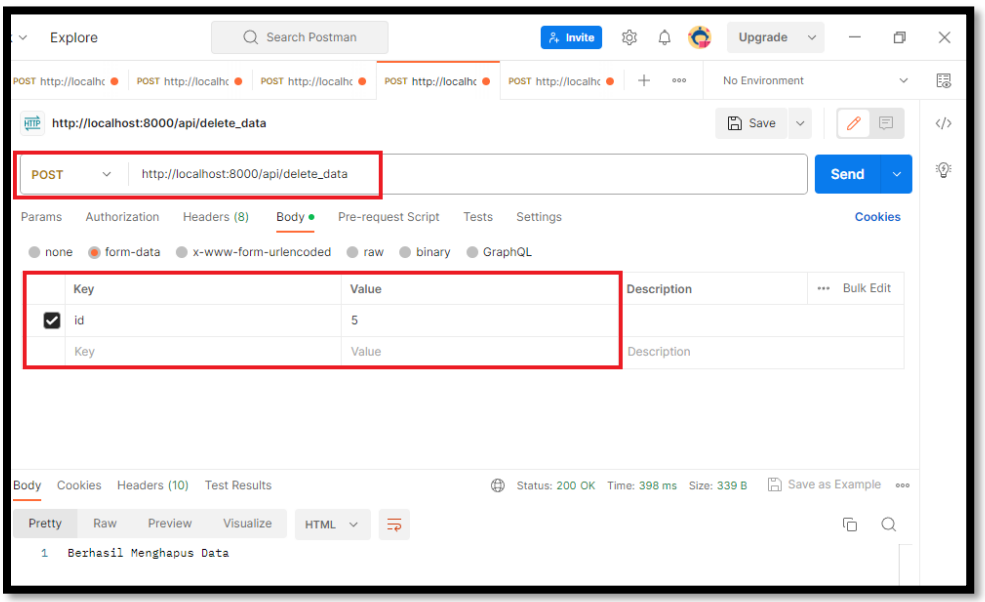
5. Ubah url pada postman menjadi "**localhost:8000/api/edit_data**" dengan **form-data** yang berisi name, bank, alamat dan id. Kemudian Isikan value **id** dengan **id** data pada database anda lalu klik "Send". (Pengujian Update Data)



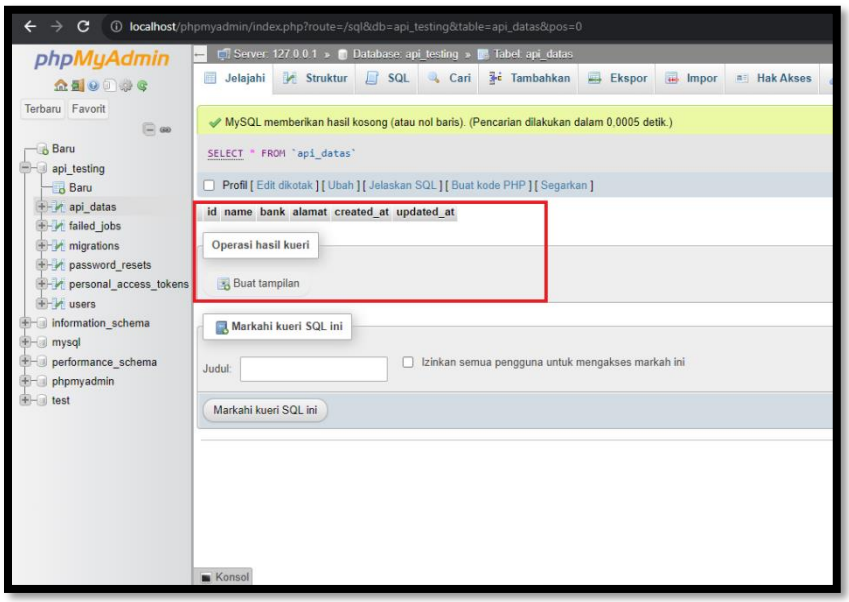
6. Buka browser anda dan refresh pada halaman data pada tabel api_datas, seharusnya data akan berubah seperti value terbaru yang baru saja anda masukkan sebelumnya.



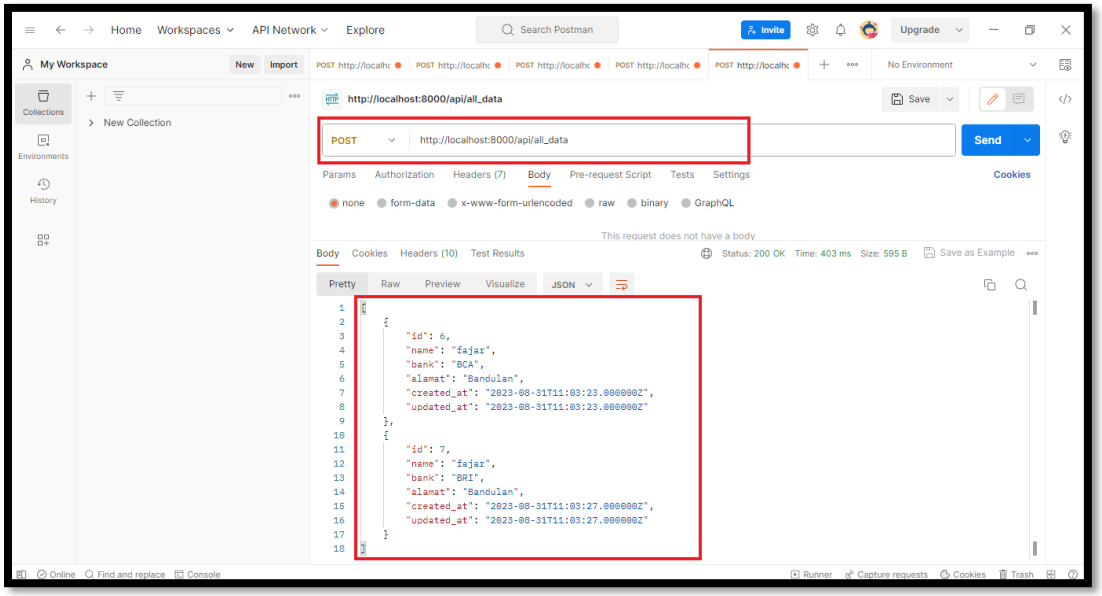
7. Ubah url pada postman menjadi "**localhost:8000/api/delete_data**" dengan **form-data** yang berisi key **id saja**. Kemudian Isikan value **id** dengan **id** data pada database anda lalu klik "Send". (Pengujian Delete Data).



8. Buka kembali browser anda dan refresh pada halaman database, data anda sebelumnya seharusnya sekarang telah berhasil dihapus.

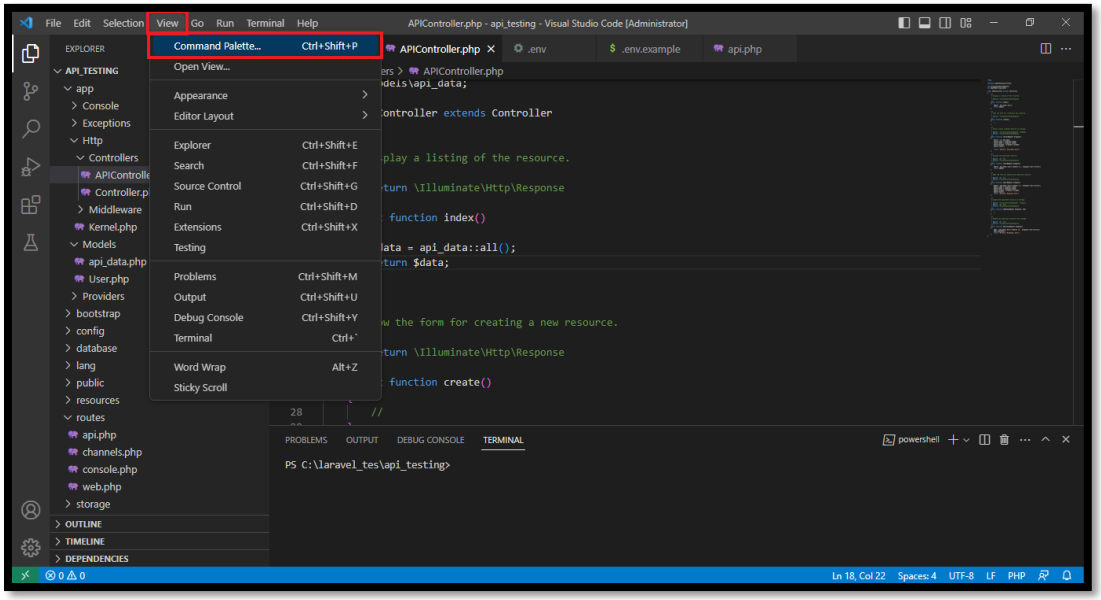


9. Lakukan penambahan data sebanyak 2x atau lebih dengan menjalankan Langkah ke 2 pada sub bab ini, kemudian ubah url pada postman menjadi "**localhost:8000/api/all_data**" tanpa **form-data** dan klik "Send". Lihat dan amati yang terjadi, seharusnya pada jendela postman anda akan menampilkan data dari semua data yang ada pada tabel api_datas pada database anda. (Pengujian Read All Data).

