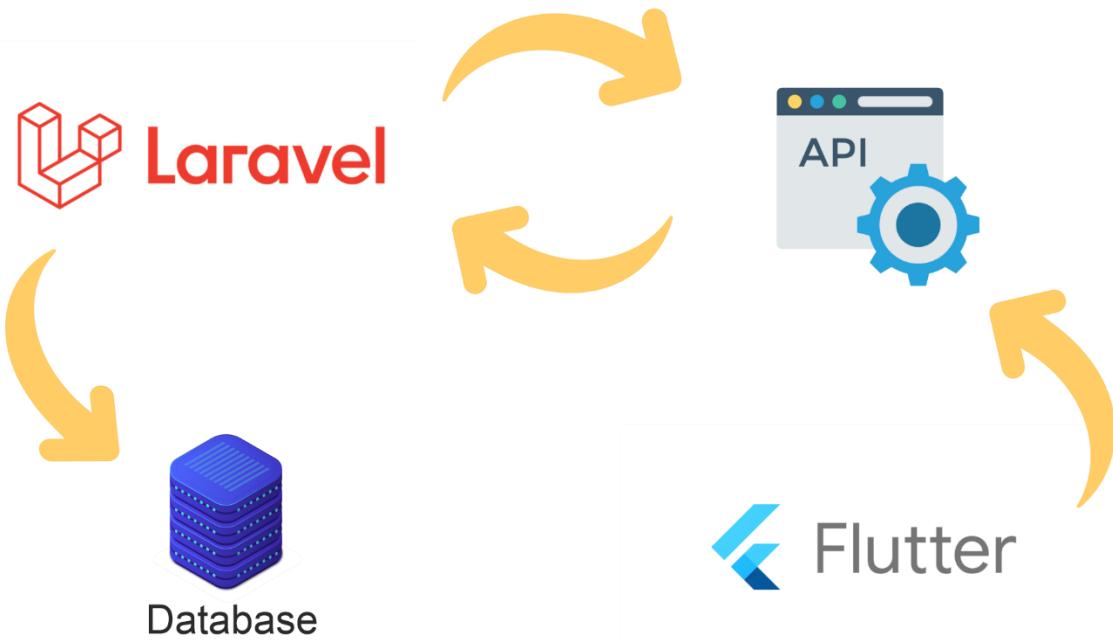


# Modul CRUD Database Flutter

## Menggunakan REST API Pada Laravel



Oleh :

~ Politeknik Negeri Malang ~

## DAFTAR ISI

DAFTAR ISI.....	1
BAB I PERSYARATAN .....	2
A. TUJUAN .....	2
B. MANFAAT.....	2
C. SOFTWARE.....	2
D. PERTANYAAN UMUM.....	2
BAB II PEMASANGAN APLIKASI.....	3
A. Instalasi XAMPP.....	3
B. Instalasi Composer .....	5
C. Instalasi Node JS .....	7
D. Instalasi Postman.....	9
E. Instalasi Visual Studio Code .....	9
F. Instalasi Android Studio .....	11
G. Instalasi Laravel .....	15
H. Instalasi Flutter.....	16
BAB III PEMBUATAN API LARAVEL DAN FLUTTER.....	18
A. Pembuatan Framework Laravel .....	18
B. Pembuatan REST API di Laravel .....	19
C. Pengujian REST API.....	25
D. Pembuatan Proyek Flutter .....	30
E. Pengujian Flutter.....	37
BAB IV LAMPIRAN CODE PROGRAM.....	42
Lampiran 1 .....	42

## 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 RESTAPI 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	Applikasi	Link Download
1	XAMPP	<a href="https://www.apachefriends.org/download.html">https://www.apachefriends.org/download.html</a>
2	Composer	<a href="https://getcomposer.org/download/">https://getcomposer.org/download/</a>
3	Node JS	<a href="https://nodejs.org/en/download">https://nodejs.org/en/download</a>
4	Postman	<a href="https://www.postman.com/downloads/">https://www.postman.com/downloads/</a>
5	Visual Studio Code	<a href="https://code.visualstudio.com/download">https://code.visualstudio.com/download</a>
6	Android Studio	<a href="https://developer.android.com/studio">https://developer.android.com/studio</a>
7	Flutter	<a href="https://docs.flutter.dev/get-started/install/windows">https://docs.flutter.dev/get-started/install/windows</a>

#### **D. PERTANYAAN UMUM**

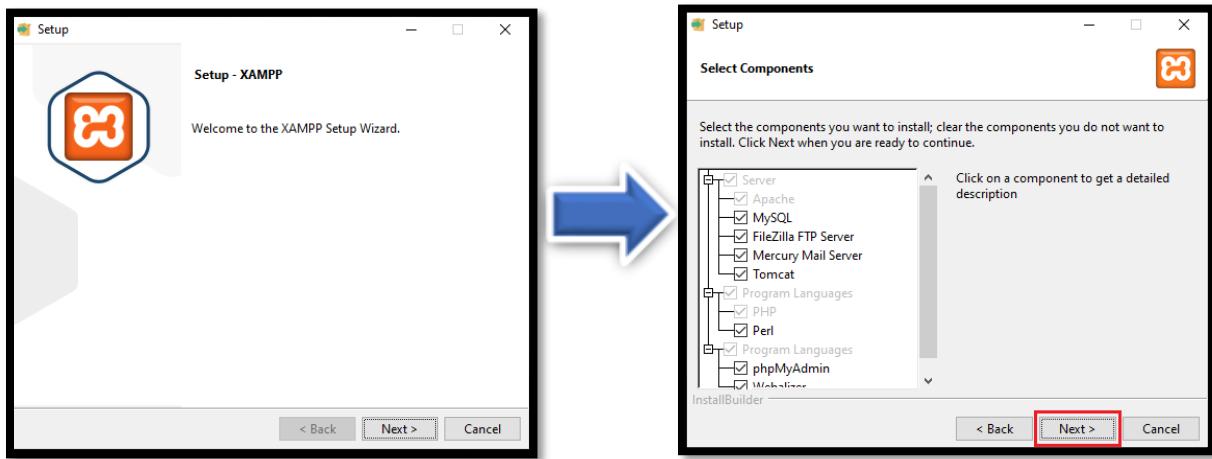
NO	PERMASALAHAN
1	Mengapa harus menggunakan API ?  Dengan menggunakan API akan membuat keamanan database tetap terjaga, karena kita hanya dapat menambah maupun mengubah isi dari database berdasarkan lingkup yang telah kita buat dalam fungsi API tersebut.
2	Mengapa terjadi error ketika saya menjalankan 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

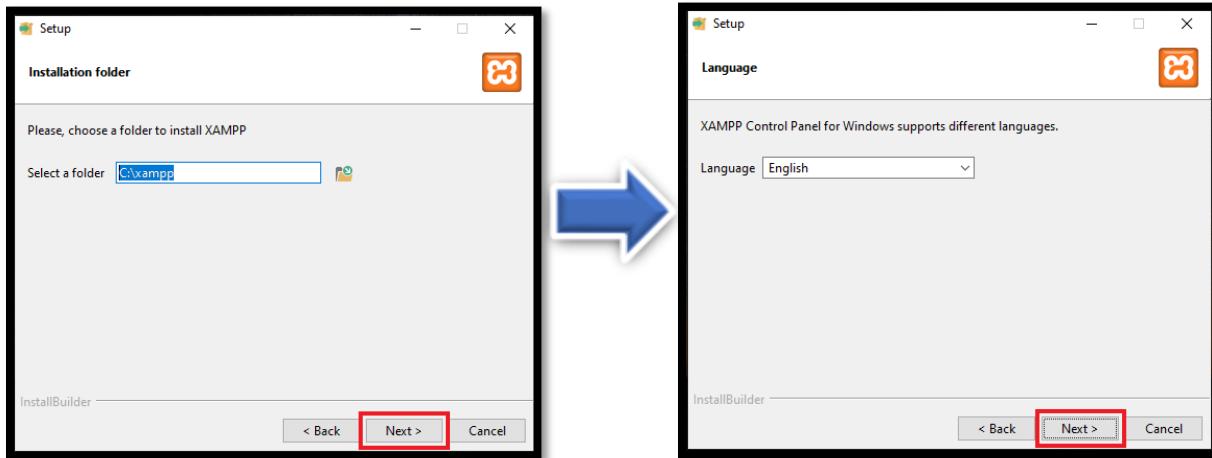
### PEMASANGAN APLIKASI

#### A. Instalasi XAMPP

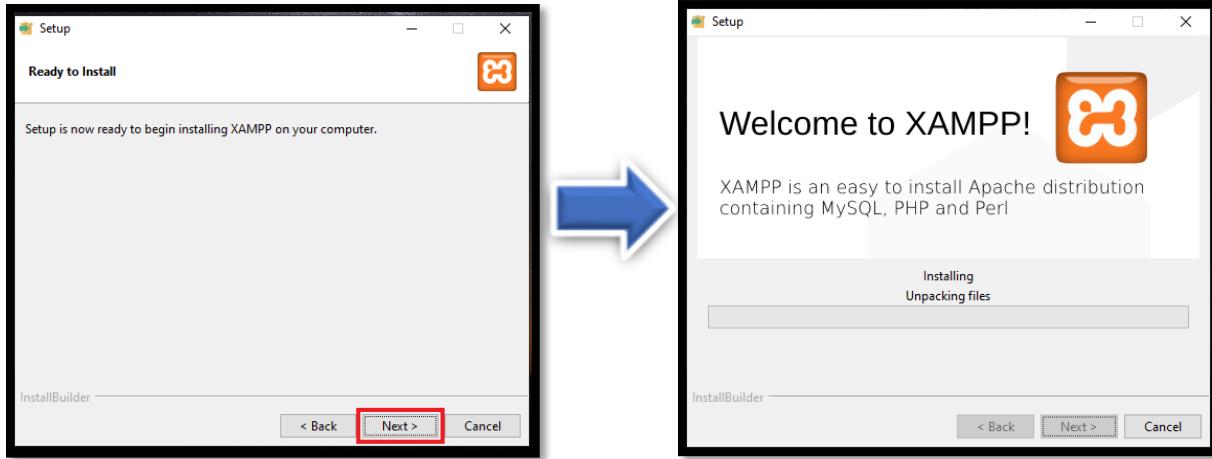
1. Buka file XAMPP yang telah didownload sebelumnya dengan cara klik 2x pada file xampp.exe kemudian klik “Next” dan pada bagian Select Component biarkan secara default dan langsung klik “Next”.



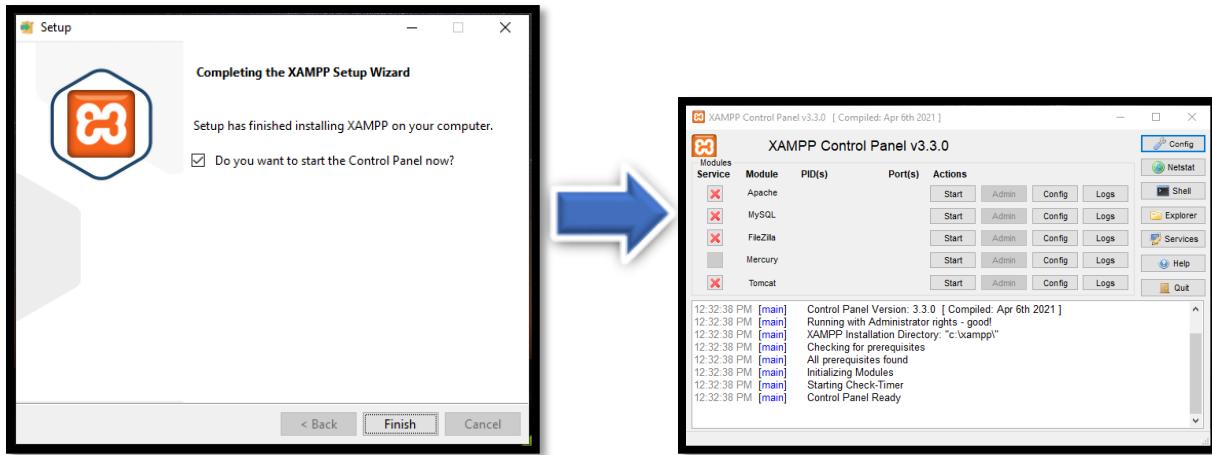
2. Biarkan lokasi folder tetap dengan pengaturan awal dan klik “Next”, pilih Bahasa yang ingin digunakan kemudian klik “Next”.



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

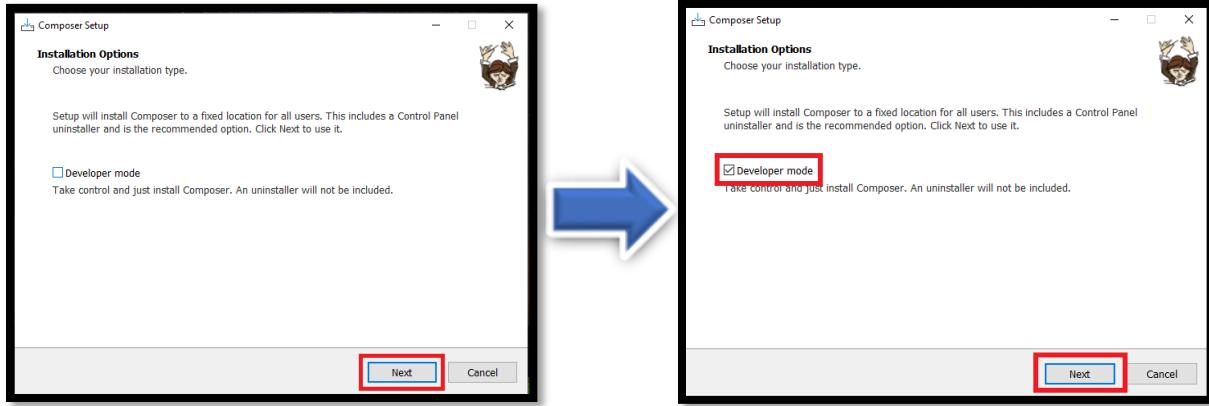


4. Setelah proses instalasi selesai klik “Finish” dan jendela panel xampp akan muncul dilayar anda, untuk dapat menjalankan project ini klik “Start” pada module Apache dan MySQL.

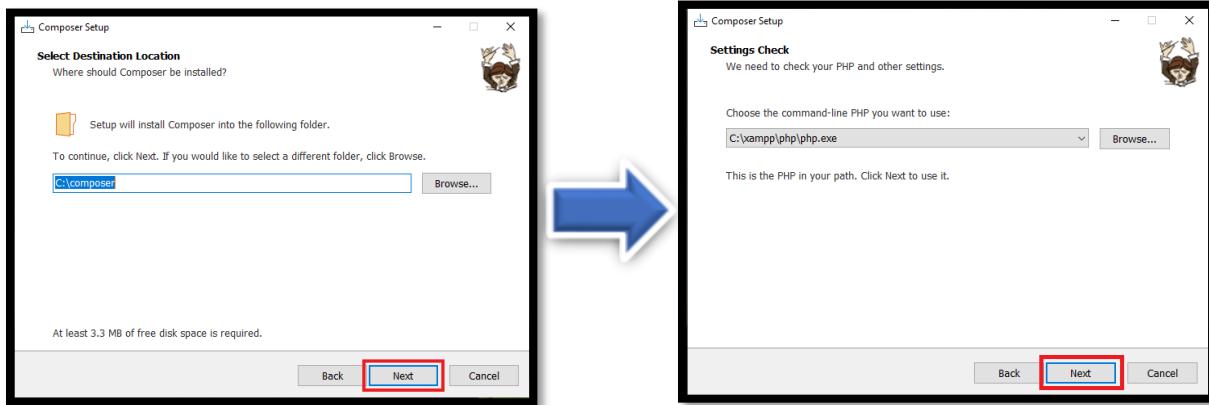


## B. Instalasi Composer

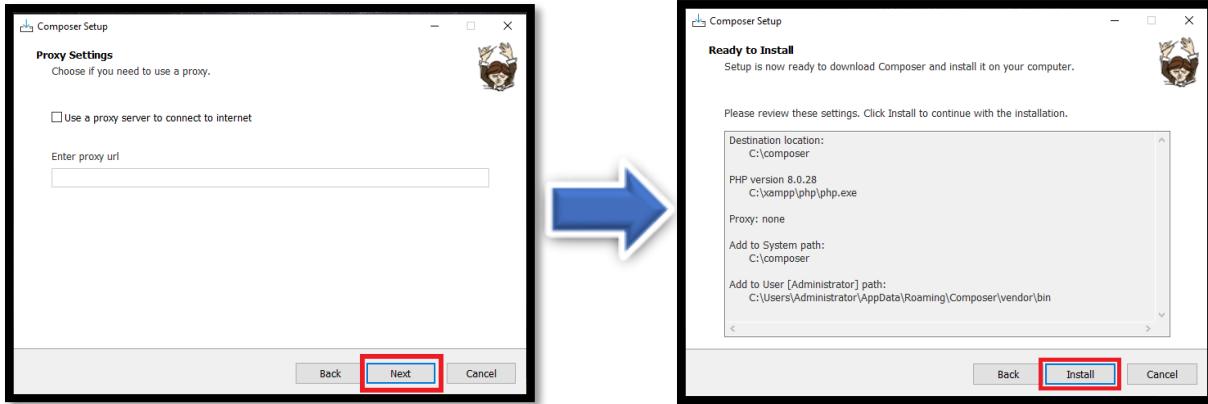
1. Buka file Composer yang telah didownload sebelumnya dengan cara klik 2x pada file composer.exe kemudian centang pada bagian “Developer Mode” dan klik “Next”.



