

## Table of Contents

<b>LARAVEL REST API LEARNING BY CODING.....</b>	<b>3</b>
<b>BASIC REST API.....</b>	<b>3</b>
<i>Apa itu rest api .....</i>	3
<i>Bagaimana REST API Digunakan Di Industri.....</i>	3
<b>FORMAT REST API YANG BAIK .....</b>	<b>4</b>
<i>Gunakan Json Sebagai Format REST API .....</i>	4
<i>Gunakan Kata Benda Di Endpoint .....</i>	5
<i>Gunakan Logical Nesting Di Endpoint .....</i>	6
<i>Handle Error Dengan Status Code Standar.....</i>	7
<i>Berikan Fitur Filter, Sorting Dan Pagination.....</i>	8
<i>Mengikuti Best Practice Dalam Security.....</i>	9
<i>Membuat Data Cache Untuk Meningkatkan Performa.....</i>	9
<i>Membuat Versi API Agar Mudah Dimigrasi .....</i>	10
<b>INSTALL SOFTWARE PENDUKUNG .....</b>	<b>10</b>
<i>Install Chocolatey .....</i>	10
<i>Mencari Perangkat Lunak / Package pada chocolatey .....</i>	11
<i>Install Perangkat Lunak Dengan Chocolatey.....</i>	12
<i>Upgrade Perangkat Lunak Dengan Chocolatey.....</i>	13
<i>Mengupgrade Chocolatey .....</i>	13
<i>Menampilkan Perangkat Lunak yang diinstall dengan chocolatey .....</i>	14
<i>Contoh penggunaan chocolatey.....</i>	14
<i>Install Php.....</i>	15
<i>Install Composer.....</i>	17
<i>Install Laravel .....</i>	18
<i>Install Mysql .....</i>	18
<i>Install Vscode.....</i>	21
<i>Install Postman.....</i>	23
<b>MULAI MEMBUAT REST API DENGAN LARAVEL.....</b>	<b>24</b>
<i>Create Project Laravel .....</i>	24
<i>Konfigurasi Project Laravel.....</i>	24
<i>Penjelasan routes web.php vs api.php .....</i>	28
<i>Membuat Endpoint GET dan Mengakses dari Postman.....</i>	28
<i>Membuat Endpoint POST dan Mengakses dari Postman.....</i>	32
<i>Membuat Endpoint PUT dan Mengakses dari Postman.....</i>	35
<i>Membuat Endpoint DELETE dan Mengakses dari Postman .....</i>	38

MINI PROJECT REST API BMI CALCULATOR .....	41
<i>Project Overview</i> .....	41
<i>Mendesain REST API</i> .....	41
<i>Membuat Controller</i> .....	42
<i>Testing REST API</i> .....	44
<i>Challenge</i> .....	46
MINI PROJECT REST API QUOTE APP .....	46
<i>Project Overview</i> .....	46
<i>Mendesain Database</i> .....	46
<i>Membuat Model</i> .....	47
<i>Membuat Migration</i> .....	48
<i>Membuat Factory</i> .....	52
<i>Mengupdate Quote Seeder</i> .....	53
<i>Memanggil QuoteSeeder di Database Seeder</i> .....	55
<i>Membuat Route di Api.php</i> .....	57
<i>Membuat Controller</i> .....	59
<i>Membuat Endpoint GET untuk menampilkan list quote</i> .....	61
<i>Membuat Resource Quote Untuk Merapikan Response</i> .....	63
<i>Membuat Endpoint POST untuk membuat quote baru di database</i> .....	68
<i>Menambahkan Form Request Pada Function Store</i> .....	73
<i>Membuat Endpoint GET untuk menampilkan satu Quote</i> .....	78
<i>Membuat Endpoint PUT untuk mengedit Quote</i> .....	81
<i>Membuat Endpont DELETE untuk menghapus Quote</i> .....	87
IMPROVE MINI PROJECT REST API QUOTE APP DENGAN ADMIN AKSES .....	92
<i>Install dan Config Laravel Sanctum</i> .....	93
<i>Protect Endpoint Menggunakan Laravel Sanctum</i> .....	94
<i>Test Protected dan Open Endpoint</i> .....	95
<i>Membuat Controller dan Method Register</i> .....	96
<i>Membuat Controller dan Method Login</i> .....	100
<i>Membuat Controller dan Method Logout</i> .....	103
<b>BIODATA PENULIS .....</b>	<b>105</b>
<b>KESIMPULAN .....</b>	<b>107</b>
<b>GLOSARIUM .....</b>	<b>108</b>
<b>DAFTAR PUSTAKA .....</b>	<b>109</b>