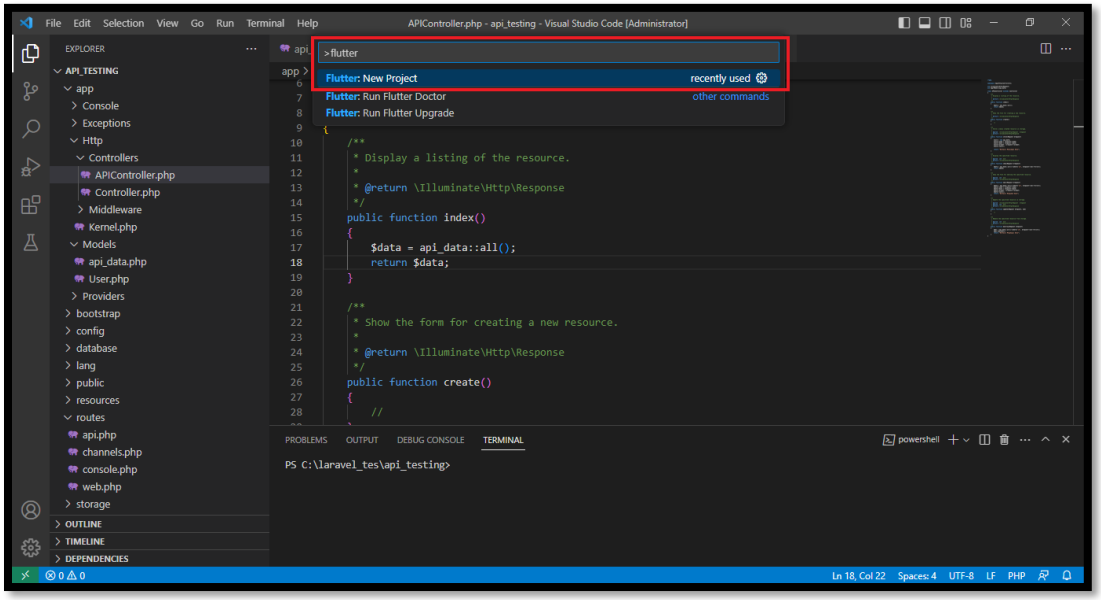


D. Pembuatan Proyek Flutter

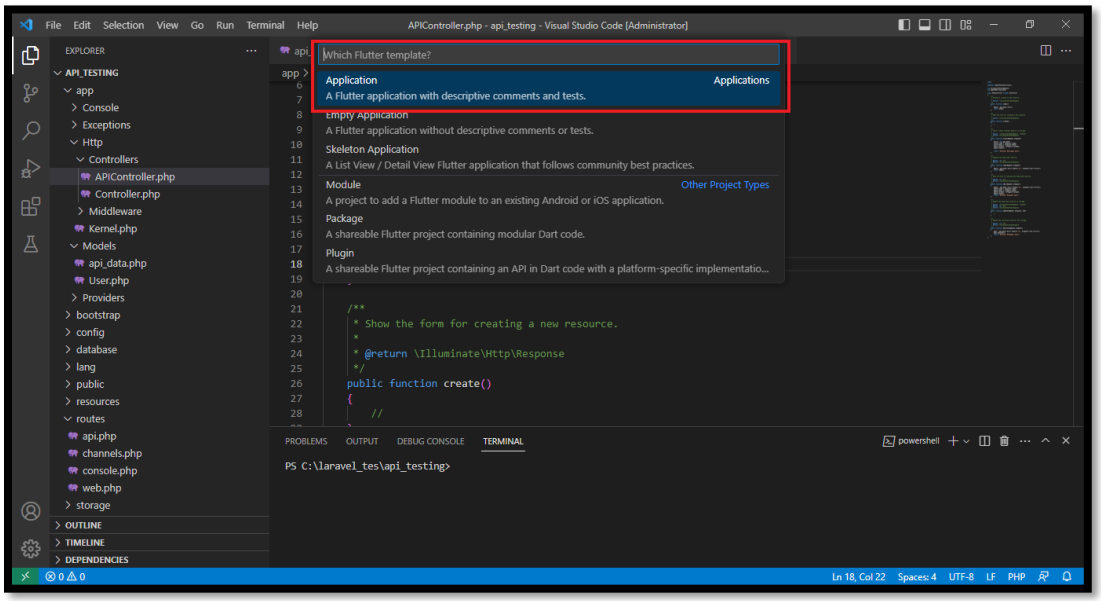
1. Buka aplikasi Visual Studio Code anda kemudian buka menu "**View**" dan pilih menu "**Command Palette…**".



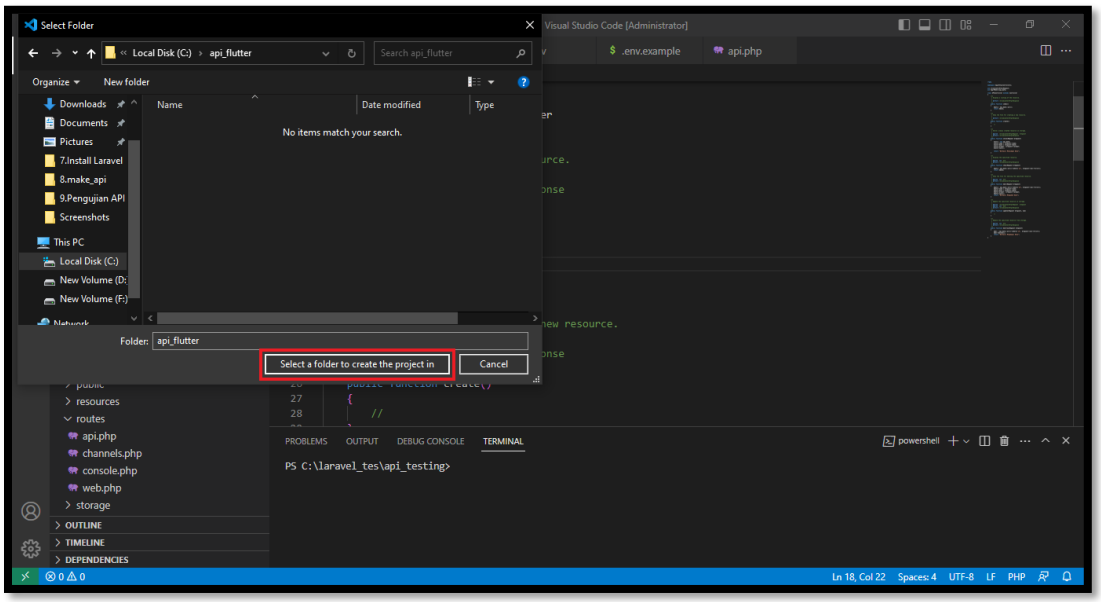
2. Ketikkan "**flutter**" pada searchbox yang muncul tanpa tanda kutip dan pilih "**Flutter: New Project**".