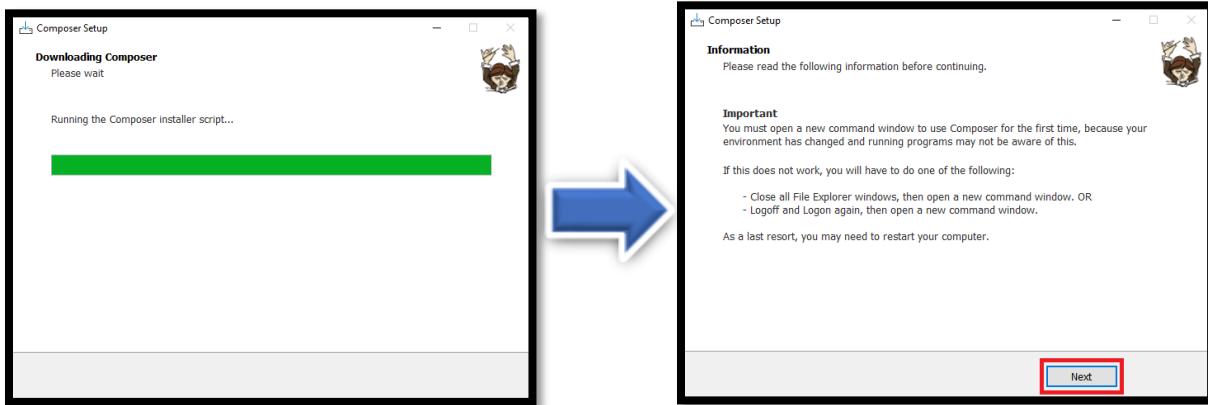
2. Biarkan lokasi folder tetap dengan pengaturan awal dan klik “Next”, lalu pada bagian Setting Check apabila file php.exe sudah terdeteksi langsung klik “Next” apabila belum silahkan klik “Browse” untuk mencari lokasi file php.exe anda.



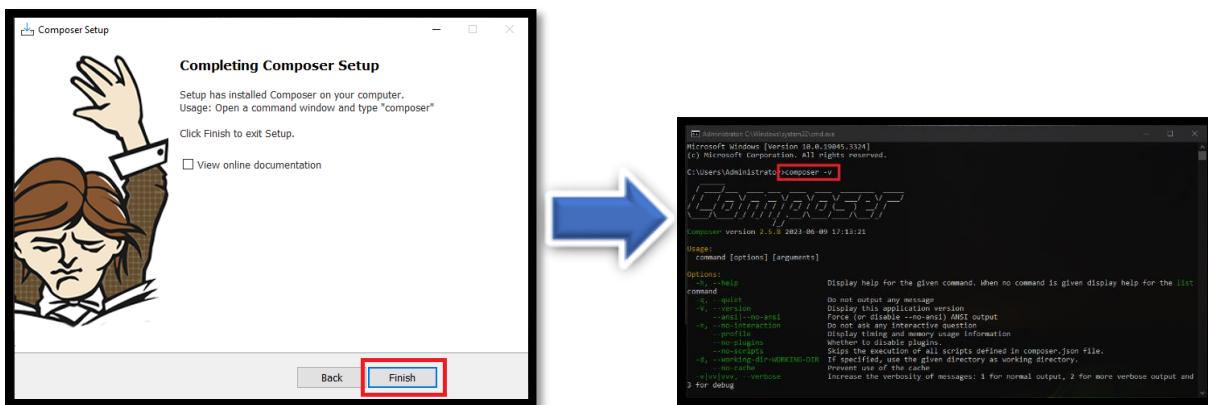
3. Klik “Next” kemudian klik “Install”.



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

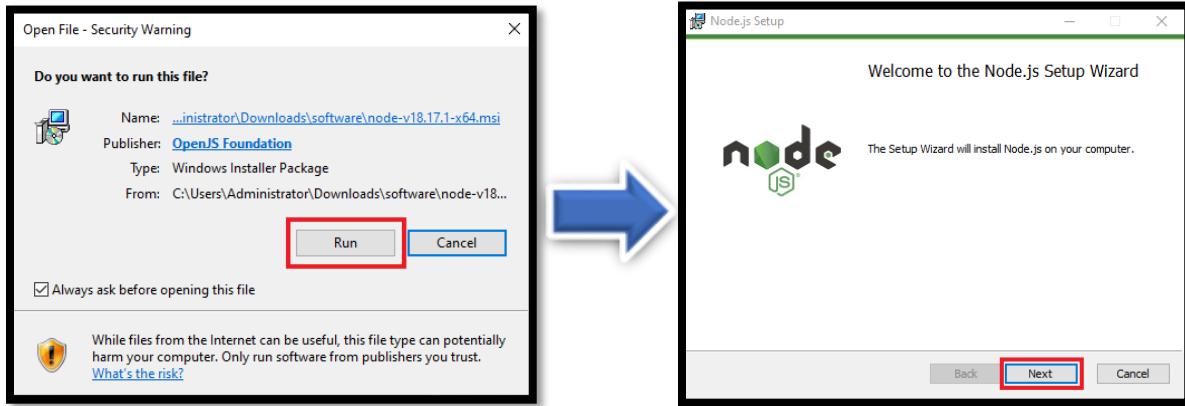


5. Klik “Finish” dan buka Terminal kemudian ketikkan perintah “composer -v” tanpa tanda kutip dan amati output yang dihasilkan, jika output yang dikeluarkan merupakan rincian versi dari composer maka pemasangan composer telah berhasil dilakukan.

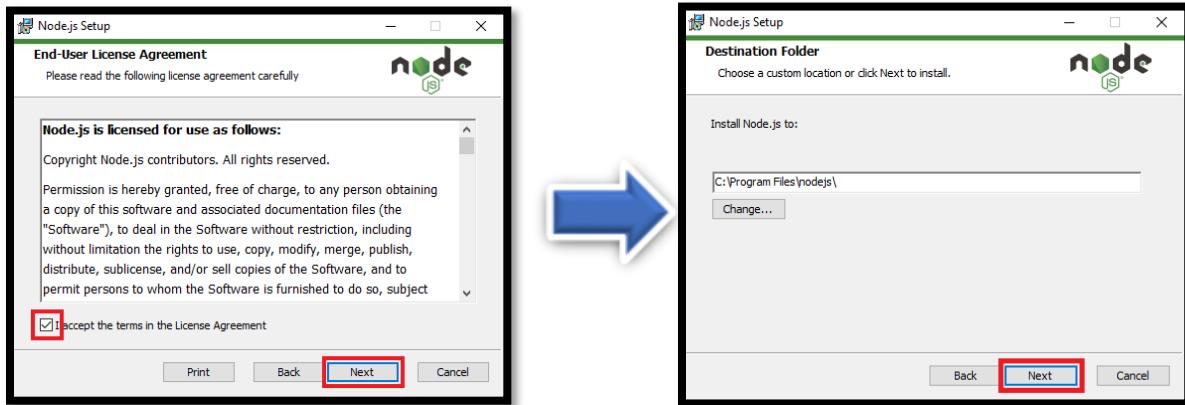


## C. Instalasi Node JS

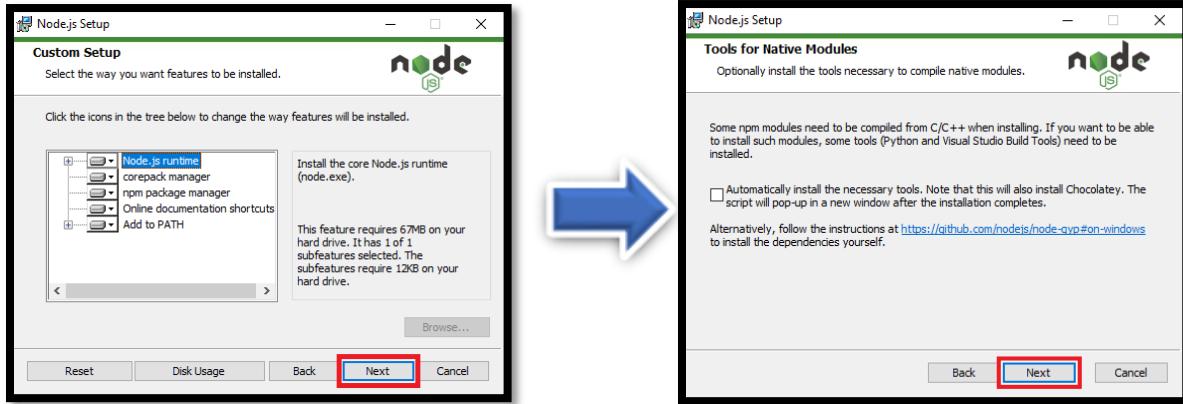
1. Buka file Node JS yang telah didownload sebelumnya dengan cara klik 2x pada file node.exe, apabila terdapat peringatan keamanan tetap jalankan dengan mengklik tombol “Run” kemudian klik “Next”.



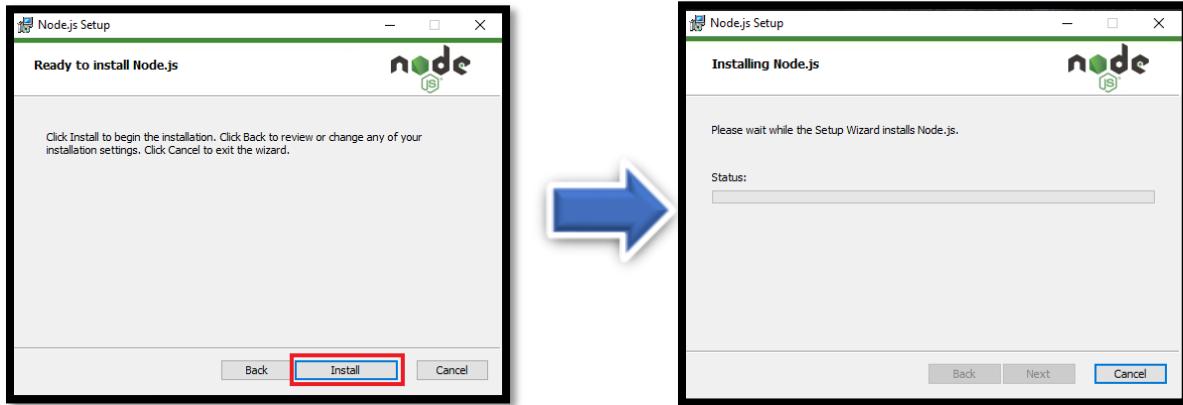
2. Centang box “I Accept ...” dan klik “Next”, kemudian pada bagian Destination Folder biarkan menggunakan lokasi yang telah disediakan lalu klik “Next”.



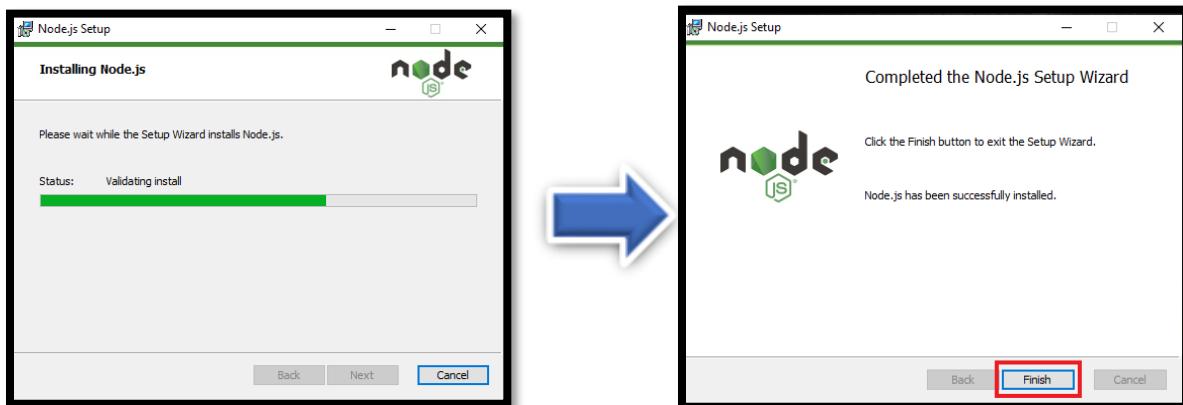
3. Gunakan pengaturan awal dan jangan diubah kemudian klik “Next” 2x.



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

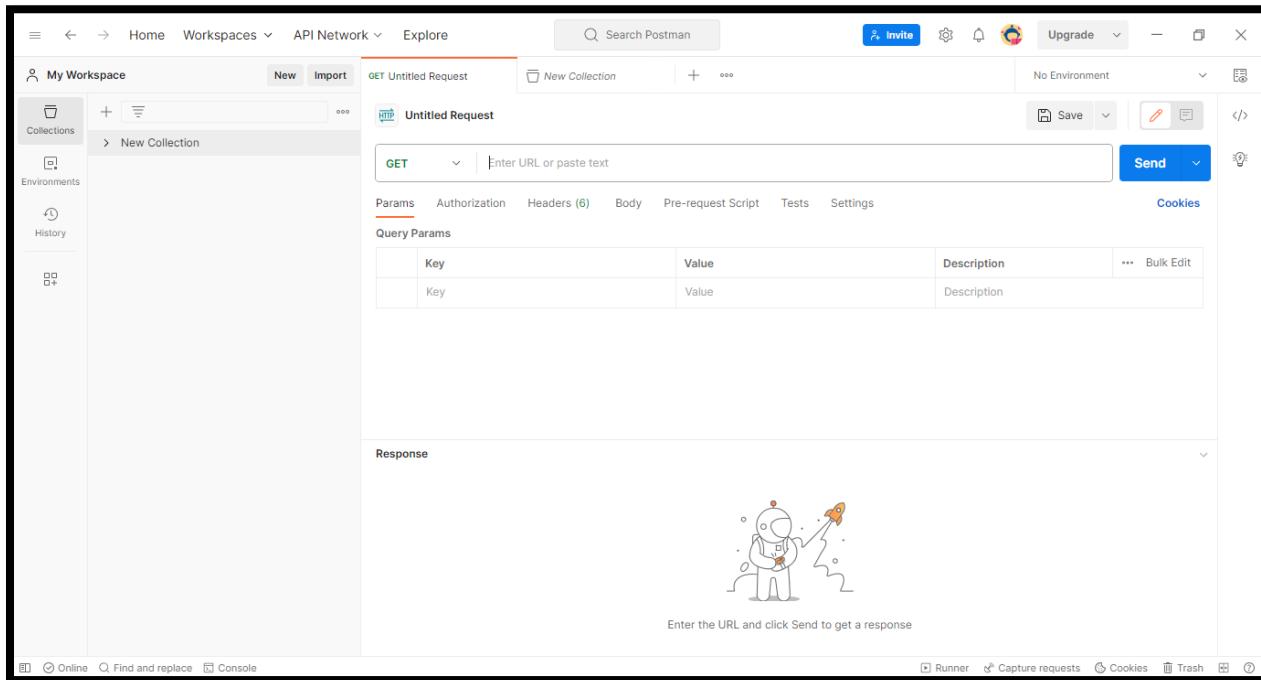


5. Setelah proses pemasangan selesai, klik “Finish”.



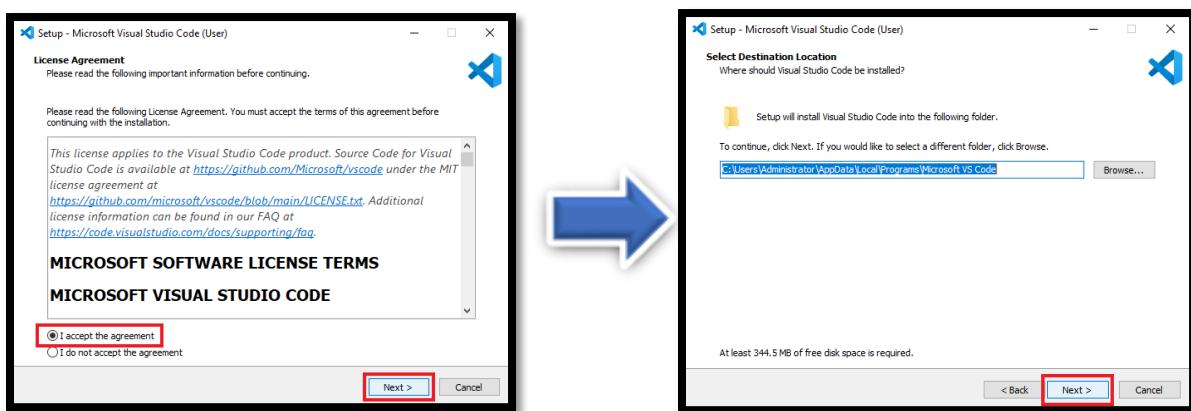
## D. Instalasi Postman

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.