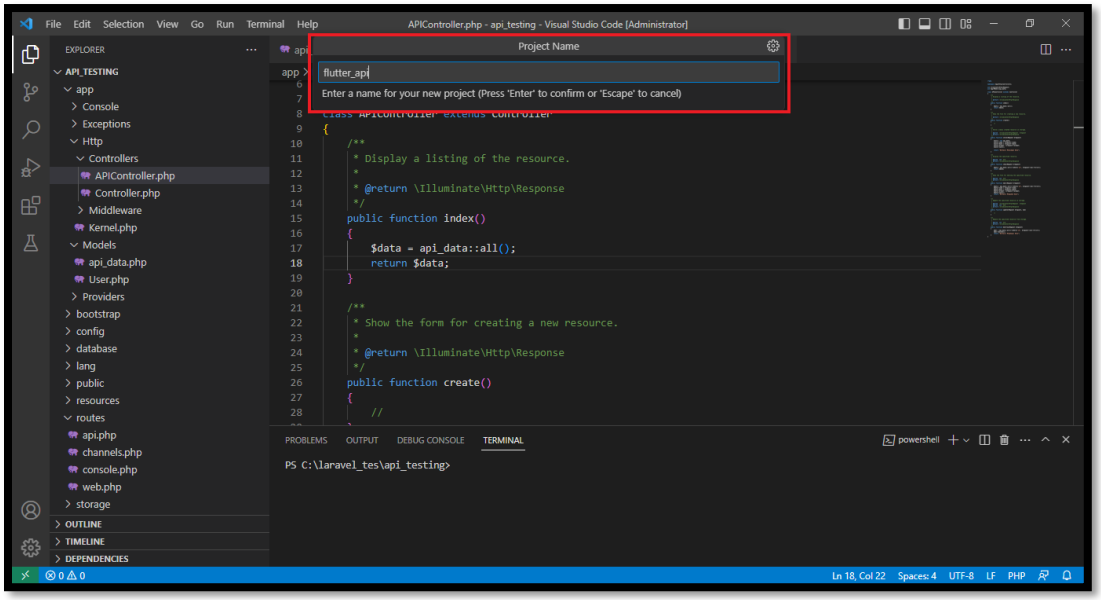
3. Kemudian pilih "**Flutter Application with descriptive..**" jangan pilih yang "Empty Application", karena kita akan menggunakan template yang sudah disediakan oleh flutter untuk mempermudah kita dalam menyelesaikan proyek ini.



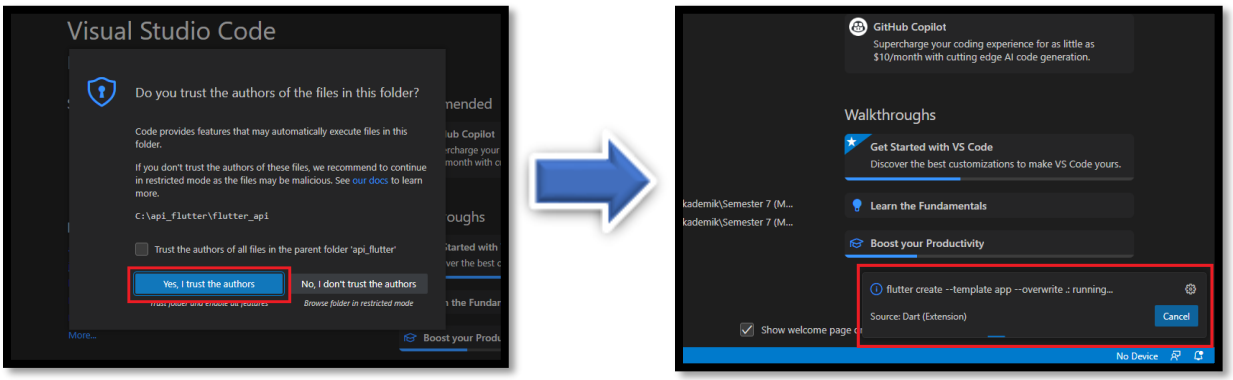
4. Kemudian pilih lokasi yang ingin anda gunakan dan klik "**Select a folder…**".



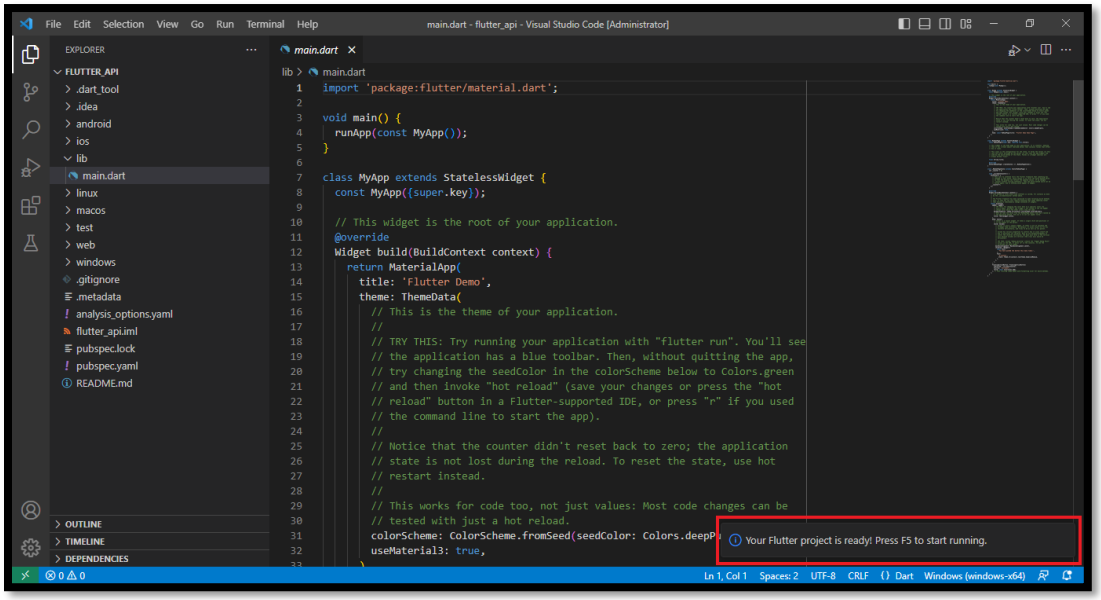
5. Lalu beri nama pada proyek yang ingin anda buat.



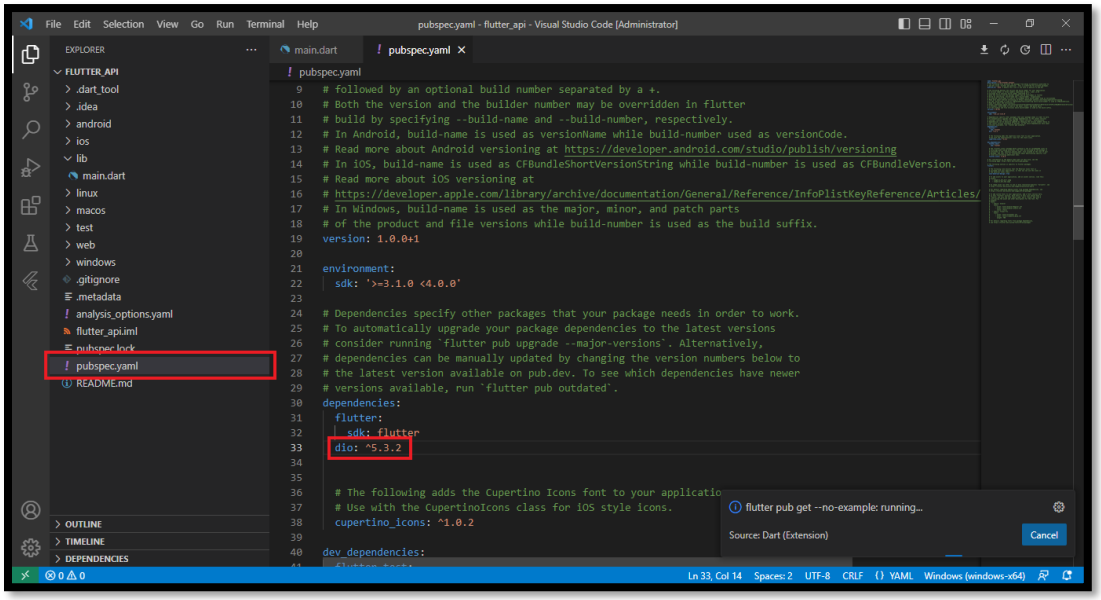
6. Setelah itu Visual Studio Code akan berpindah ke proyek flutter yang baru saja kita buat, apabila terdapat peringatan, pilih saja "Yes, I trush the author" dan tunggu hingga proses pembuatan proyek selesai.



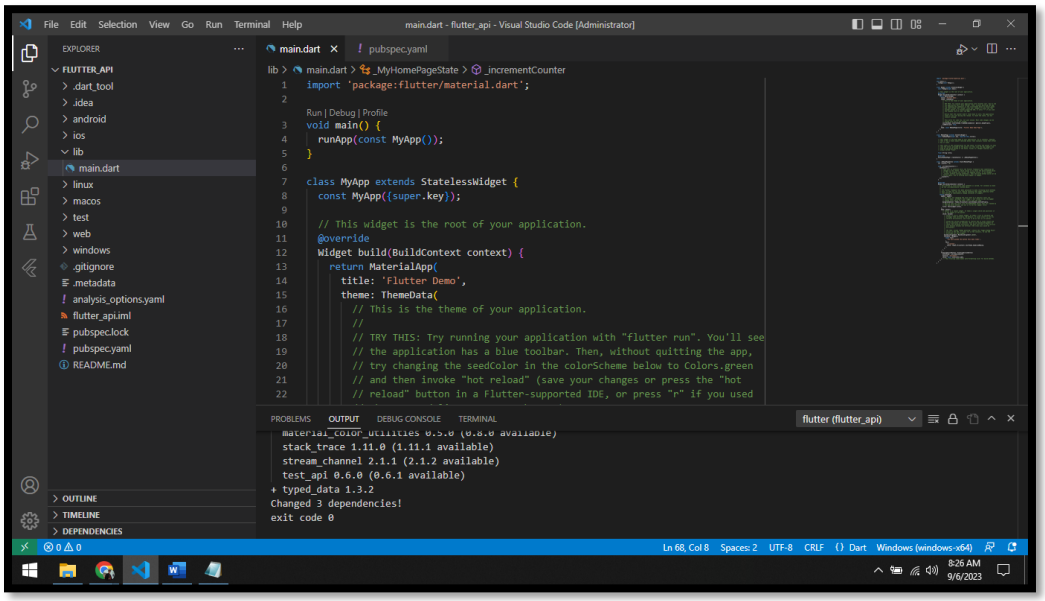
7. Berikut adalah tampilan proyek flutter yang telah berhasil dibuat.



8. Selanjutnya buka file pubspec.yml pada proyek dan ketikkan “dio: ^5.3.2” pada dependensi proyek flutter seperti gambar dibawah ini.



9. Buka kembali file main.dart dan sekarang pada file ini kita akan mulai mengerjakan UI dan backend untuk proyek ini nantinya. Ubah seluruh kode program pada file main.dart menjadi seperti kode program pada bagian Lampiran 1. Perhatikan langkah selanjutnya karena penjelasan dari kode program tersebut terdapat pada langkah setelah ini.



10. Seperti yang terlihat pada gambar dibawah, kode tersebut digunakan untuk melakukan control pada value yang dimiliki oleh input text pada proyek ini.

```
final dio = Dio();

var all_data = [];

final TextEditingController nameController = TextEditingController();
final TextEditingController bankController = TextEditingController();
final TextEditingController alamatController = TextEditingController();

final TextEditingController idController_update = TextEditingController();
final TextEditingController nameController_update = TextEditingController();
final TextEditingController bankController_update = TextEditingController();
final TextEditingController alamatController_update = TextEditingController();
```

11. Kode dibawah ini merupakan link url yang akan digunakan untuk mengakses REST API pada laravel kita, ubah url_domain menjadi ip address pada computer anda yang dapat dilihat pada command prompt dengan memasukkan perintah "ipconfig". (example: "http://ip_address:8000").

```
String url_domain = "http://10.208.18.217:8000/";
String url_all_data = url_domain + "api/all_data";
String url_create_data = url_domain + "api/create_data";
String url_show_data = url_domain + "api/show_data";
String url_update_data = url_domain + "api/edit_data";
String url_delete_data = url_domain + "api/delete_data";
```

12. Terdapat beberapa fungsi yang dideklarasikan dengan "void" yang berfungsi sebagai perantara untuk mengakses REST API pada laravel kita.

```
void update_data(String id, String name, String bank, String alamat) async {
  Response response;
  response = await dio.post(
    url_update_data,
    queryParameters: {'id': id, 'name': name, 'bank': bank, 'alamat': alamat},
  );

  idController_update.text = "";
  nameController_update.text = "";
  bankController_update.text = "";
  alamatController_update.text = "";
}
```

```
void delete_data(int id) async {
  Response response;
  response = await dio.post(
    url_delete_data,
    queryParameters: {'id': id},
  );
}
```

```
void show_all_data() async {
  Response response;
  response = await dio.post(
    url_all_data,
  );

  all_data = response.data;
}
```

```
void create_data(String name, String bank, String alamat) async {
  Response response;
  response = await dio.post(
    url_create_data,
    queryParameters: {'name': name, 'bank': bank, 'alamat': alamat},
  );

  nameController.text = "";
  bankController.text = "";
  alamatController.text = "";
}
```

13. Pada proyek ini kita menggunakan 2 buah Pop-up yang digunakan sebagai tempat pengisian Form. Form yang berada pada Pop-up ada 2 yaitu form untuk menambahkan data baru dan form untuk mengubah data yang sudah ada.

```
Widget _buildPopupDialog(BuildContext context) {
  return new AlertDialog(
```

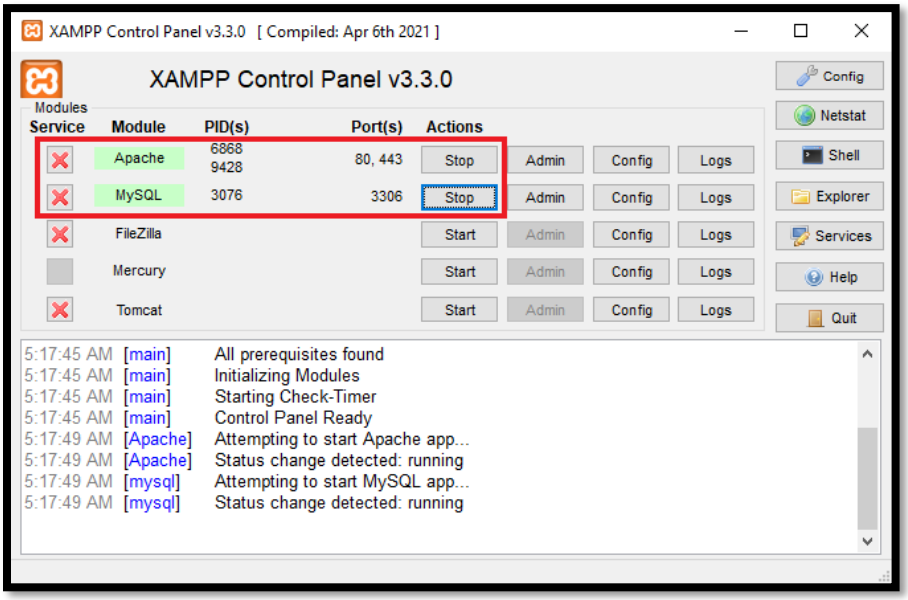
```
Widget _buildPopupDialog_update(BuildContext context) {
  return new AlertDialog(
```

14. Kode dibawah ini berfungsi untuk melakukan reload pada data tabel yang sudah kita buat. Agar data dapat berubah ketika kita mengubah maupun menambah serta menghapus data yang sudah ada.