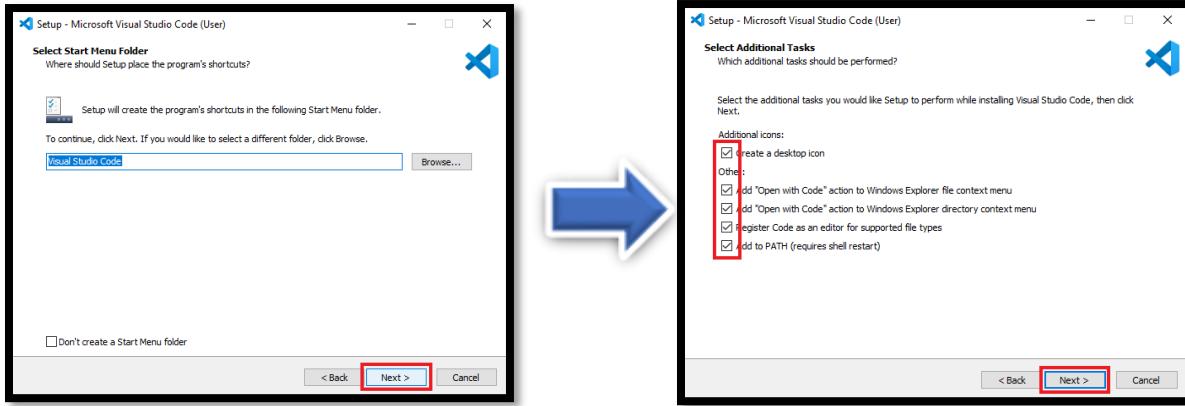


## E. Instalasi Visual Studio Code

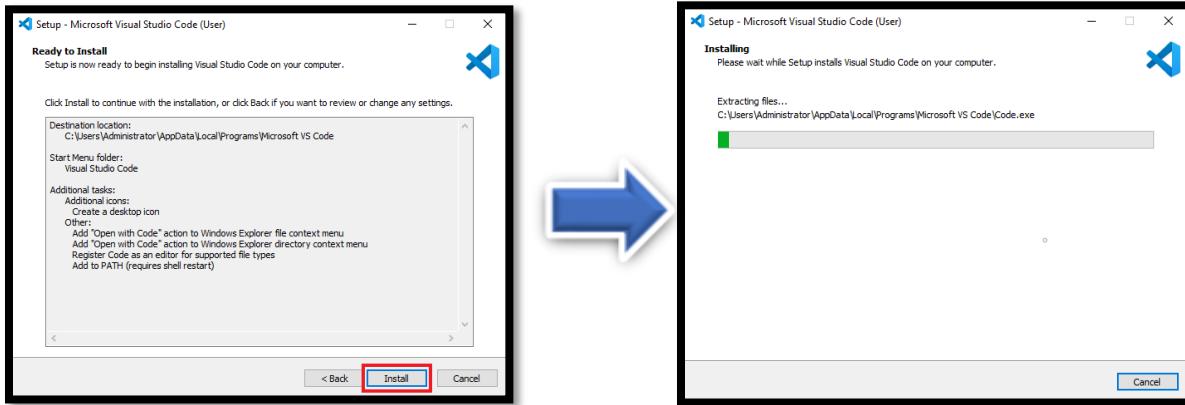
1. Buka file installer yang telah didownload sebelumnya, kemudian pilih “I Accept ...” dan klik “Next”. Lalu langsung klik “Next” pada bagian Select Destination Location.



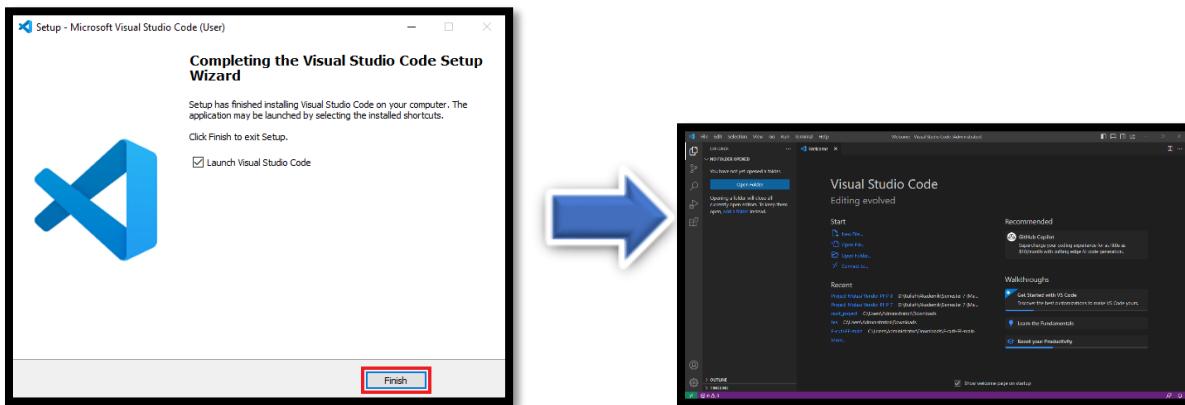
2. Klik “Next” kemudian centang semua opsi pilihan dan klik “Next” lagi.



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

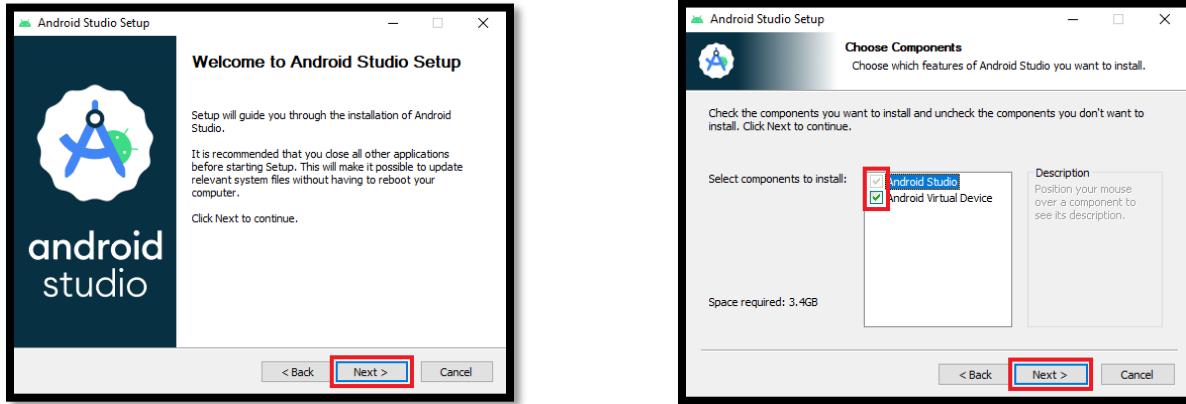


4. Setelah proses pemasangan selesai, klik “Finish” dan akan terbuka jendela baru yang merupakan tampilan dari visual studio code.

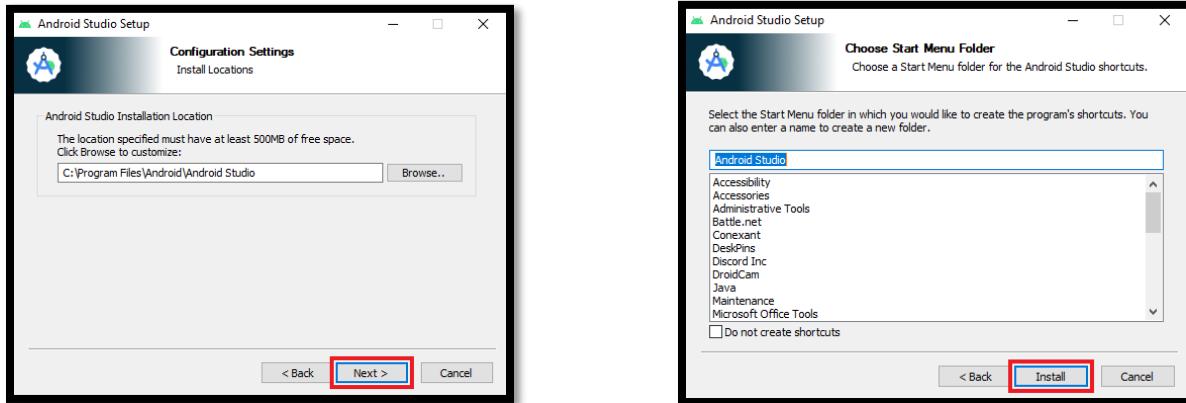


## F. Instalasi Android Studio

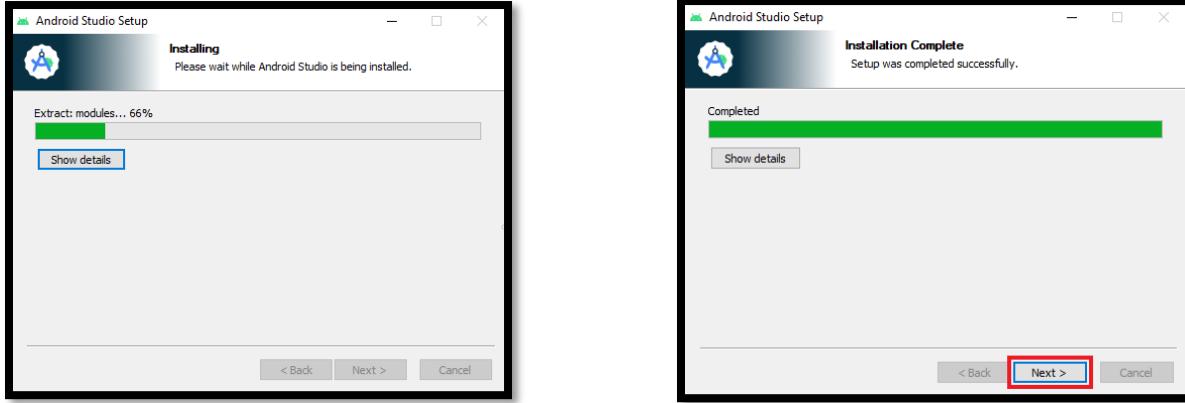
1. Buka file Android Studio yang telah didownload sebelumnya dengan cara klik 2x pada file android-studio.exe, kemudian klik “Next” dan centang pada bagian “Android Virtual Device” lalu klik “Next”.



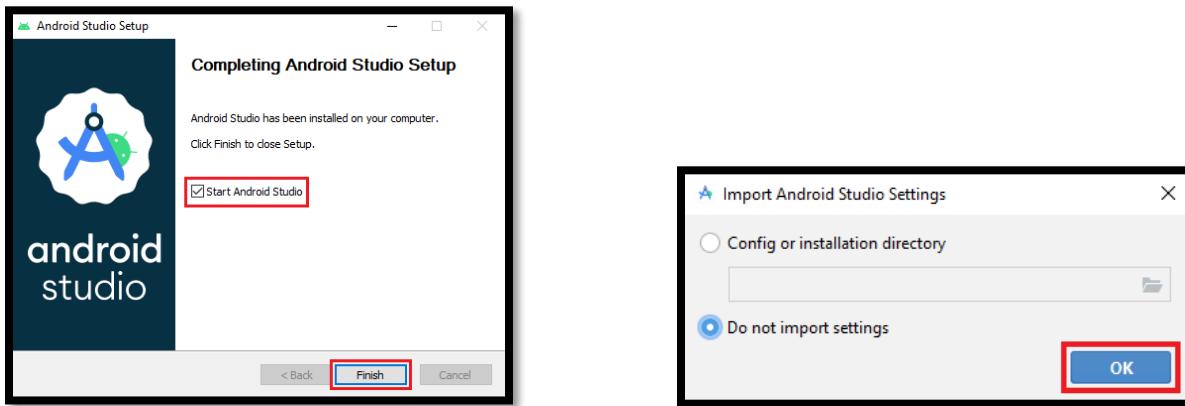
2. Pilih lokasi untuk tempat menginstall kemudian klik “Next” dan klik “Install”.



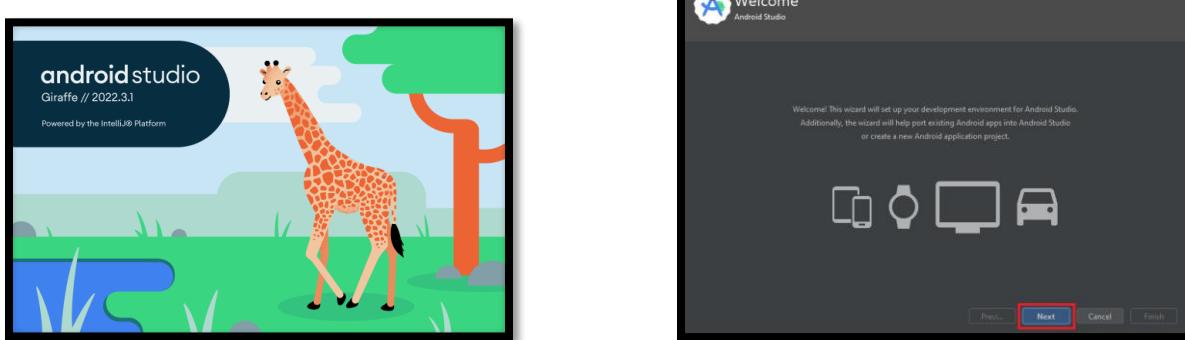
3. Tunggu hingga proses pemasangan selesai, kemudian klik “Next”.



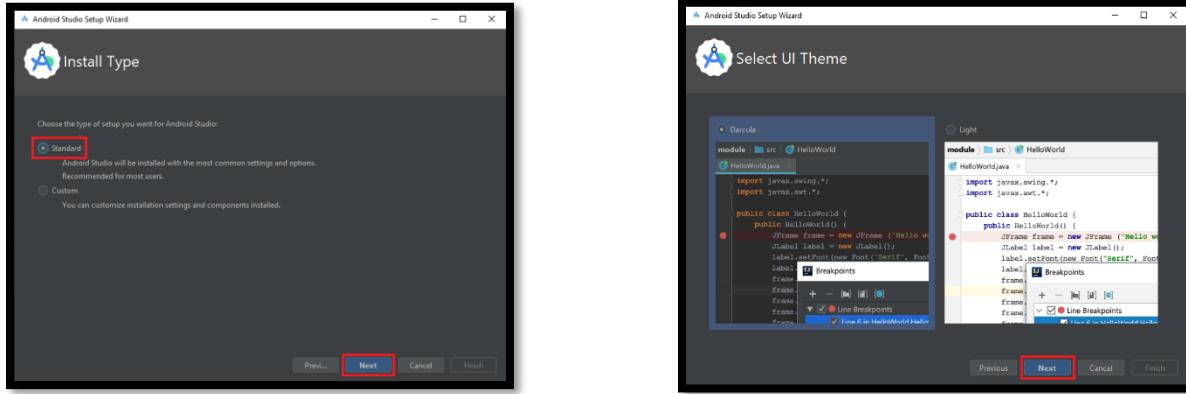
4. Centang pada bagian “Start Android Studio” dan klik “Finish”.



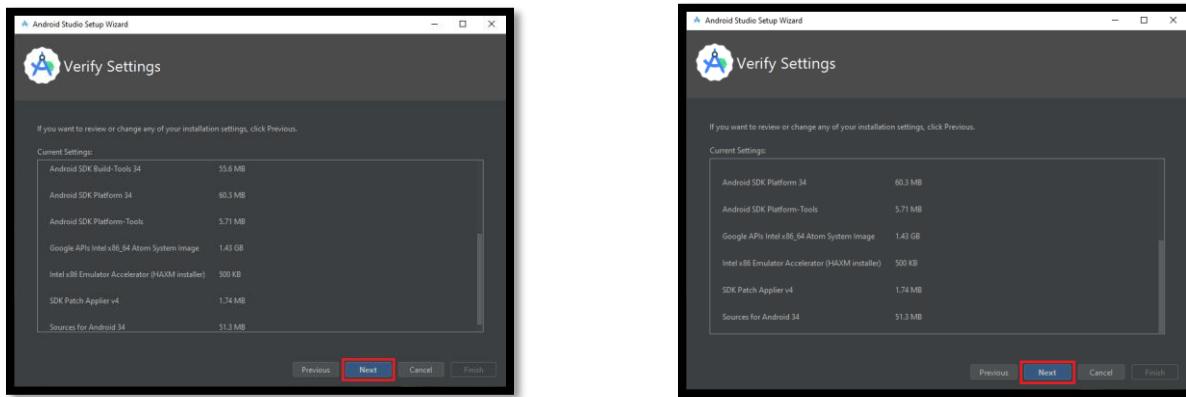
5. Setelah proses install berhasil, aplikasi android studio akan membuka jendela baru untuk pengaturan awal, klik “Next” pada jendela yang telah muncul.