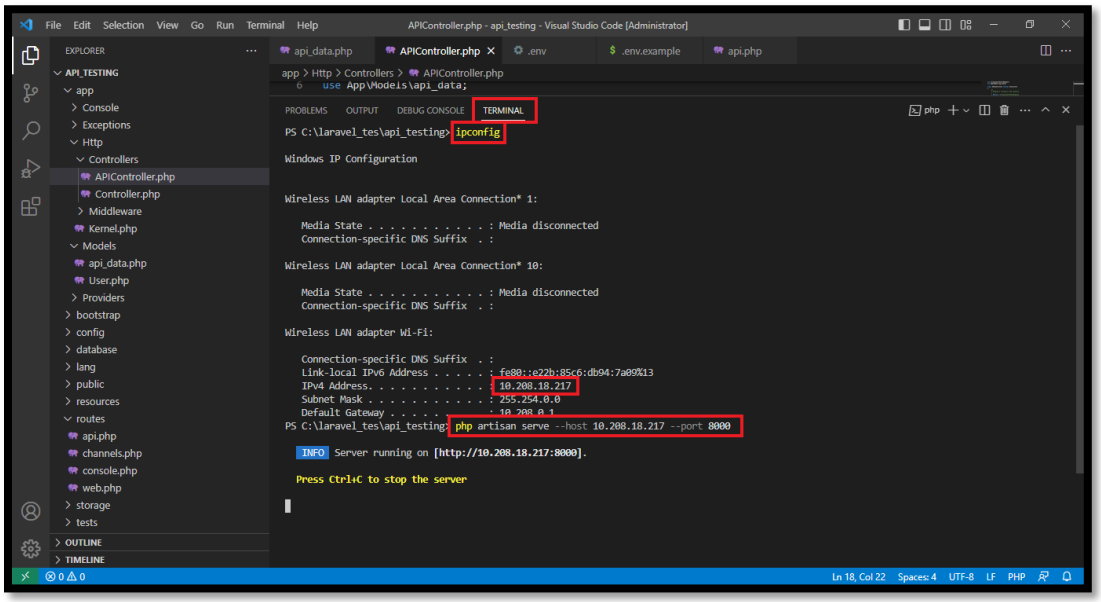
```
void dispose() {
  WidgetsBinding.instance.removeObserver(this);
  super.dispose();
}

@override
void didChangeAppLifecycleState(AppLifecycleState state) {
  print('MyApp state = $state');
  if (state == AppLifecycleState.inactive) {
    // app transitioning to other state.
  } else if (state == AppLifecycleState.paused) {
    // app is on the background.
  } else if (state == AppLifecycleState.detached) {
    // flutter engine is running but detached from views
  } else if (state == AppLifecycleState.resumed) {
    // app is visible and running.
    runApp(MyApp()); // run your App class again
  }
}
```

E. Pengujian Flutter

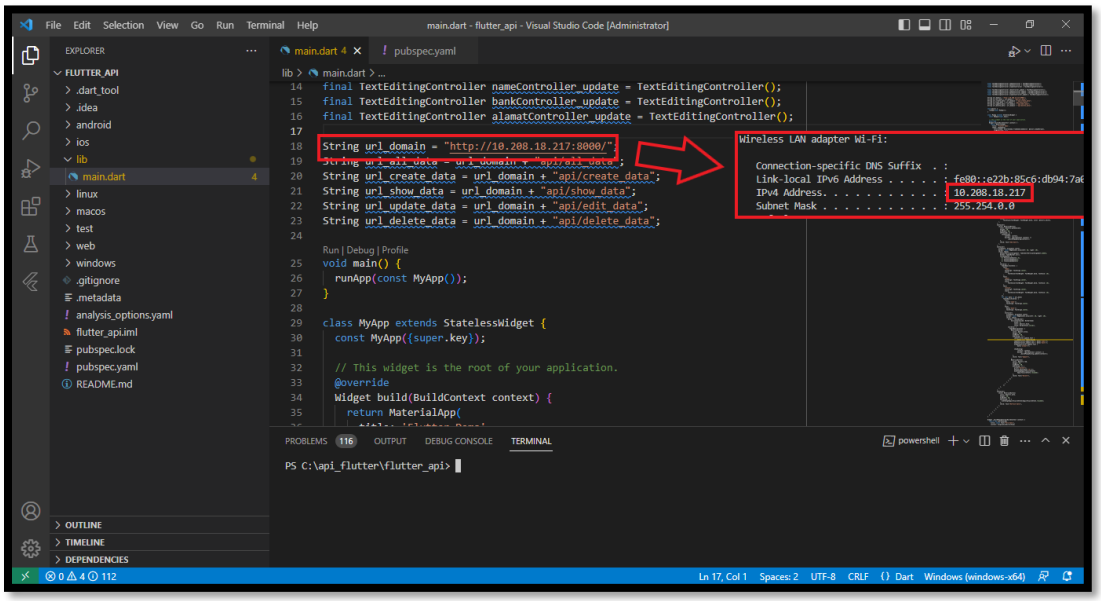
1. Buka aplikasi XAMPP dan pastikan Module "**Apache dan MySQL**" sudah berjalan.



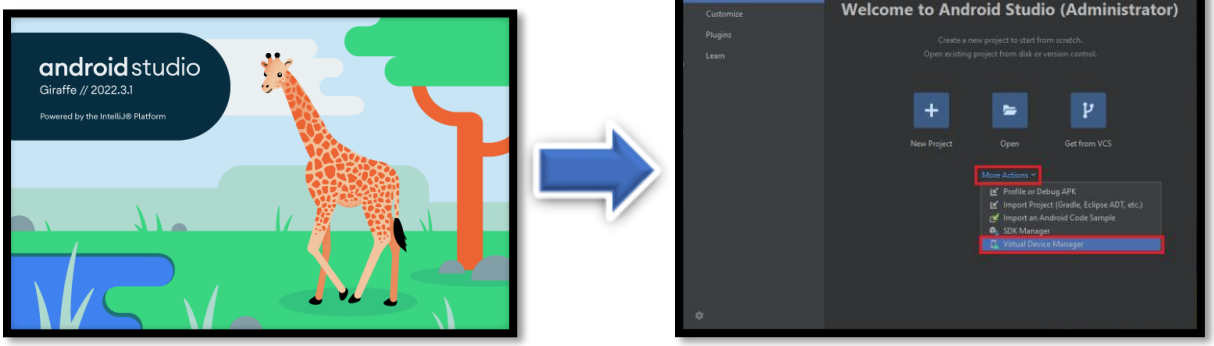
2. Kemudian buka proyek laravel sebelumnya dan ketik "ipconfig" pada terminal lalu enter. Terminal akan menampilkan detail dari address yang ada pada perangkat anda kemudian ketikkan perintah "**php artisan serve –host ip_anda –port 8000**" tanpa tanda kutip dan ubah ip_anda menjadi Alamat ip_address yang tadi telah ditampilkan oleh terminal. Kemudian klik enter dan silahkan minimize proyek laravel anda tetapi jangan di tutup.



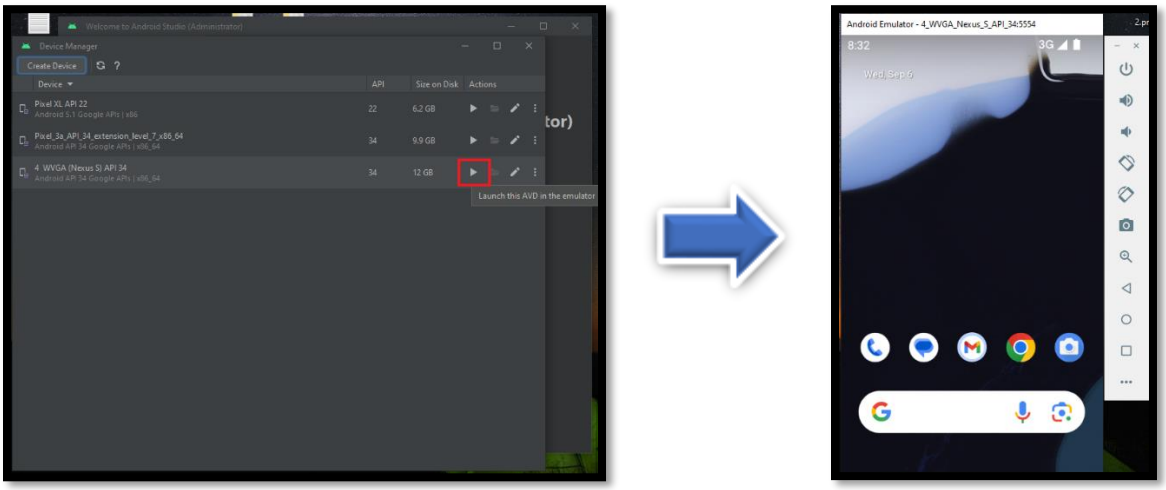
3. Kemudian buka proyek flutter anda dengan Visual Studio Code. Jadi disini anda akan membuka 2 Visual Studio Code dimana satu untuk membuka proyek laravel dan satunya untuk membuka proyek flutter. Ubah value dari url_domain dengan ip_address anda sebelumnya.