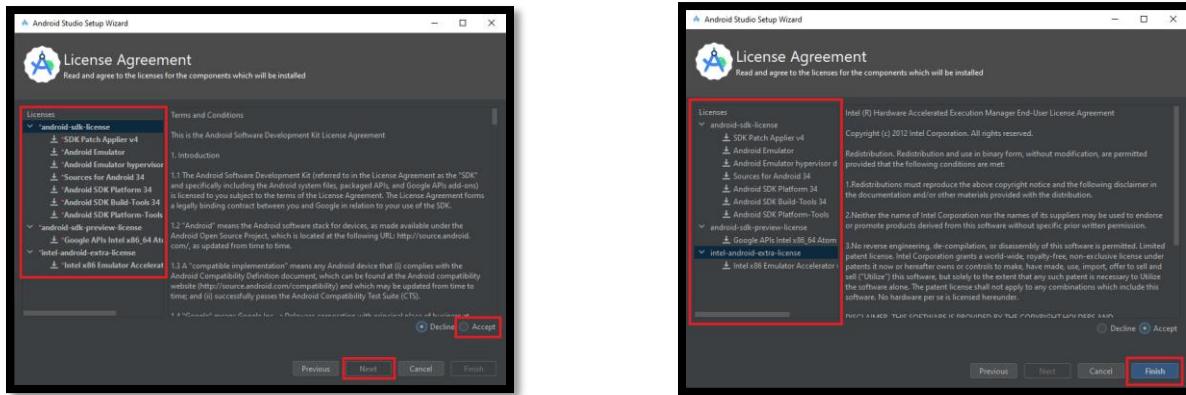
6. Pilih tipe “Standard” dan klik “Next”, kemudian pilih tema yang anda inginkan kemudian klik “Next”.



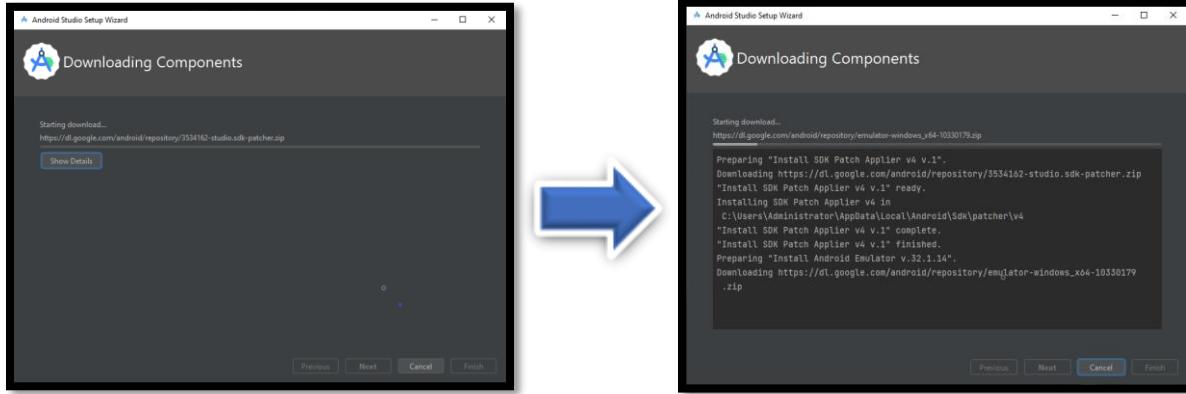
7. Terdapat rincian dari paket-paket yang akan terdownload lagi secara otomatis nantinya, klik “Next”.



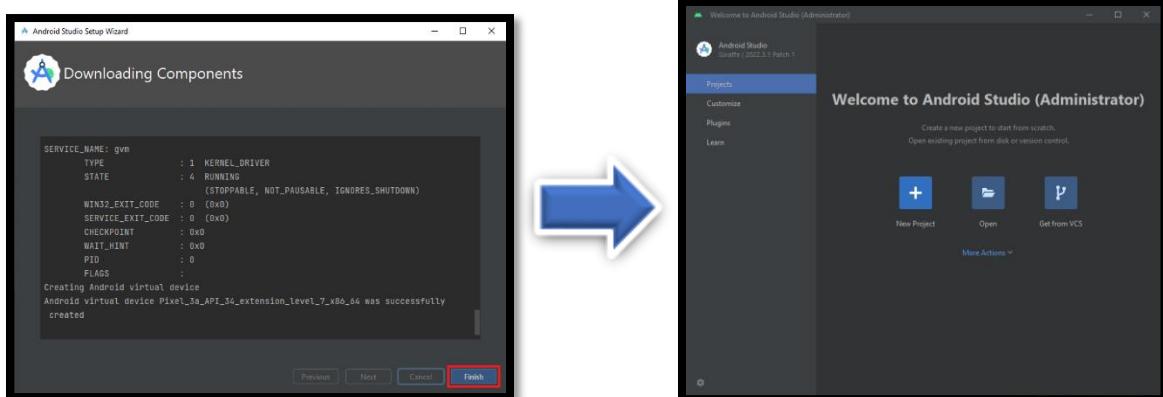
8. Pada setiap lisensi paket disebelah kiri jendela, pilih “Accept” kemudian klik “Finish”.



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

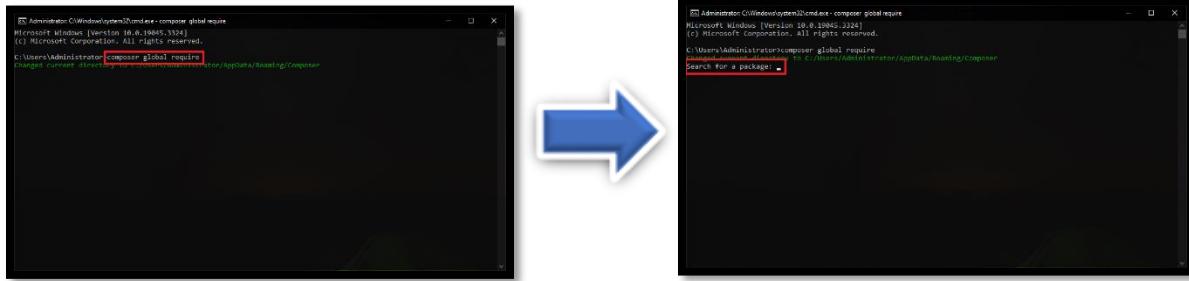


10. Setelah proses download selesai klik “Finish” dan android studio telah siap untuk digunakan.



## G. Instalasi Laravel

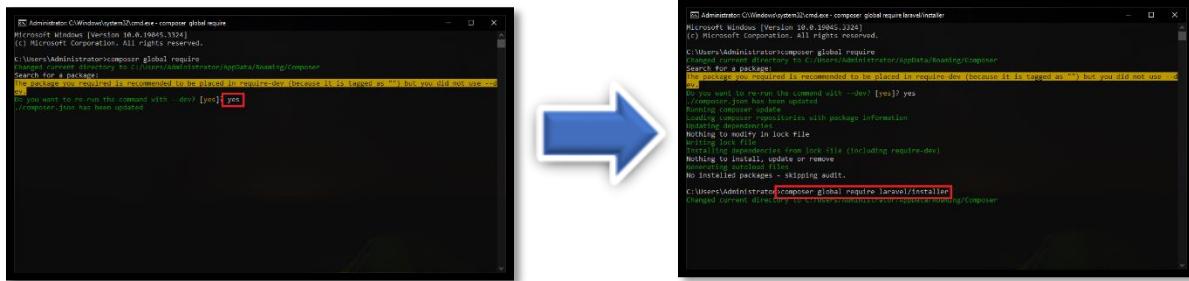
1. Buka Terminal atau Command Prompt di perangkat anda dan ketikkan perintah “composer global require” tanpa tanda kutip kemudian klik enter, apabila muncul baris yang berisi “Search for a package...” langsung saja klik enter lagi.



```
Administrator: C:\Windows\system32\cmd.exe - composer global require
Microsoft Windows [Version 10.0.19045.324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>composer global require
Changed current directory to C:\Users\Administrator\AppData\Roaming\Composer
Search for a package...
```

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.



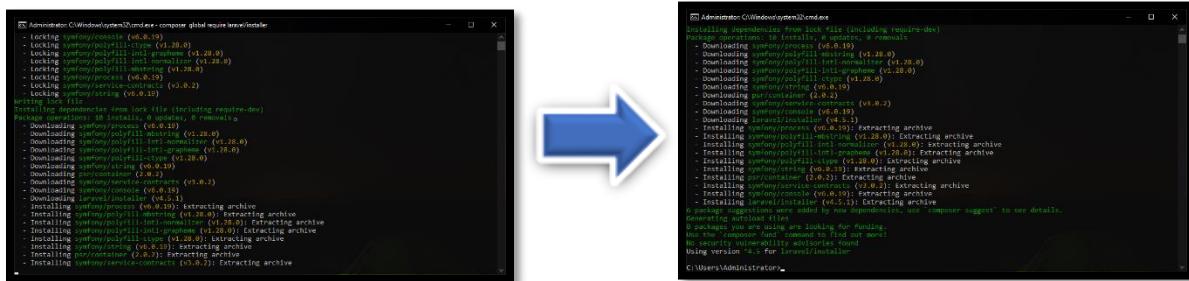
```
Administrator: C:\Windows\system32\cmd.exe - composer global require laravel/installer
Microsoft Windows [Version 10.0.19045.324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>composer global require
Search for a package...
The package you resolved is recommended to be placed in require-dev (because it is tagged as "") but you did not use --dev.
Do you want to re-run this command with --dev? [yes] yes
/composer.json has been updated

Administrator: C:\Windows\system32\cmd.exe - composer global require laravel/installer
Microsoft Windows [Version 10.0.19045.324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>composer global require laravel/installer
Changed current directory to C:\Users\Administrator\AppData\Roaming\Composer
Search for a package...
```

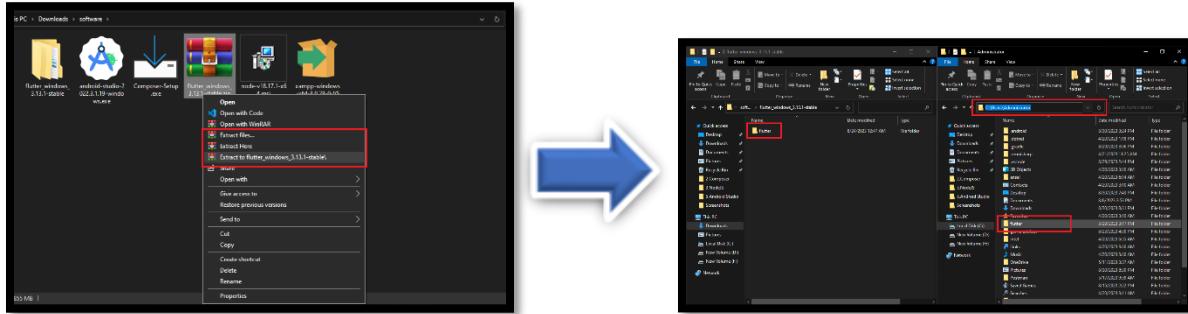
3. Pastikan koneksi internet anda lancar dan stabil, tunggu hingga proses pemasangan selesai.



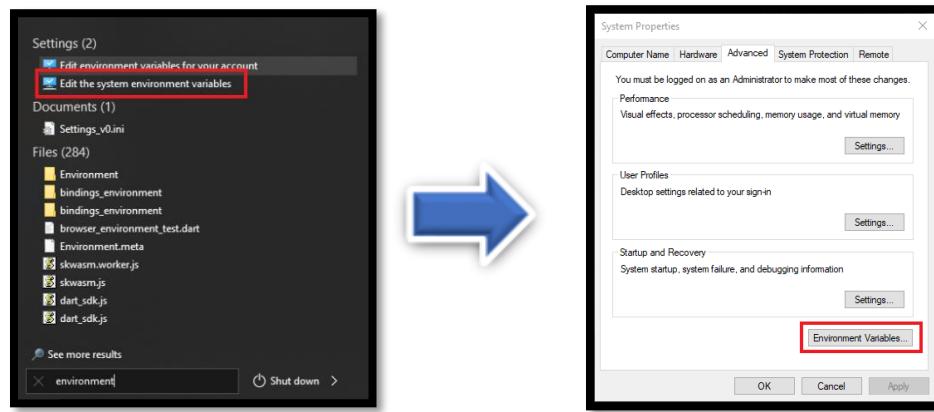
```
Administrator: C:\Windows\system32\cmd.exe - composer global require laravel/installer
- Locking composer (v6.2.10)
- Locking symfony/polyfill-ctype (v1.28.0)
- Locking symfony/polyfill-intl-icu (v1.28.0)
- Locking symfony/polyfill-intl-normalizer (v1.28.0)
- Locking symfony/polyfill-mbstring (v1.28.0)
- Locking symfony/process (v6.2.10)
- Locking symfony/service-contracts (v3.0.2)
- Locking symfony/translation (v6.2.10)
writing lock file
read composer.lock (including require-dev)
Package "laravel/installer" is already installed, skipping.
Package "laravel/tinker" is already installed, skipping.
- Downloading symfony/polyfill-ctype (v1.28.0)
- Downloading symfony/polyfill-intl-icu (v1.28.0)
- Downloading symfony/polyfill-intl-normalizer (v1.28.0)
- Downloading symfony/polyfill-mbstring (v1.28.0)
- Downloading symfony/process (v6.2.10)
- Downloading symfony/service-contracts (v3.0.2)
- Downloading symfony/translation (v6.2.10)
writing lock file
read composer.lock (including require-dev)
Package "laravel/installer" is already installed, skipping.
Package "laravel/tinker" is already installed, skipping.
- Downloading symfony/polyfill-ctype (v1.28.0)
- Downloading symfony/polyfill-intl-icu (v1.28.0)
- Downloading symfony/polyfill-intl-normalizer (v1.28.0)
- Downloading symfony/polyfill-mbstring (v1.28.0)
- Downloading symfony/process (v6.2.10)
- Downloading symfony/service-contracts (v3.0.2)
- Downloading symfony/translation (v6.2.10)
- Installing laravel/installer (v6.5.1)
- Installing laravel/tinker (v2.6.19)
- Extracting archive
Installing symfony/polyfill-ctype (v1.28.0): Extracting archive
Installing symfony/polyfill-intl-icu (v1.28.0): Extracting archive
Installing symfony/polyfill-intl-normalizer (v1.28.0): Extracting archive
Installing symfony/polyfill-mbstring (v1.28.0): Extracting archive
Installing symfony/process (v6.2.10): Extracting archive
Installing symfony/service-contracts (v3.0.2): Extracting archive
Installing symfony/translation (v6.2.10): Extracting archive
- Installing laravel/installer (v6.5.1): Extracting archive
Generating optimized autoload files
No packages you're using are looking for funding.
No security vulnerabilities were found.
Using version 6.5.1 for laravel/installer
C:\Users\Administrator>
```

## H. Instalasi Flutter

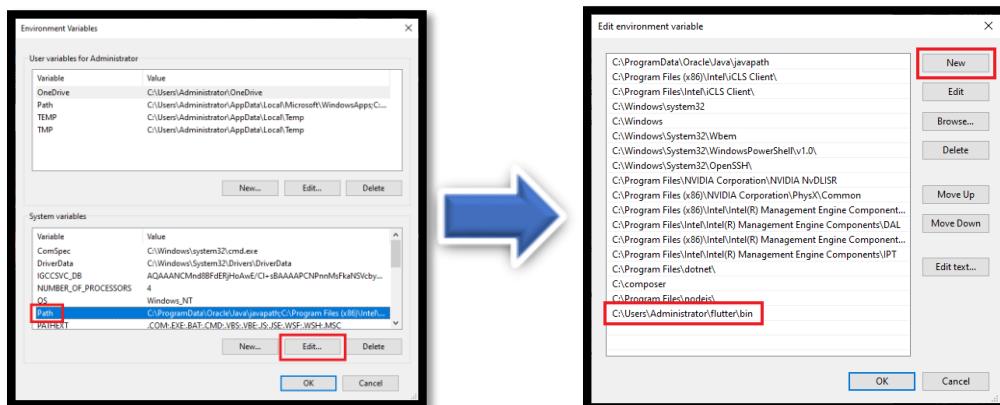
- Klik kanan pada file flutter yang telah didownload sebelumnya dan extrak file tersebut terlebih dahulu. Kemudian pindahkan folder bernama “flutter” hasil dari extrak tadi ke direktori “C:\Users\(*nama\_user*)”.