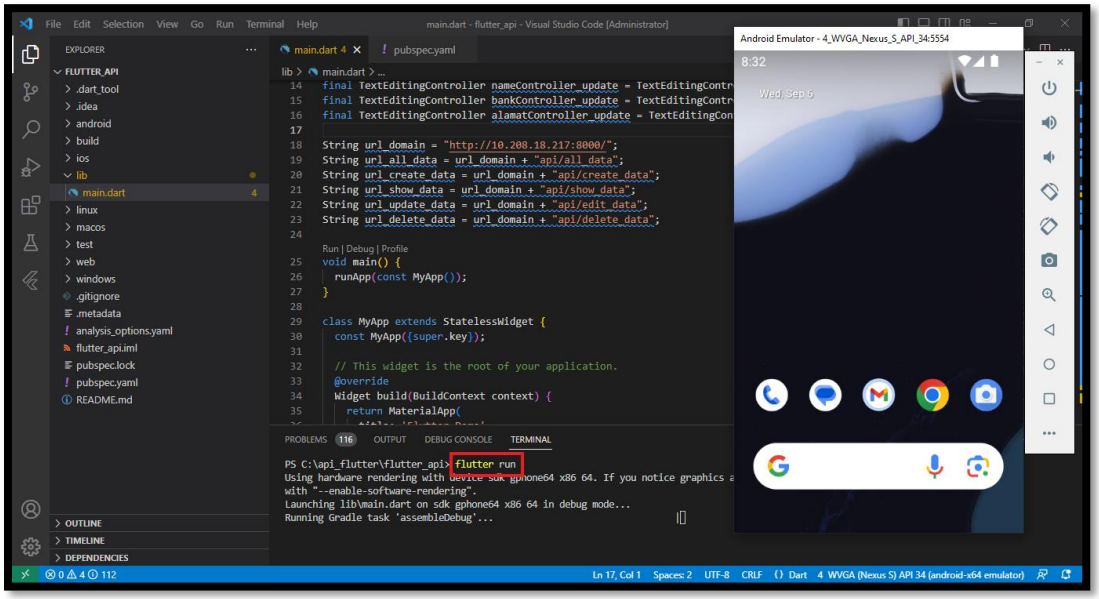
4. Selanjutnya buka aplikasi android studio, tunggu proses loading selesai dan pilih "More Action" kemudian pilih **Virtual Device Manager**.



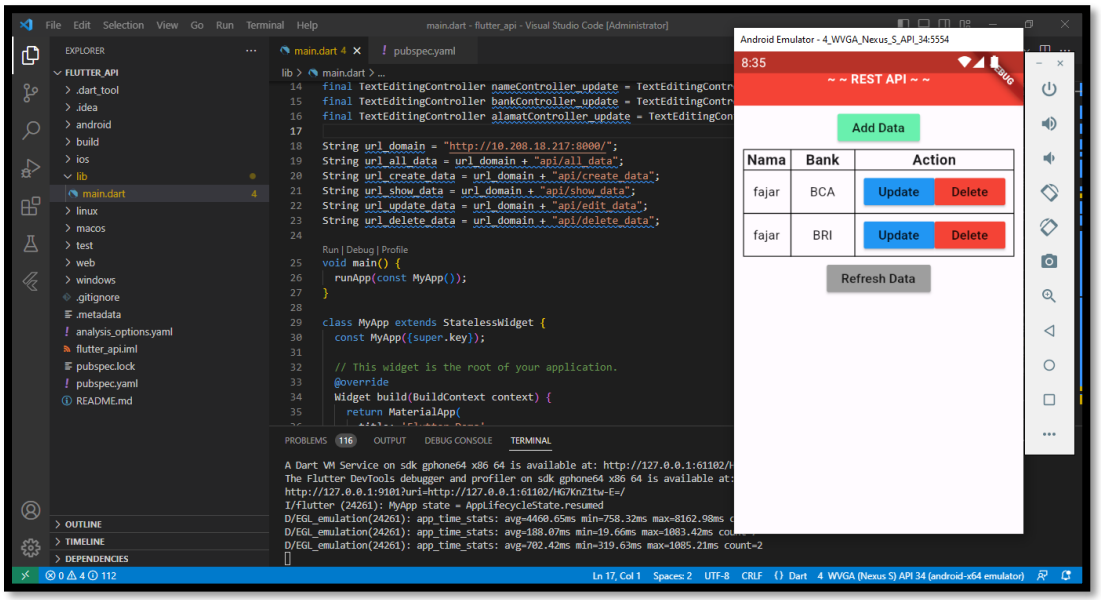
5. Klik tombol play pada salah satu device android emulator yang telah anda install. Karena android SDK ini tergolong memberatkan pada perangkat anda, pastikan anda telah mengatur terlebih dahulu dengan mengklik tombol “Pensil” disebelah kanan tombol “Play” agar android device dapat berjalan lancar.



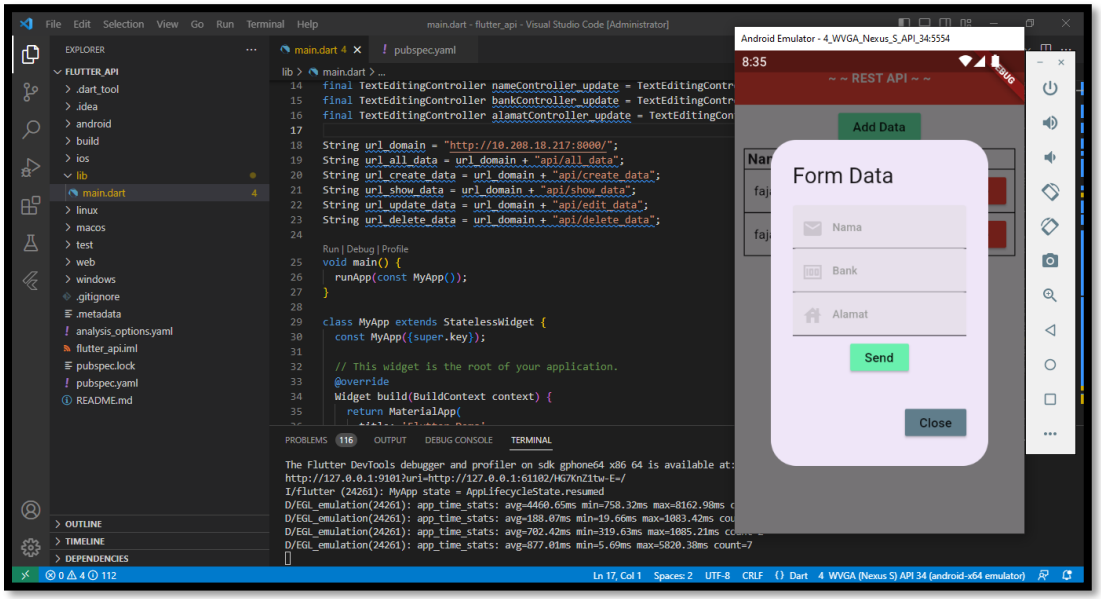
6. Setelah android SDK berhasil dibuka dengan lancer, kembali ke terminal pada proyek flutter anda dan ketik “flutter run” tanpa tanda kutip lalu enter.