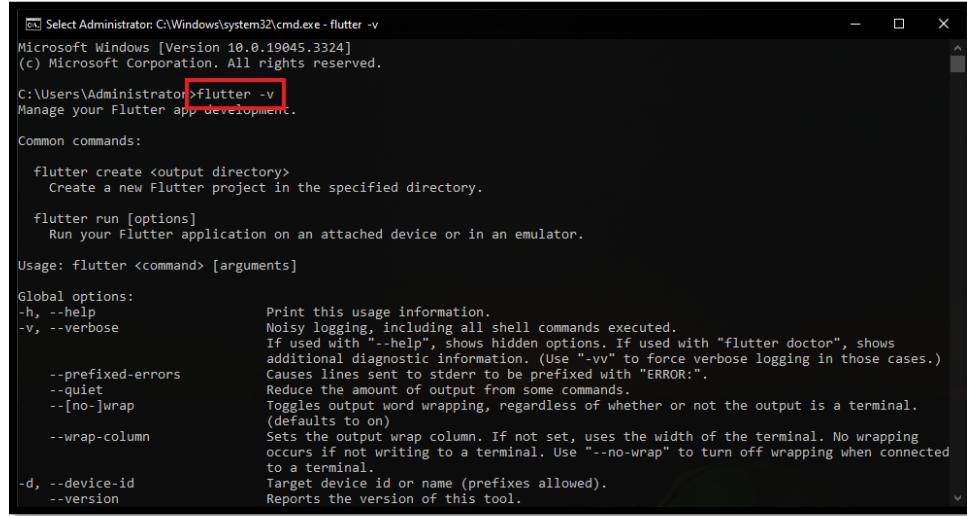
- Setelah itu cari pengaturan “environment” pada fitur pencarian di perangkat anda. Lalu klik “Environment Variables...”.



- 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.



```

Select Administrator: C:\Windows\system32\cmd.exe - flutter -v
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>flutter -v
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

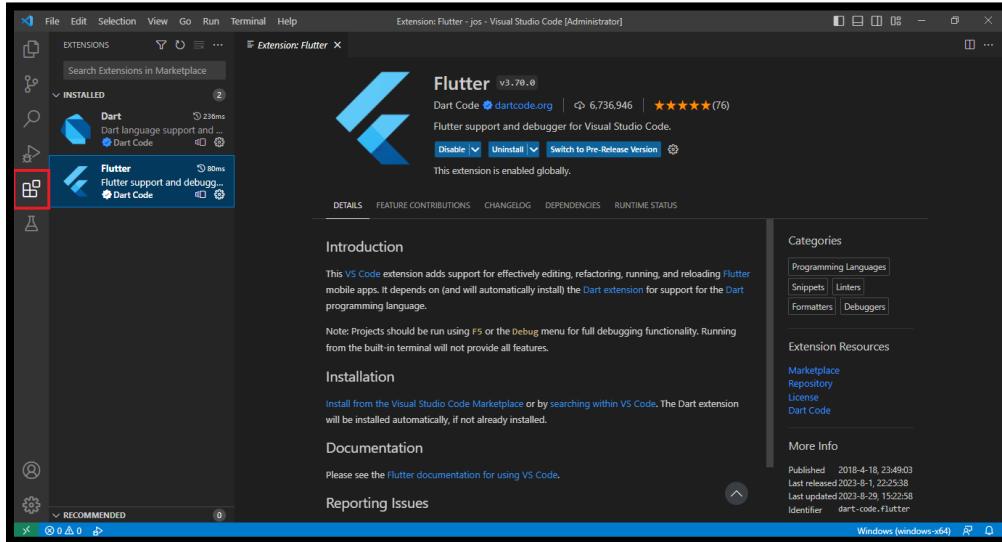
  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [<arguments>]

Global options:
  -h, --help          Print this usage information.
  -v, --verbose       Noisy logging, including all shell commands executed.
                      If used with "-help", shows hidden options. If used with "flutter doctor", shows
                      additional diagnostic information. (Use "-vv" to force verbose logging in those cases.)
  --prefixed-errors  Causes lines sent to stderr to be prefixed with "ERROR:".
  --quiet            Reduce the amount of output from some commands.
  --[no-]wrap         Toggles output word wrapping, regardless of whether or not the output is a terminal.
                      (defaults to on)
  --wrap-column      Sets the output wrap column. If not set, uses the width of the terminal. No wrapping
                      occurs if not writing to a terminal. Use "--no-wrap" to turn off wrapping when connected
                      to a terminal.
  -d, --device-id    Target device id or name (prefixes allowed).
  --version          Reports the version of this tool.

```

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)

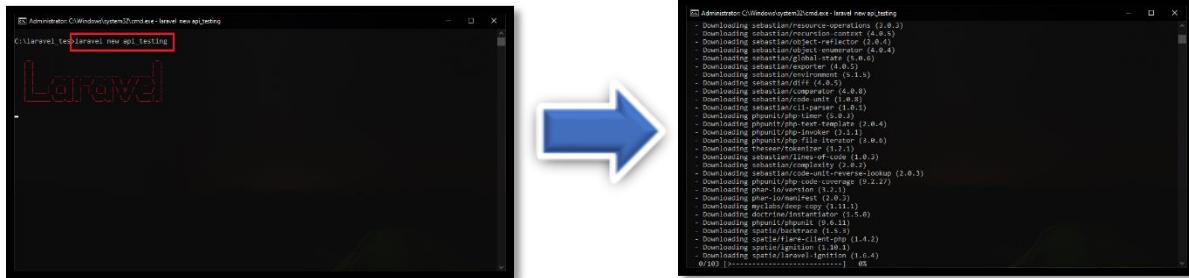


### BAB III

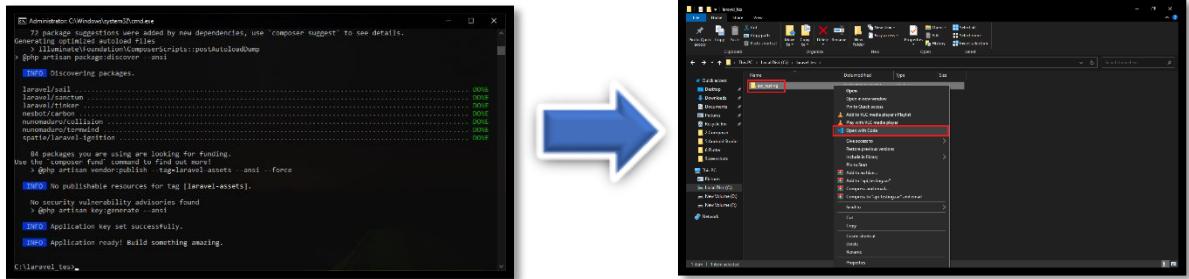
## PEMBUATAN API LARAVEL DAN FLUTTER

#### A. Pembuatan Framework Laravel

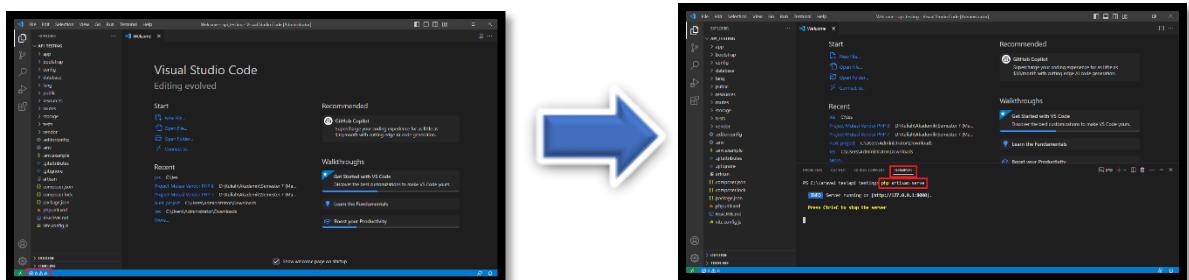
1. Buka Terminal atau Command Prompt di perangkat anda dan ketikkan perintah “laravel new api\_testing” kemudian klik enter dan tunggu hingga proses pembuatan proyek laravel selesai.



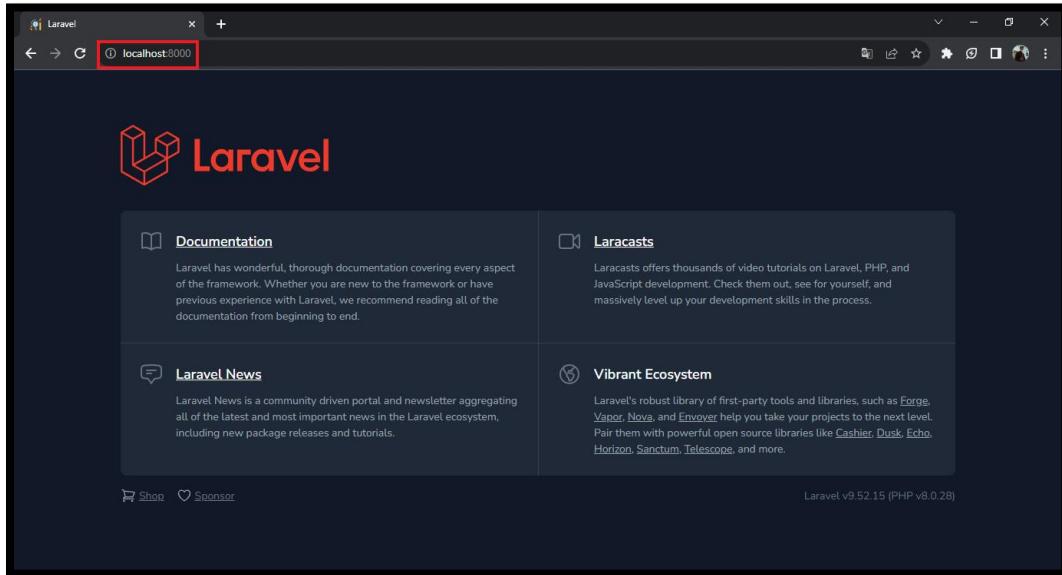
2. Setelah proses loading selesai, buka Windows Explore di perangkat anda dan cari folder dengan nama “api\_testing”, kemudian klik kanan pada folder tersebut dan pilih “Open With Code” untuk membuka proyek tersebut dengan aplikasi Visual Studio Code.



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.

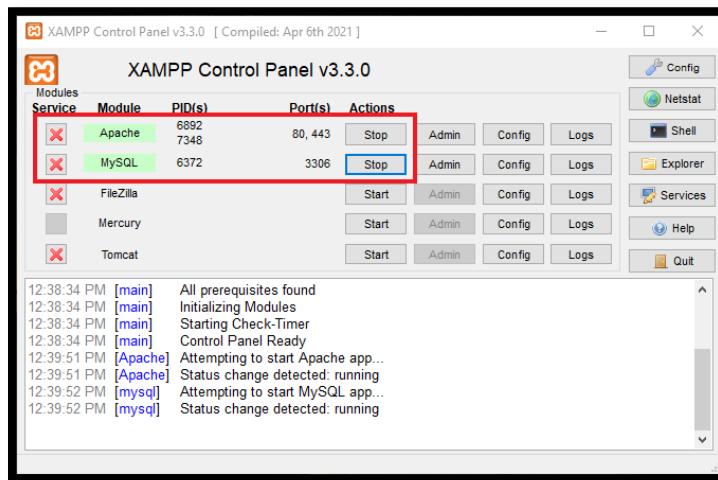


4. Buka browser di perangkat anda dan tuliskan url “localhost:8000” tanpa tanda kutip di browser anda. Apabila tampilan yang dihasilkan sama seperti gambar dibawah ini maka pembuatan proyek laravel baru telah berhasil.

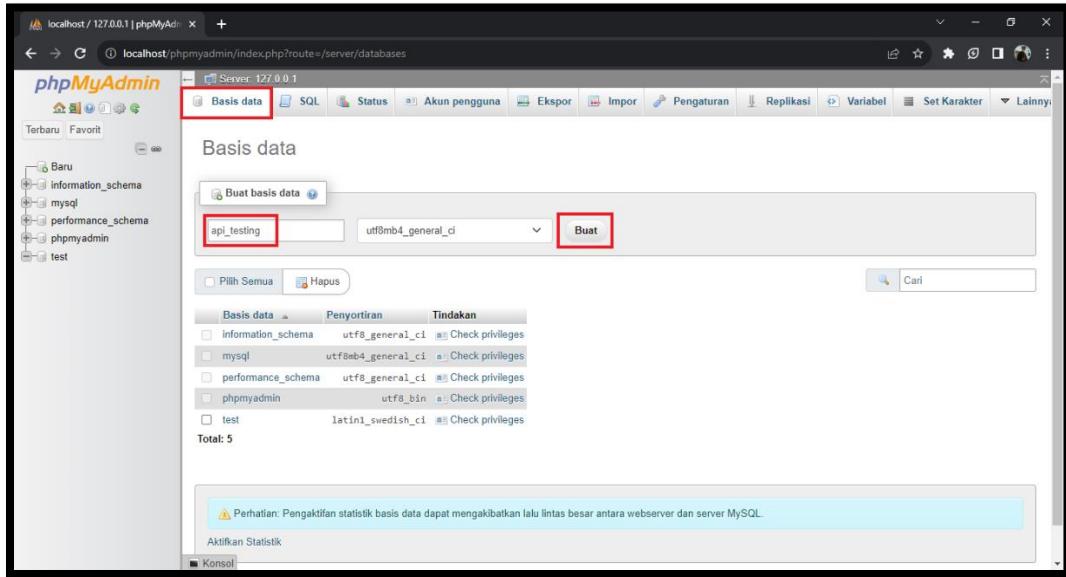


## B. Pembuatan REST API di Laravel

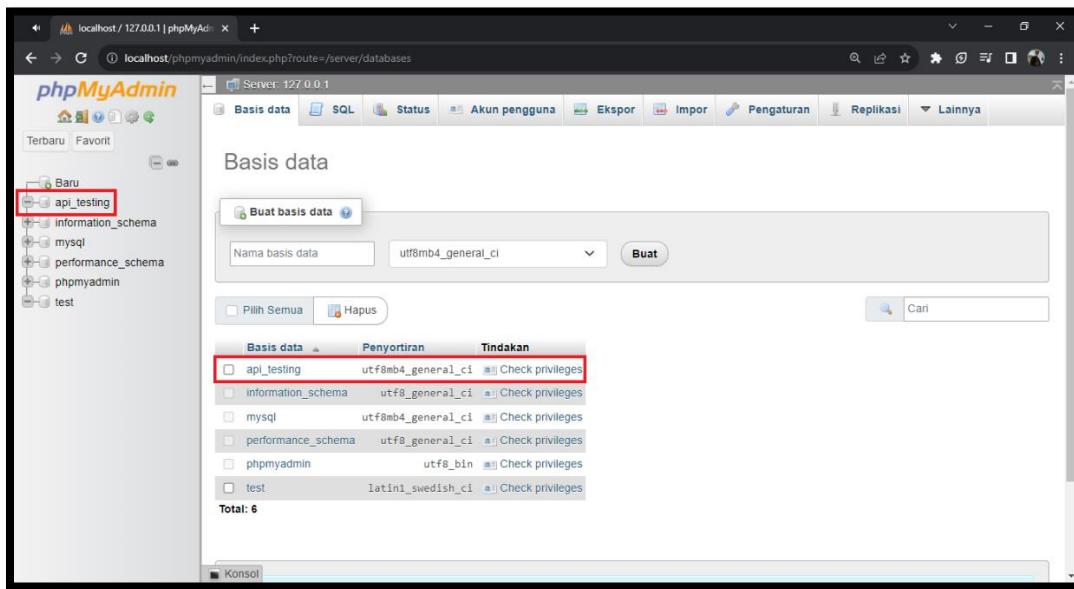
1. Pertama-tama buka aplikasi XAMPP di perangkat anda dan pastikan bahwa module “Apache” dan “MySQL” telah aktif.