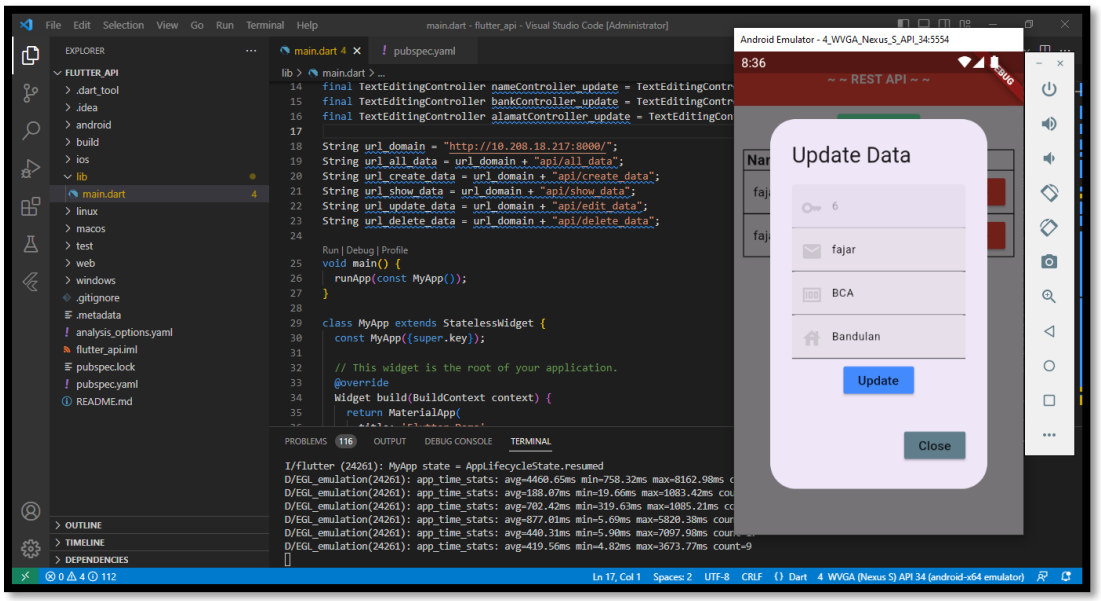
7. Tunggu proses loading selesai dan android SDK akan menampilkan output seperti di bawah ini. Terdapat berbagai macam tombol yang dapat dioperasikan, seperti tombol “Add Data” yang berguna untuk menambahkan data baru, tombol “Refresh Data” berguna untuk merefresh data pada tabel, serta tombol “Updata” dan “Delete” yang berguna untuk mengubah dan menghapus.



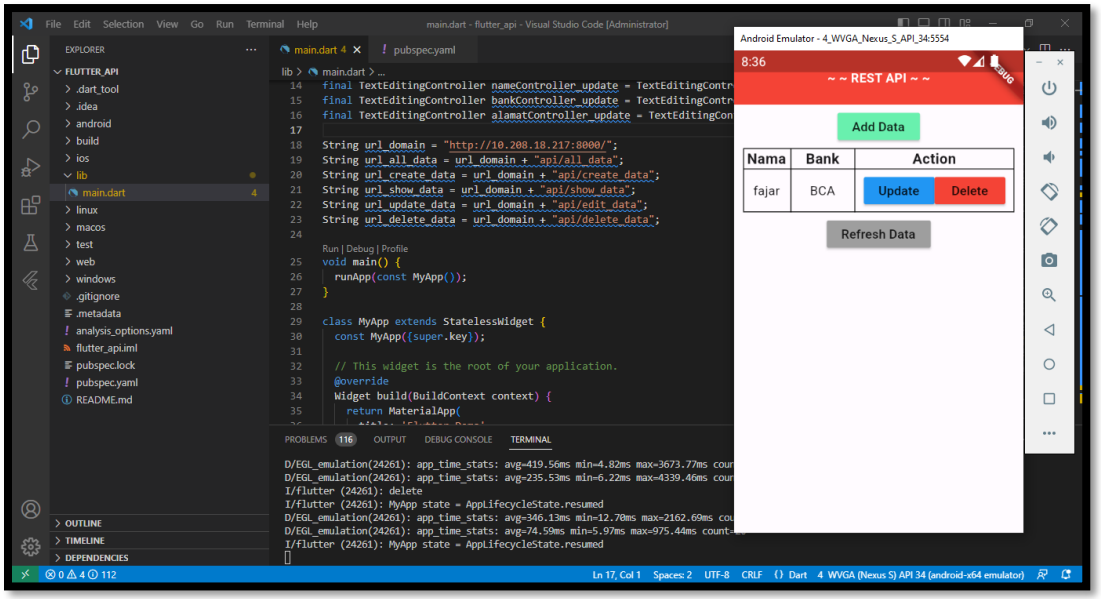
8. Apabila anda mengklik tombol “Add Data” sistem akan menampilkan Form untuk mengisi data yang baru.



9. Kemudian apabila anda mengklik tombol “Update” anda akan mendapatkan form untuk mengubah data yang anda pilih.



10. Tombol “Delete” akan langsung menghapus data yang anda pilih untuk dihapus.



BAB IV
LAMPIRAN CODE PROGRAM

Lampiran 1

Main.dart

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

import 'package:dio/dio.dart';

final dio = Dio();

var all_data = [];

final TextEditingController nameController = TextEditingController();
final TextEditingController bankController = TextEditingController();
final TextEditingController alamatController = TextEditingController();

final TextEditingController idController_update = TextEditingController();
final TextEditingController nameController_update = TextEditingController();
final TextEditingController bankController_update = TextEditingController();
final TextEditingController alamatController_update = TextEditingController();

String url_domain = "http://10.208.18.217:8000/";
String url_all_data = url_domain + "api/all_data";
String url_create_data = url_domain + "api/create_data";
String url_show_data = url_domain + "api/show_data";
String url_update_data = url_domain + "api/edit_data";
String url_delete_data = url_domain + "api/delete_data";

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
```

```
      ),
      home: data_tes(),
    );
  }
}

class data_tes extends StatefulWidget {
  const data_tes({super.key});

  @override
  State<data_tes> createState() => _data_tesState();
}

class _data_tesState extends State<data_tes> with WidgetsBindingObserver {
  void initState() {
    super.initState();
    WidgetsBinding.instance.addObserver(this);
  }

  @override
  void dispose() {
    WidgetsBinding.instance.removeObserver(this);
    super.dispose();
  }

  @override
  void didChangeAppLifecycleState(AppLifecycleState state) {
    print('MyApp state = $state');
    if (state == AppLifecycleState.inactive) {
      // app transitioning to other state.
    } else if (state == AppLifecycleState.paused) {
      // app is on the background.
    } else if (state == AppLifecycleState.detached) {
      // flutter engine is running but detached from views
    } else if (state == AppLifecycleState.resumed) {
      // app is visible and running.
      runApp(MyApp()); // run your App class again
    }
  }

  @override
  Widget build(BuildContext context) {
    show_all_data();

    return Scaffold(
```

```
      body: Center(
        child: Column(
          // crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            Container(
              height: 60,
              decoration: BoxDecoration(color: Colors.red),
              alignment: Alignment.center,
              child: Text(
                '~ ~ REST API ~ ~',
                style:
                    TextStyle(fontWeight: FontWeight.bold, color:
Colors.white),
              ),
            ),
            Container(
              child: MaterialButton(
                color: Colors.greenAccent,
                height: 30,
                minWidth: 20,
                onPressed: () {
                  showDialog(
                      context: context,
                      builder: (BuildContext context) =>
                          _buildPopupDialog(context));
                },
                child: Text("Add Data"),
              ),
            ),
            Container(
              alignment: Alignment.center,
              margin: const EdgeInsets.only(left: 10, right: 10),
              child: Table(
                defaultVerticalAlignment: TableCellVerticalAlignment.middle,
                border: TableBorder.all(),
                columnWidths: {
                  0: FlexColumnWidth(1.5),
                  1: FlexColumnWidth(2),
                  2: FlexColumnWidth(5)
                },
                children: [
                  TableRow(children: [
                    Text(
                      'Nama',
                      textAlign: TextAlign.center,
```

```
                    style:
                        TextStyle(fontWeight: FontWeight.bold, fontSize: 16),
                  ),
                  Text(
                    'Bank',
                    textAlign: TextAlign.center,
                    style:
                        TextStyle(fontWeight: FontWeight.bold, fontSize: 16),
                  ),
                  Text(
                    'Action',
                    textAlign: TextAlign.center,
                    style:
                        TextStyle(fontWeight: FontWeight.bold, fontSize: 16),
                  )
                ]),
                for (var data in all_data)
                  TableRow(children: [
                    Text(
                      data['name']!,
                      textAlign: TextAlign.center,
                    ),
                    Text(
                      data['bank']!,
                      textAlign: TextAlign.center,
                    ),
                    Container(
                      alignment: Alignment.center,
                      margin: const EdgeInsets.only(left: 10, right: 10),
                      child: Table(
                        border: TableBorder(
                            horizontalInside: BorderSide(
                                width: 0,
                                color: Colors.blue,
                                style: BorderStyle.solid)),
                        children: [
                          TableRow(children: [
                            MaterialButton(
                              color: Colors.blue,
                              height: 30,
                              minWidth: 20,
                              onPressed: () {
                                idController_update.text =
                                    data['id'].toString()!;
                                nameController_update.text = data['name']!;
```

```
                                bankController_update.text = data['bank']!;
                                alamatController_update.text =
                                    data['alamat']!;

                                showDialog(
                                    context: context,
                                    builder: (BuildContext context) =>
                                        _buildPopupDialog_update(context));
                              },
                              child: Text("Update"),
                            ),
                            MaterialButton(
                              color: Colors.red,
                              height: 30,
                              minWidth: 20,
                              onPressed: () {
                                print("delete");
                                delete_data(data['id']!);
                                didChangeAppLifecycleState(
                                    AppLifecycleState.resumed);
                              },
                              child: Text("Delete"),
                            ),
                          ])
                        ],
                      ),
                    )
                  ])
              ],
            ),
          ),
          Container(
            child: MaterialButton(
              color: Colors.grey,
              height: 30,
              minWidth: 20,
              onPressed: () {
                didChangeAppLifecycleState(AppLifecycleState.resumed);
              },
              child: Text("Refresh Data"),
            ),
          ),
        ],
      ),
    ),
```

```
    );
  }
}

Widget _buildPopupDialog(BuildContext context) {
  return new AlertDialog(
    title: const Text('Form Data'),
    content: SingleChildScrollView(
      child: Column(
        children: [
          TextField(
            style: TextStyle(fontSize: 12),
            controller: nameController,
            textAlign: TextAlign.left,
            decoration: InputDecoration(
              prefixIconConstraints:
                  BoxConstraints(minWidth: 23, maxHeight: 20),
              prefixIcon: Padding(
                padding: const EdgeInsets.only(right: 10, left: 10),
                child: Icon(
                  Icons.email,
                  color: Colors.black12,
                ),
              ),
              filled: true,
              contentPadding: EdgeInsets.symmetric(vertical: 0),
              hintText: 'Nama',
              hintStyle: TextStyle(color: Colors.grey),
            ),
          ),
          TextField(
            style: TextStyle(fontSize: 12),
            controller: bankController,
            textAlign: TextAlign.left,
            decoration: InputDecoration(
              prefixIconConstraints:
                  BoxConstraints(minWidth: 23, maxHeight: 20),
              prefixIcon: Padding(
                padding: const EdgeInsets.only(right: 10, left: 10),
                child: Icon(
                  Icons.money,
                  color: Colors.black12,
                ),
              ),
              filled: true,
```

```
            contentPadding: EdgeInsets.symmetric(vertical: 0),
            hintText: 'Bank',
            hintStyle: TextStyle(color: Colors.grey),
          ),
        ),
        TextField(
          style: TextStyle(fontSize: 12),
          controller: alamatController,
          textAlign: TextAlign.left,
          decoration: InputDecoration(
            prefixIconConstraints:
                BoxConstraints(minWidth: 23, maxHeight: 20),
            prefixIcon: Padding(
              padding: const EdgeInsets.only(right: 10, left: 10),
              child: Icon(
                Icons.house,
                color: Colors.black12,
              ),
            ),
            filled: true,
            contentPadding: EdgeInsets.symmetric(vertical: 0),
            hintText: 'Alamat',
            hintStyle: TextStyle(color: Colors.grey),
          ),
        ),
        MaterialButton(
          color: Colors.greenAccent,
          height: 30,
          minWidth: 20,
          onPressed: () {
            print("object");
            if (nameController.text == "" ||
                bankController.text == "" ||
                alamatController.text == "") {
              print("zero");
            } else {
              create_data(nameController.text, bankController.text,
                  alamatController.text);
              Navigator.of(context).pop();
            }
          },
          child: Text("Send"),
        ),
      ],
    ),
```

```
    ),
    actions: <Widget>[
      new MaterialButton(
        color: Colors.blueGrey,
        height: 30,
        minWidth: 20,
        onPressed: () {
          Navigator.of(context).pop();
        },
        child: Text("Close"),
      ),
    ],
  );
}

Widget _buildPopupDialog_update(BuildContext context) {
  return new AlertDialog(
    title: const Text('Update Data'),
    content: SingleChildScrollView(
      child: Column(
        children: [
          TextField(
            enabled: false,
            style: TextStyle(fontSize: 12),
            controller: idController_update,
            textAlign: TextAlign.left,
            decoration: InputDecoration(
              prefixIconConstraints:
                  BoxConstraints(minWidth: 23, maxHeight: 20),
              prefixIcon: Padding(
                padding: const EdgeInsets.only(right: 10, left: 10),
                child: Icon(
                  Icons.key,
                  color: Colors.black12,
                ),
              ),
              filled: true,
              contentPadding: EdgeInsets.symmetric(vertical: 0),
              hintText: 'Id',
              hintStyle: TextStyle(color: Colors.grey),
            ),
          ),
          TextField(
            style: TextStyle(fontSize: 12),
            controller: nameController_update,
```

```
            textAlign: TextAlign.left,
            decoration: InputDecoration(
              prefixIconConstraints:
                  BoxConstraints(minWidth: 23, maxHeight: 20),
              prefixIcon: Padding(
                padding: const EdgeInsets.only(right: 10, left: 10),
                child: Icon(
                  Icons.email,
                  color: Colors.black12,
                ),
              ),
              filled: true,
              contentPadding: EdgeInsets.symmetric(vertical: 0),
              hintText: 'Nama',
              hintStyle: TextStyle(color: Colors.grey),
            ),
          ),
          TextField(
            style: TextStyle(fontSize: 12),
            controller: bankController_update,
            textAlign: TextAlign.left,
            decoration: InputDecoration(
              prefixIconConstraints:
                  BoxConstraints(minWidth: 23, maxHeight: 20),
              prefixIcon: Padding(
                padding: const EdgeInsets.only(right: 10, left: 10),
                child: Icon(
                  Icons.money,
                  color: Colors.black12,
                ),
              ),
              filled: true,
              contentPadding: EdgeInsets.symmetric(vertical: 0),
              hintText: 'Bank',
              hintStyle: TextStyle(color: Colors.grey),
            ),
          ),
          TextField(
            style: TextStyle(fontSize: 12),
            controller: alamatController_update,
            textAlign: TextAlign.left,
            decoration: InputDecoration(
              prefixIconConstraints:
                  BoxConstraints(minWidth: 23, maxHeight: 20),
              prefixIcon: Padding(
```

```
              padding: const EdgeInsets.only(right: 10, left: 10),
              child: Icon(
                Icons.house,
                color: Colors.black12,
              ),
            ),
            filled: true,
            contentPadding: EdgeInsets.symmetric(vertical: 0),
            hintText: 'Alamat',
            hintStyle: TextStyle(color: Colors.grey),
          ),
        ),
        MaterialButton(
          color: Colors.blueAccent,
          height: 30,
          minWidth: 20,
          onPressed: () {
            print("object");
            if (idController_update.text == "" ||
                nameController_update.text == "" ||
                bankController_update.text == "" ||
                alamatController_update.text == "") {
              print("zero");
            } else {
              update_data(
                  idController_update.text,
                  nameController_update.text,
                  bankController_update.text,
                  alamatController_update.text);
              Navigator.of(context).pop();
            }
          },
          child: Text("Update"),
        ),
      ],
    ),
  ),
  actions: <Widget>[
    new MaterialButton(
      color: Colors.blueGrey,
      height: 30,
      minWidth: 20,
      onPressed: () {
        Navigator.of(context).pop();
      },
```

```
        child: Text("Close"),
      ),
    ],
  );
}

void show_all_data() async {
  Response response;
  response = await dio.post(
    url_all_data,
  );

  all_data = response.data;
}

void create_data(String name, String bank, String alamat) async {
  Response response;
  response = await dio.post(
    url_create_data,
    queryParameters: {'name': name, 'bank': bank, 'alamat': alamat},
  );

  nameController.text = "";
  bankController.text = "";
  alamatController.text = "";
}

void delete_data(int id) async {
  Response response;
  response = await dio.post(
    url_delete_data,
    queryParameters: {'id': id},
  );
}

void update_data(String id, String name, String bank, String alamat) async {
  Response response;
  response = await dio.post(
    url_update_data,
    queryParameters: {'id': id, 'name': name, 'bank': bank, 'alamat': alamat},
  );

  idController_update.text = "";
  nameController_update.text = "";
  bankController_update.text = "";
```

```
  alamatController_update.text = "";
}
```