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 ketikkan perintah “`php artisan make:migration api_datas`” tanpa tanda kutip dan klik enter untuk membuat file migration yang berfungsi untuk mengatur tabel dalam database.

```
File Edit Selection View Go Run Terminal Help
api_testing - Visual Studio Code [Administrator]
EXPLORER
APPI_TESTING
  > app
  > bootstrap
  > config
  > database
    > factories
    > migrations
      2014_10_12_000000_create_users_table.php
      2014_10_12_100000_create_password_resets_table.php
      2019_08_19_000000_create_failed_jobs_table.php
      2019_12_14_000001_create_personal_details_table.php
      2023_08_31_083929_api_datas.php
    > seeders
    .gitignore
  > lang
  > public
  > resources
  > routes
  > storage
  > tests
  > vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  artisan
> OUTLINE
> TIMELINE
x 0 △ 0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\laravel\tes\api_testing> php artisan make:migration api_datas
[INFO] Migration [C:\laravel\tes\api_testing\database\migrations\2023_08_31_083929_api_datas.php] created successfully.

PS C:\laravel\tes\api_testing>
```

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.

```
File Edit Selection View Go Run Terminal Help
2023_08_31_083929_api_datas.php - api_testing - Visual Studio Code [Administrator]
EXPLORER
APPI_TESTING
  > app
  > bootstrap
  > config
  > database
    > factories
    > migrations
      2014_10_12_000000_create_users_table.php
      2014_10_12_100000_create_password_resets_table.php
      2019_08_19_000000_create_failed_jobs_table.php
      2019_12_14_000001_create_personal_details_table.php
      2023_08_31_083929_api_datas.php
    > seeders
    .gitignore
  > lang
  > public
  > resources
  > routes
  > storage
  > tests
  > vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  artisan
> OUTLINE
> TIMELINE
x 0 △ 0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\laravel\tes\api_testing> php artisan make:migration api_datas
[INFO] Migration [C:\laravel\tes\api_testing\database\migrations\2023_08_31_083929_api_datas.php] created successfully.

PS C:\laravel\tes\api_testing>
```

```

class Up extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('api_datas', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('bank');
            $table->string('alamat');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('api_datas');
    }
}
```

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

The screenshot shows the Visual Studio Code interface with the terminal tab active. The command `php artisan migrate` is entered, followed by its execution output:

```

PS C:\laravel\tes\api_testing> php artisan migrate
INFO: Preparing database.
Creating migration table ..... 26ms DONE
INFO: Running migrations.
2014_10_12_000000_create_users_table ..... 40ms DONE
2014_10_12_100000_create_password_resets_table ..... 50ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 30ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 65ms DONE
2023_08_31_083929_api_datas ..... 21ms DONE

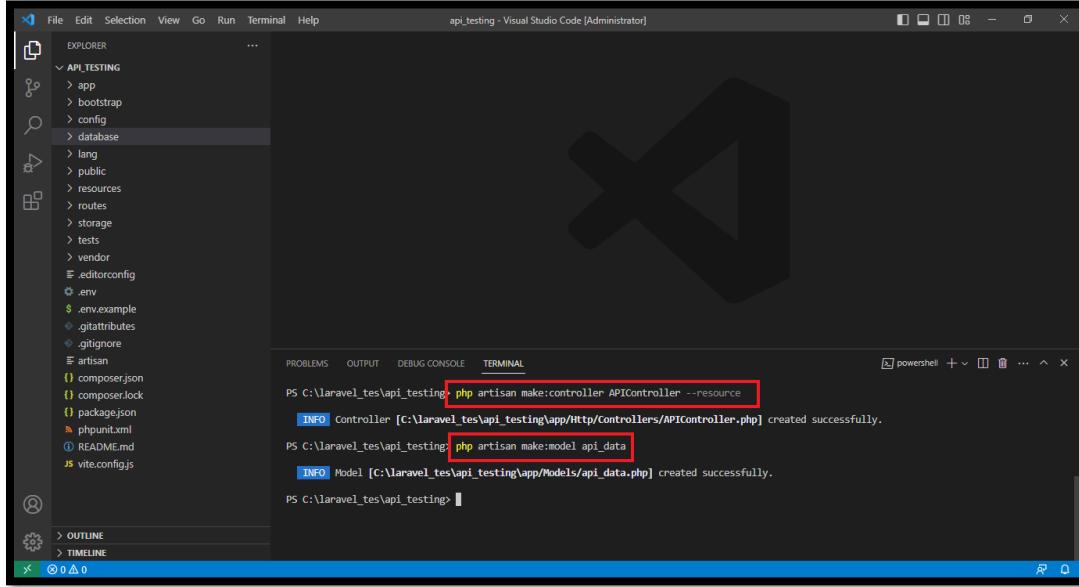
```

7. Buka kembali database pada browser anda dan pastikan bahwa tabel dengan nama “api\_datas” telah berada di dalam database api\_testing.

The screenshot shows the phpMyAdmin interface connected to the `api_testing` database. The left sidebar shows the database structure with tables: `api_datas`, `failed_jobs`, `migrations`, `password_resets`, `personal_access_tokens`, and `users`. The main area displays a list of tables with their details:

Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
<code>api_datas</code>		0	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
<code>failed_jobs</code>		0	InnoDB	utf8mb4_unicode_ci	32,0 KB	-
<code>migrations</code>		5	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
<code>password_resets</code>		0	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
<code>personal_access_tokens</code>		0	InnoDB	utf8mb4_unicode_ci	48,0 KB	-
<code>users</code>		0	InnoDB	utf8mb4_unicode_ci	32,0 KB	-
<b>6 tabel</b>	<b>Jumlah</b>	<b>0</b>	<b>InnoDB</b>	<b>utf8mb4_general_ci</b>	<b>160,0 KB</b>	<b>0 B</b>

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.

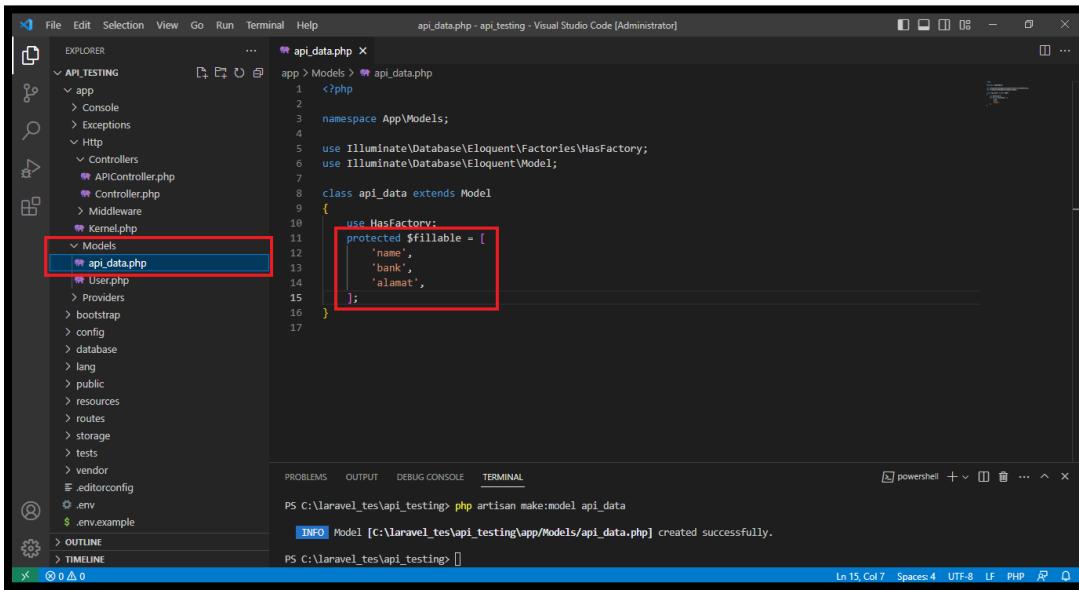


The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays two commands run in the directory C:\laravel\_tes\api\_testing:

```
PS C:\laravel_tes\api_testing> php artisan make:controller APIController --resource
[INFO] Controller [C:\laravel_tes\api_testing\app\Http\Controllers\APIController.php] created successfully.

PS C:\laravel_tes\api_testing> php artisan make:model api_data
[INFO] Model [C:\laravel_tes\api_testing\app\Models\api_data.php] created successfully.
```

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



The screenshot shows the Visual Studio Code interface with the code editor tab active. The file api\_data.php is open, showing the following PHP code:

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class api_data extends Model
{
    use HasFactory;
    protected $fillable = [
        'name',
        'bank',
        'alamat',
    ];
}
```

The code editor shows the file path as app/Models/api\_data.php. The terminal at the bottom shows the command was run successfully.

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.

```

File Edit Selection View Go Run Terminal Help APIController.php - api_testing - Visual Studio Code [Administrator]
EXPLORER API_CONTROLLER app api_data.php APIController.php
app > Http > Controllers > APIController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\api_data;
7
8 class APIController extends Controller
9 {
10     /**
11      * Display a listing of the resource.
12      *
13      * @return \Illuminate\Http\Response
14     */
15    public function index()
16    {
17        //
18    }
19
20    /**
21     * Show the form for creating a new resource.
22     *
23     * @return \Illuminate\Http\Response
24     */
25    public function create()
26    {
27        //
28    }
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\laravel_tes\api_testing> php artisan make:model api_data
[INFO] Model C:\laravel_tes\api_testing\app\Models\api_data.php created successfully.
PS C:\laravel_tes\api_testing>

```

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.

```

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

```

```

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

```

```

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";
}

```

```

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";
}

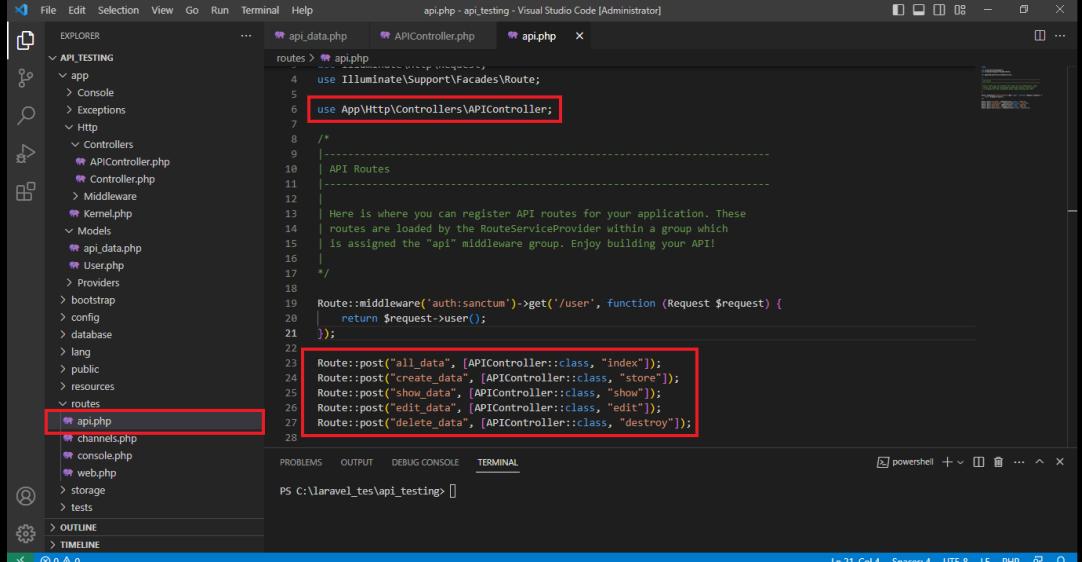
```

```

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.



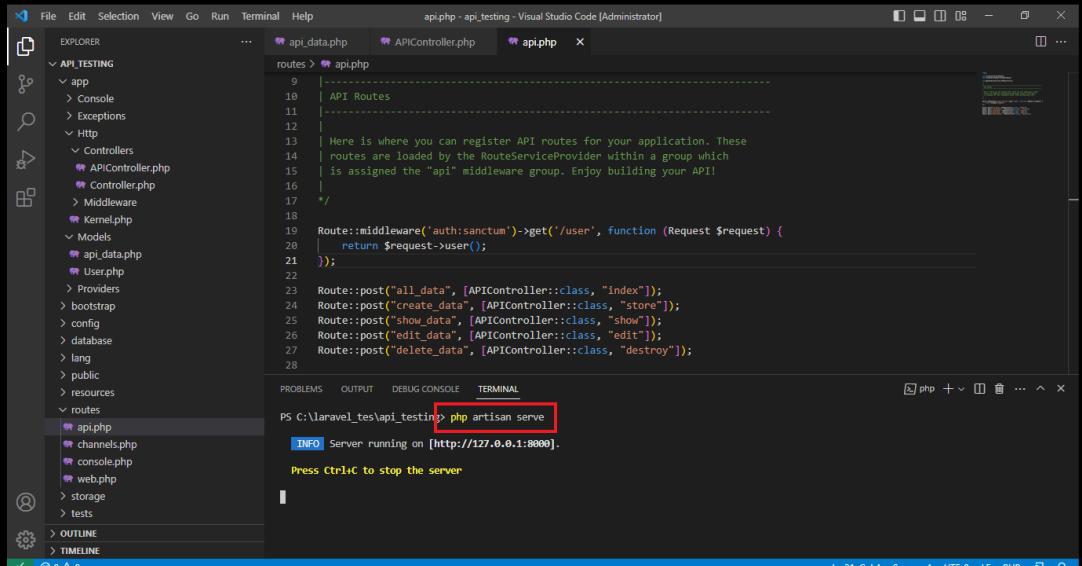
```

File Edit Selection View Go Run Terminal Help
api.php - api_testing - Visual Studio Code [Administrator]
EXPLORER API TESTING ...
routes > api.php
4 use Illuminate\Support\Facades\Route;
5
6 use App\Http\Controllers\APIController;
7
8 /*
9 |-----|
10 | API Routes
11 |
12 |
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider within a group which
15 | is assigned the "api" middleware group. Enjoy building your API!
16 |
17 */
18
19 Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
20 |     return $request->user();
21 });
22
23 Route::post("all_data", [APIController::class, "index"]);
24 Route::post("create_data", [APIController::class, "store"]);
25 Route::post("show_data", [APIController::class, "show"]);
26 Route::post("edit_data", [APIController::class, "edit"]);
27 Route::post("delete_data", [APIController::class, "destroy"]);
28

```

## 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.



```

File Edit Selection View Go Run Terminal Help
api.php - api_testing - Visual Studio Code [Administrator]
EXPLORER API TESTING ...
routes > api.php
9
10 |-----|
11 | API Routes
12 |
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider within a group which
15 | is assigned the "api" middleware group. Enjoy building your API!
16 |
17 */
18
19 Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
20 |     return $request->user();
21 });
22
23 Route::post("all_data", [APIController::class, "index"]);
24 Route::post("create_data", [APIController::class, "store"]);
25 Route::post("show_data", [APIController::class, "show"]);
26 Route::post("edit_data", [APIController::class, "edit"]);
27 Route::post("delete_data", [APIController::class, "destroy"]);
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\laravel\tes\api_testing> php artisan serve
[INFO] Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server

```

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)

The screenshot shows the Postman interface. A red box highlights the 'Body' tab and the 'form-data' option. Another red box highlights the 'Send' button. The response body contains the message 'Berhasil Menyimpan Data' with a red arrow pointing to it, and 'Nilai kembalian ketika proses berhasil' below it.

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.

The screenshot shows the phpMyAdmin interface. The left sidebar shows the database structure. The main area shows the 'api\_datas' table with one record. The 'id' column is highlighted with a red box. The record details are: id: 5, name: fajar, bank: BCA, alamat: Bandulan, created\_at: 2023-08-31 11:01:10, updated\_at: 2023-08-31 11:01:10.

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)

POST http://localhost:8000/api/show\_data

Key	Value
<input checked="" type="checkbox"/> id	5

```

1
2   "id": 5,
3   "name": "fajar",
4   "bank": "BCA",
5   "alamat": "Bandulan",
6   "created_at": "2023-08-31T11:01:10.000000Z",
7   "updated_at": "2023-08-31T11:01:10.000000Z"
8

```

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)

POST http://localhost:8000/api/edit\_data

Key	Value
<input checked="" type="checkbox"/> name	fajar
<input checked="" type="checkbox"/> bank	MANDIRI
<input checked="" type="checkbox"/> alamat	Bandulan
<input checked="" type="checkbox"/> id	5

```

1 Berhasil Mengubah Data

```

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

The screenshot shows the phpMyAdmin interface for the 'api\_testing' database. The 'api\_data' table is selected. A single row is highlighted with a red box, showing the following data:

	<b>id</b>	<b>name</b>	<b>bank</b>	<b>alamat</b>	<b>created_at</b>	<b>updated_at</b>
	5	fajar	MANDIRI	Bandulan	2023-08-31 11:01:10	2023-08-31 11:02:02

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).

The screenshot shows a POST request in Postman to the URL `http://localhost:8000/api/delete_data`. The request is set to 'POST' method. In the 'Body' tab, the 'form-data' option is selected, and a key-value pair 'id' with value '5' is present. The response status is 200 OK, and the message is 'Berhasil Menghapus Data'.

Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/> id	5		
Key	Value	Description	

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

The screenshot shows the phpMyAdmin interface for the 'api\_testing' database. The left sidebar lists tables such as 'Baru', 'api\_datas', 'failed\_jobs', 'migrations', 'password\_resets', 'personal\_access\_tokens', and 'users'. The main area displays the results of a SQL query: 'SELECT \* FROM `api\_datas`'. The results table is empty, indicated by the message 'MySQL memberikan hasil kosong (atau nol baris). (Pencarian dilakukan dalam 0.0005 detik.)'. Below the results, there is a form for creating a view ('Operasi hasil kueri') and a button to generate a view ('Buat tampilan').

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](http://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).

The screenshot shows the Postman application interface. A red box highlights the 'Body' tab of a POST request to 'http://localhost:8000/api/all\_data'. The response body is displayed in a code editor, showing two JSON objects representing data from the 'api\_datas' table:

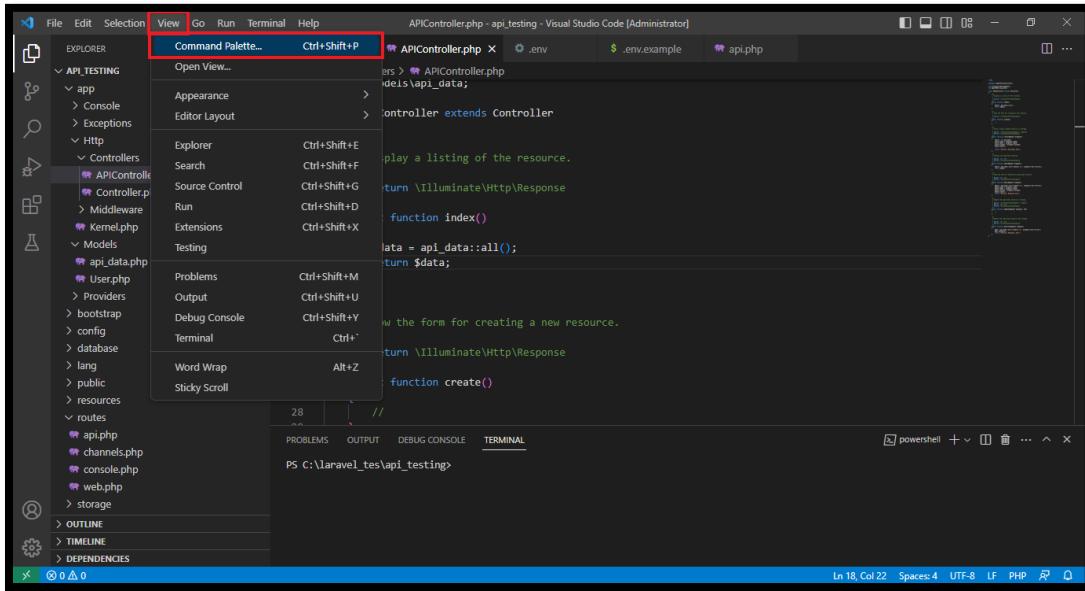
```

1 {
2   "id": 6,
3   "name": "fajars",
4   "pank": "BCA",
5   "alamat": "Bandulan",
6   "created_at": "2023-08-31T11:03:23.000000Z",
7   "updated_at": "2023-08-31T11:03:23.000000Z"
8 },
9 {
10   "id": 7,
11   "name": "fajars",
12   "pank": "BRI",
13   "alamat": "Bandulan",
14   "created_at": "2023-08-31T11:03:27.000000Z",
15   "updated_at": "2023-08-31T11:03:27.000000Z"
16 }
17
18

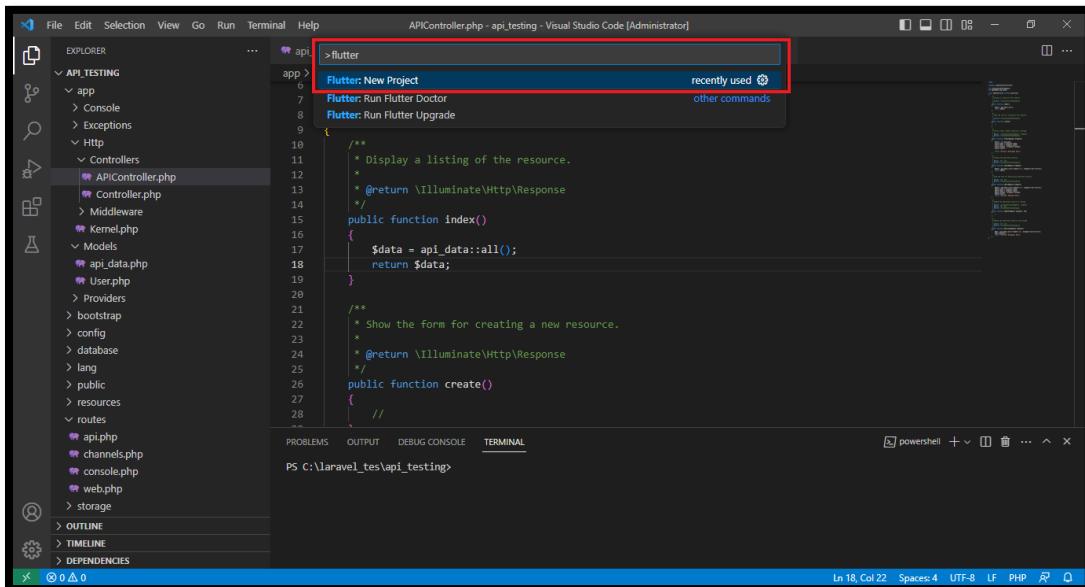
```

## D. Pembuatan Projek Flutter

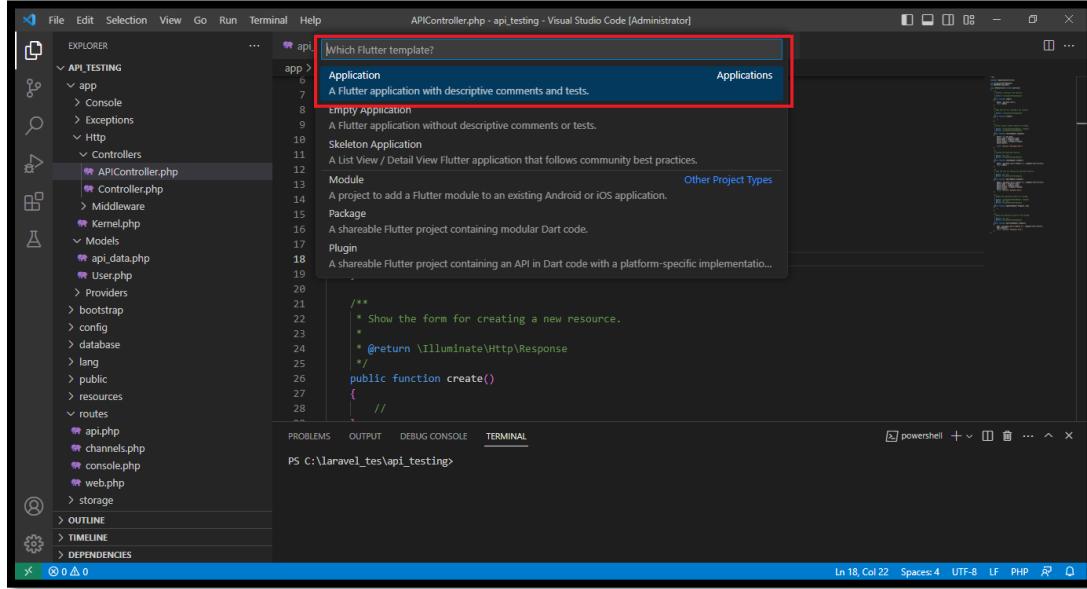
1. Buka aplikasi Visual Studio Code anda kemudian buka menu “View” dan pilih menu “Command Palette...”.



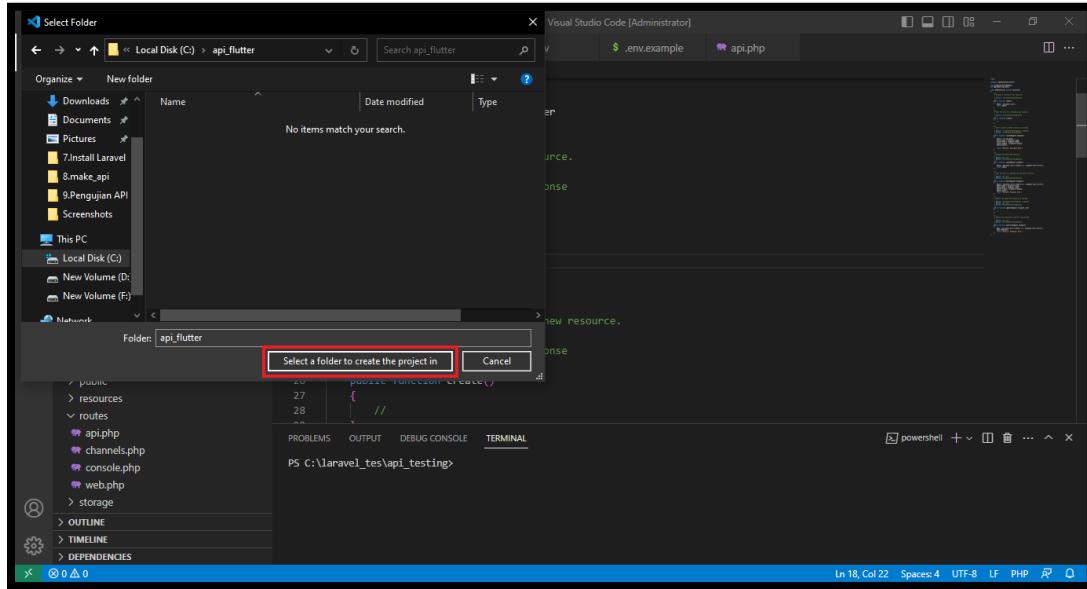
2. Ketikkan “flutter” pada searchbox yang muncul tanpa tanda kutip dan pilih “Flutter: New Project”.



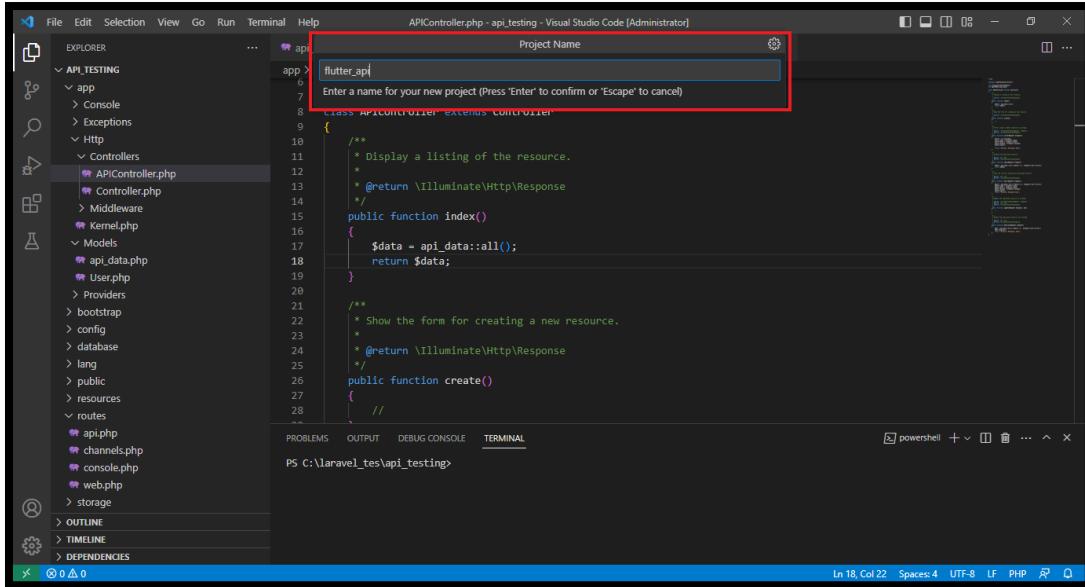
3. Kemudian pilih “**Flutter Application with descriptive..**” jangan pilih yang “Empty Application”, karena kita akan menggunakan template yang sudah disediakan oleh flutter untuk mempermudah kita dalam menyelesaikan proyek ini.



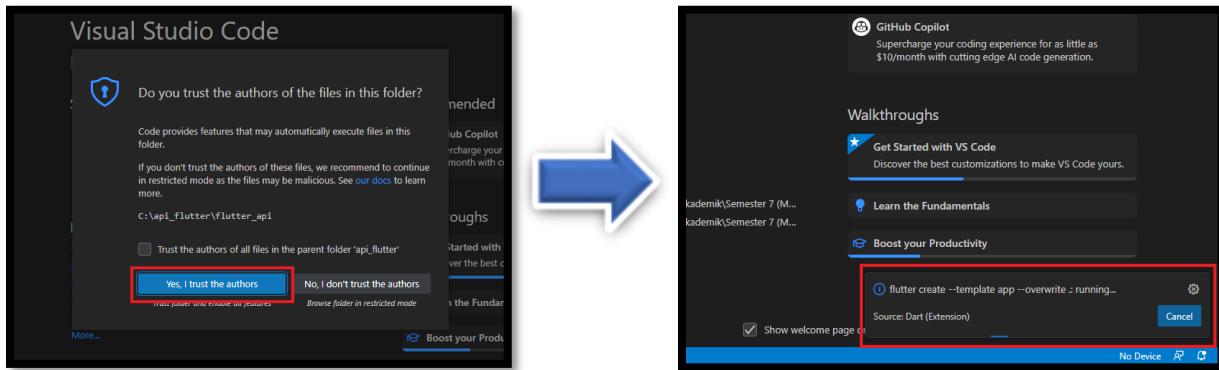
4. Kemudian pilih lokasi yang ingin anda gunakan dan klik “Select a folder...”.



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 trust the author” dan tunggu hingga proses pembuatan proyek selesai.



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

The screenshot shows the Visual Studio Code interface with a Flutter project open. The left sidebar displays the project structure under 'FLUTTER\_API'. The main editor window shows the file 'main.dart' with the following code:

```
lib > main.dart
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Demo',
15      theme: ThemeData(
16        // This is the theme of your application.
17        //
18        // TRY THIS: Try running your application with "flutter run". You'll see
19        // the application has a blue toolbar. Then, without quitting the app,
20        // try changing the seedColor in the colorScheme below to Colors.green
21        // and then invoke "hot reload" (save your changes or press the "hot
22        // reload" button in a Flutter-supported IDE, or press "r" if you used
23        // the command line to start the app).
24        //
25        // Notice that the counter didn't reset back to zero; the application
26        // state is not lost during the reload. To reset the state, use hot
27        // restart instead.
28        //
29        // This works for code too, not just values: Most code changes can be
30        // tested with just a hot reload.
31        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
32        useMaterial3: true,
33      ),
34    );
35  }
36}
```

A red box highlights the status bar at the bottom right, which displays the message: 'Your Flutter project is ready! Press F5 to start running.'

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

The screenshot shows the VS Code interface with the following details:

- File Explorer:** On the left, it shows the project structure with files like `main.dart`, `pubspec.yaml`, `flutter_api.yaml`, and `pubspec.inch`. The `pubspec.yaml` file is currently selected and highlighted with a red box.
- Editor:** The main area displays the `pubspec.yaml` file content. A specific line, `dio: '^3.2'`, is highlighted with a red box.
- Bottom Status Bar:** Shows the current file is `pubspec.yaml`, with line 40, column 14. It also shows tabs for `main.dart` and `! pubspec.yaml`.
- Bottom Task Bar:** Shows the command `flutter pub get --no-example: running...` with a progress bar and a `Cancel` button.

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.

```

File Edit Selection View Go Run Terminal Help main.dart - flutter_api - Visual Studio Code [Administrator]
EXPLORER main.dart pubspec.yaml
FLUTTER_API
> dart_tool
> idea
> android
> ios
> lib
> main.dart
> linux
> macos
> test
> web
> windows
> .gitignore
> .metadata
analysis_options.yaml
flutter_apiml
pubspec.lock
pubspec.yaml
README.md

main.dart
import 'package:flutter/material.dart';
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(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a blue toolbar. Then, without quitting the app,
        // try changing the seedColor in the colorScheme below to Colors.green
        // and then invoke "hot reload" (save your changes or press the "hot
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // flutter run).
      ),
    );
  }
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
material_color_utilities 0.5.0 (0.8.0 available)
stack_trace 1.11.0 (1.11.1 available)
stream_channel 2.1.1 (2.1.2 available)
test_api 0.6.0 (0.6.1 available)
+ typed_data 1.3.2
Changed 3 dependencies!
exit code 0
flutter (flutter_api) 8:26 AM 9/6/2023

```

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(  
        title: ...  
        content: ...  
        actions: ...  
    );  
}
```

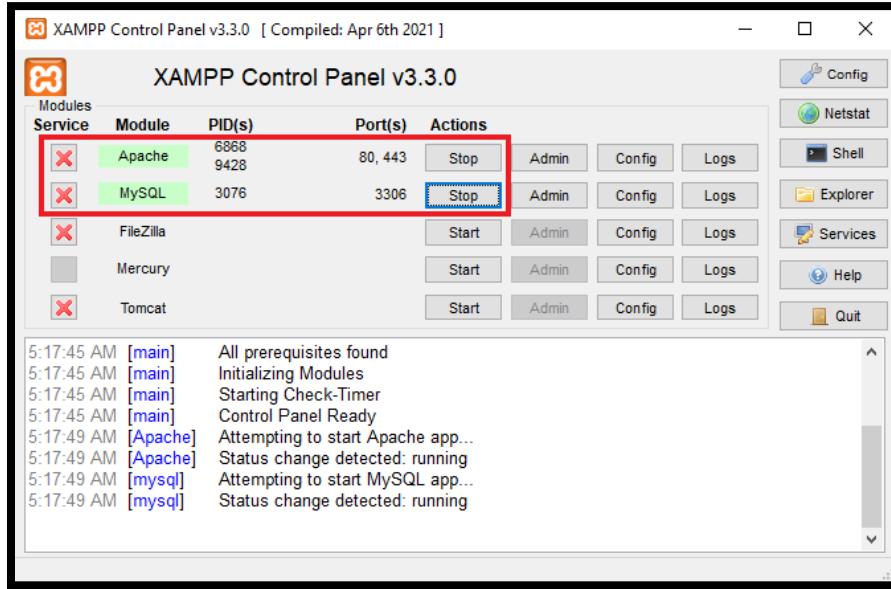
```
Widget _buildPopupDialog_update(BuildContext context) {  
    return new AlertDialog(  
        title: ...  
        content: ...  
        actions: ...  
    );  
}
```

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.

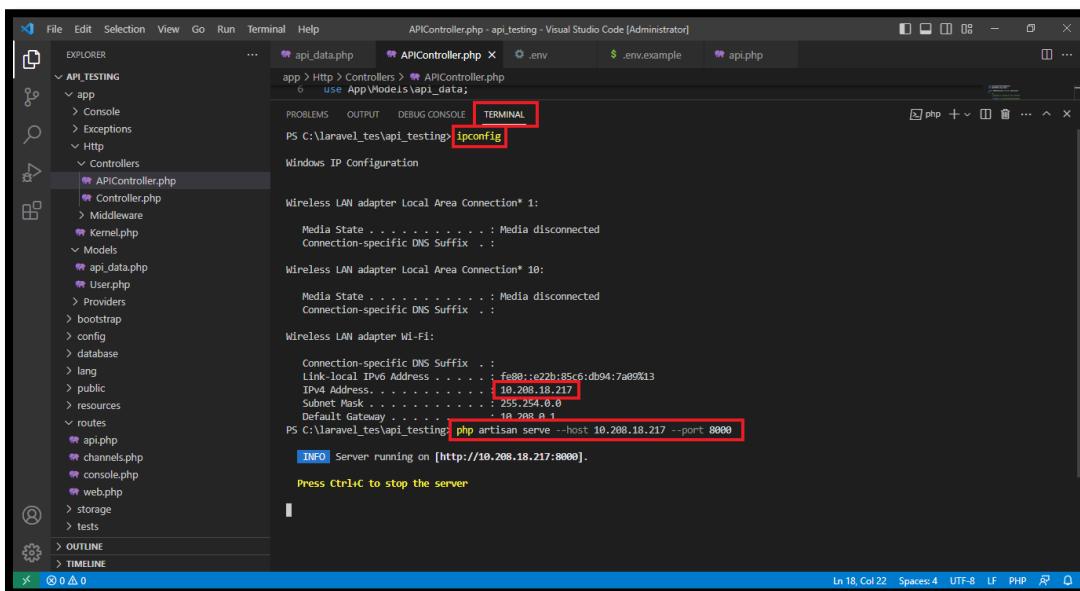
```
void dispose() {  
    WidgetsBinding.instance.removeObserver(this);  
    super.dispose();  
}  
  
@override  
void didChangeAppLifecycleState(AppLifecycleState state) {  
    print('MyApp state = $state');  
    if (state == AppLifecycleState.inactive) {  
        // app transitioning to other 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

- Buka aplikasi XAMPP dan pastikan Module “Apache dan MySQL” sudah berjalan.



- 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.

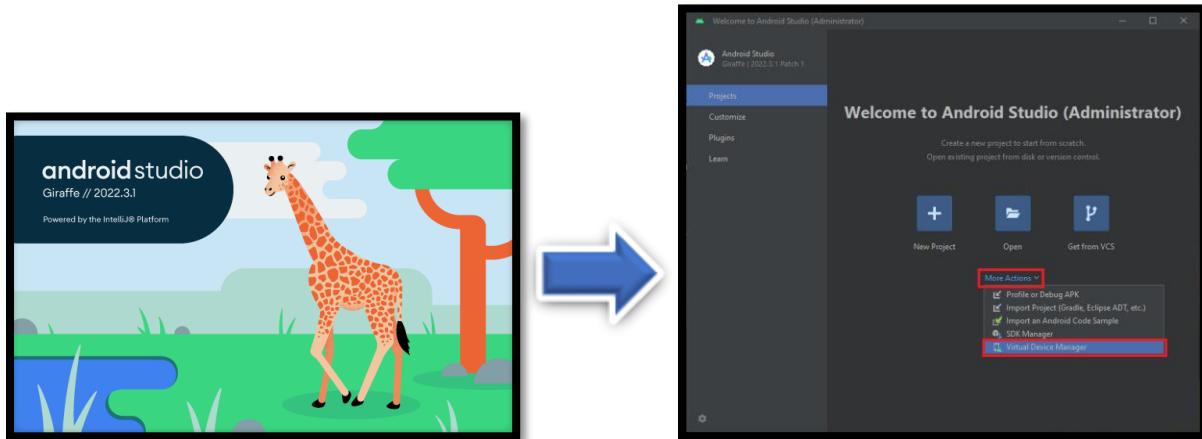
```

main.dart 4 | pubspec.yaml
lib/main.dart:18
18 String url_domain = "http://10.208.18.217:8000/";
19 String url_get_data = url_domain + "/api/get_data";
20 String url_create_data = url_domain + "/api/create_data";
21 String url_show_data = url_domain + "/api/show_data";
22 String url_update_data = url_domain + "/api/edit_data";
23 String url_delete_data = url_domain + "/api/delete_data";
24
25 void main() {
26   runApp(const MyApp());
27 }
28
29 class MyApp extends StatelessWidget {
30   const MyApp({super.key});
31
32   // This widget is the root of your application.
33   @override
34   Widget build(BuildContext context) {
35     return MaterialApp(
36       title: 'Flutter Demo',
37       theme: ThemeData(
38         colorScheme: ColorScheme.fromSwatch().copyWith(
39           primary: Colors.purple,
40           secondary: Colors.amber,
41           background: Colors.pink,
42           error: Colors.pink,
43           surface: Colors.pink,
44           onPrimary: Colors.pink,
45           onSecondary: Colors.pink,
46           onBackground: Colors.pink,
47           onError: Colors.pink,
48           onSurface: Colors.pink,
49         ),
50         textTheme: TextTheme(
51           headline1: TextStyle(
52             color: Colors.pink,
53             fontSize: 24,
54             fontWeight: FontWeight.w500,
55             letterSpacing: 0.2,
56             height: 1.2,
57             wordSpacing: 0.5,
58           ),
59         ),
60       ),
61       home: const MyHomePage(),
62     );
63   }
64 }

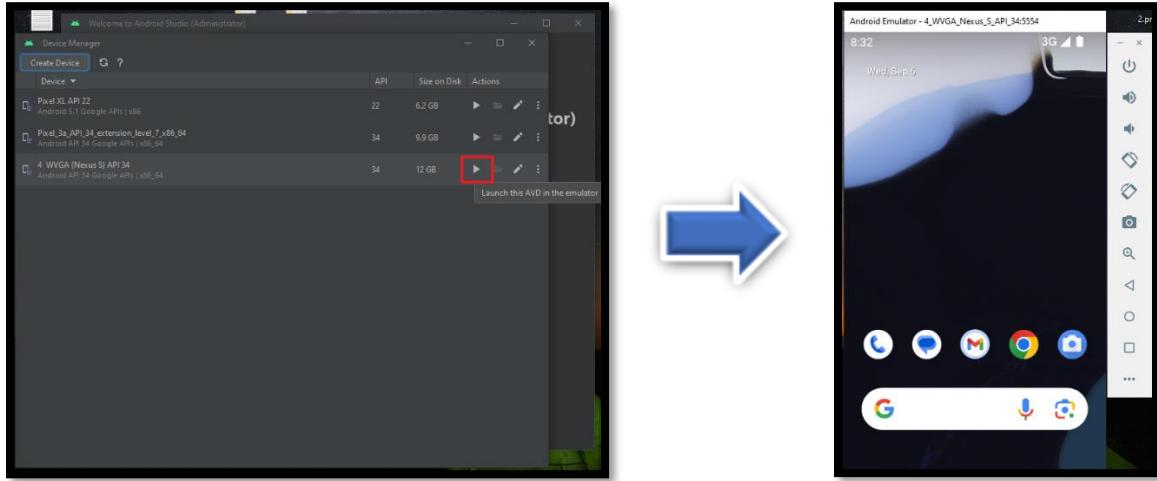
PS C:\api_flutter\flutter_api>

```

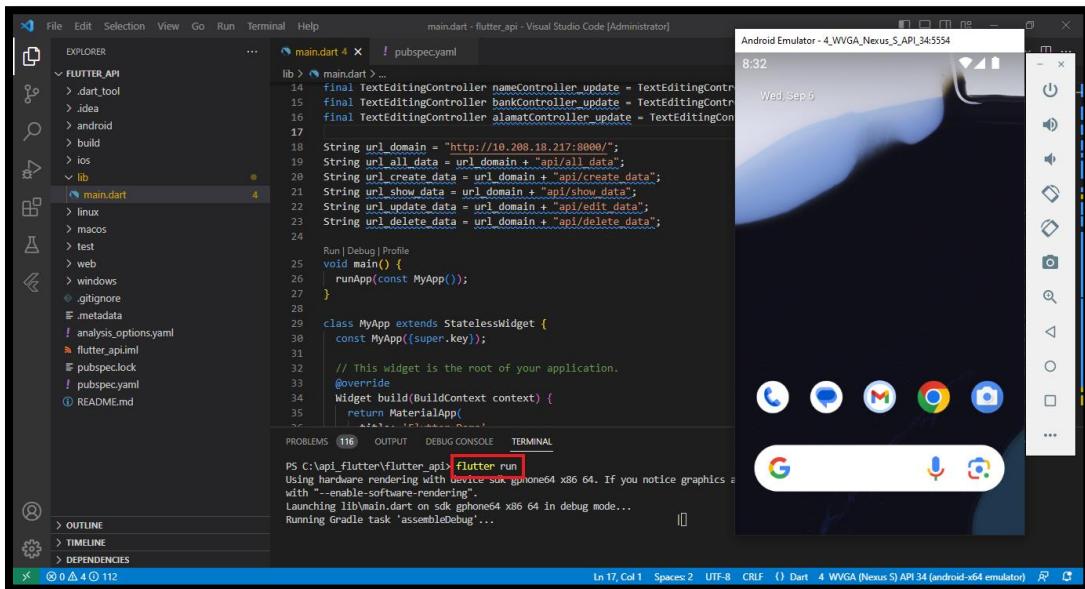
4. Selanjutnya buka aplikasi android studio, tunggu proses loading selesai dan pilih “More Action” kemudian pilih **Virtual Device Manager**.



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.



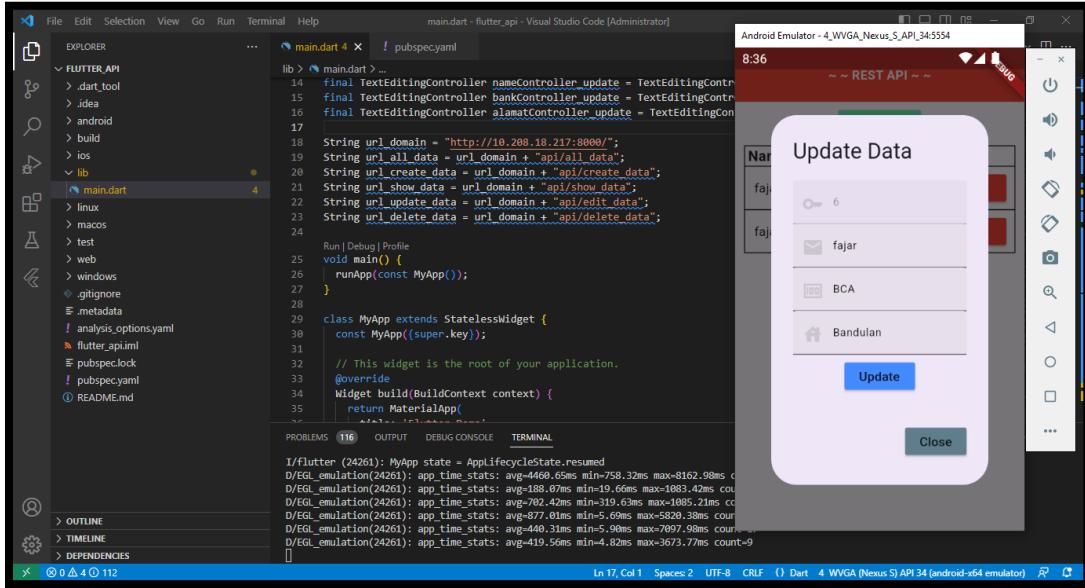
7. Tunggu proses loading selesai dan android SDK akan menampilkan output seperti di bawah ini. Terdapat berbagai macam tombol yang dapat dioperasikan, seperti tombol “Add Data” yang berguna untuk menambahkan data baru, tombol “Refresh Data” berguna untuk merefresh data pada tabel, serta tombol “Updata”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

The screenshot shows the Visual Studio Code interface with the main.dart file open in the editor. The code implements a simple REST API client to interact with a database. The Android emulator window shows a table with two rows of data: 'fajar' at 'BCA' and 'fajar' at 'BRI'. There are 'Update' and 'Delete' buttons for each row. A green 'Add Data' button is also visible. The emulator's status bar indicates it's running on an 'WVGA (Nexus S) API 34 (android-x64 emulator)' device.

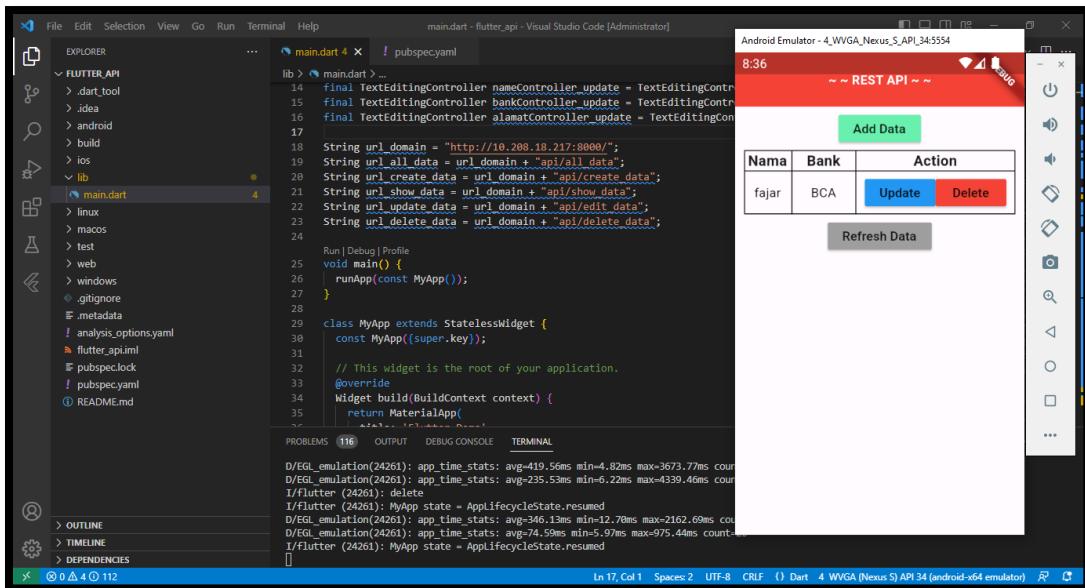
8. Apabila anda mengklik tombol “Add Data” sistem akan menampilkan Form untuk mengisi data yang baru.

The screenshot shows the Visual Studio Code interface with the main.dart file open. The Android emulator window displays a modal dialog titled 'Form Data' containing three text input fields labeled 'Nama', 'Bank', and 'Alamat', along with a 'Send' button and a 'Close' button. This dialog is triggered by clicking the 'Add Data' button in the previous screenshot.

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
<pre> 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,       ),     ); } </pre>

```
        ),
        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(
    // mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    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 = "";  
}
```