# Laravel REST API Learning By Coding

## Basic REST API

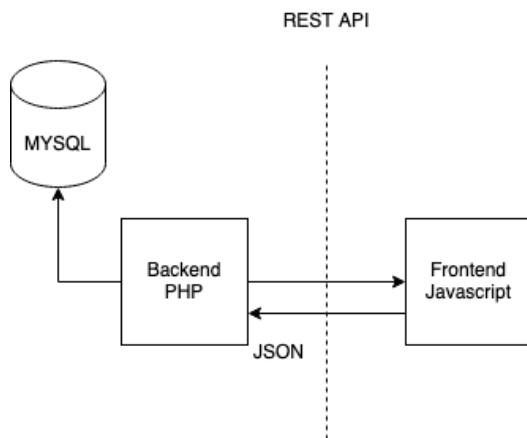
Sebelum memulai membuat REST API alangkah lebih baiknya jika kita dapat memahami dengan baik ap aitu REST API dan bagaimana implementasinya di dunia industri.

### Apa itu rest api

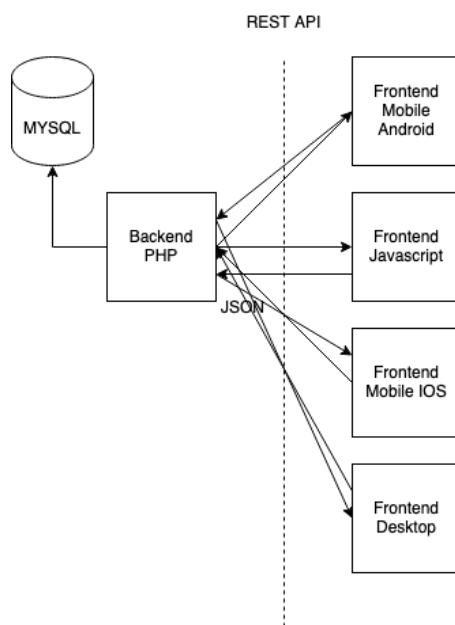
REST adalah akronim dari **REpresentational State Transfer** and architectural style for distributed hypermedia systems. Yang dikemukakan oleh Roy Fielding pada tahun 2000 di desertasinya dengan judul **Representational State Transfer (REST)**. REST berisi aturan aturan dan Batasan yang harus di miliki oleh sebuah sistem, sistem ini nantinya dapat dinamakan RESTful (Massé & Massé, 2012).

### Bagaimana REST API Digunakan Di Industri

Di industri REST API digunakan sebagai platform untuk komunikasi antara backend dengan front end. Komunikasi ini dilakukan dengan menggunakan format JSON(JavaScript Object Notation). Dengan menggunakan JSON sebagai media komunikasi REST API dapat digunakan menghubungkan antara backend dengan frontend walaupun backend dan front end menggunakan Bahasa pemrograman yang berbeda. Dengan menggunakan pendekatan yang seperti ini sebuah tim dapat membuat aplikasi yang memiliki aplikasi dengan banyak front end misalnya web, mobile android, mobile ios dan desktop yang menggunakan satu backend yang sama contoh desain REST API dengan satu frontend dan banyak frontend dapat dilihat pada gambar



Gambar 1 Diagram REST API dengan single frontend



Gambar 2 Diagram REST API dengan multi frontend

## Format REST API Yang Baik

Ketika membuat sebuah backend yang menggunakan REST API ada beberapa best practice yang perlu diikuti dan diimplementasikan agar REST API yang dibuat memiliki performa yang baik dan mudah digunakan oleh tim frontend.

### Gunakan Json Sebagai Format REST API

REST API sebaiknya menggunakan JSON untuk proses request dan response. JSON adalah standar yang diterima untuk mengirim data. Hampir semua teknologi yang ada sekarang dapat menggunakan JSON. Contohnya di javascript ada method bawaan yang

mampu melakukan JSON encoding dan decoding baik melalui library Fetch API atau menggunakan HTTP client bawaan. Sedangkan teknologi server side seperti Laravel juga sudah memiliki fitur decode dan encode JSON sebagai fitur bawaan.

Ada cara lain yang dapat digunakan mengirim data yaitu XML, namun XML tidak didukung oleh semua frameworks, akibatnya developer harus melakukan transformasi data secara manual. Sedangkan jika digunakan di client side (front end) khususnya web browser XML tidak dapat digunakan secara mudah akhirnya akan menambah effort untuk membuat kode program tambahan agar XML dapat dibaca oleh Browser.

Sedangkan dalam proses pengiriman data JSON mempermudah proses tersebut karena ketika proses pengiriman data yang ada di format JSON dapat dibaca dengan mudah oleh framework front end maupun framework backend. Sejauh ini menggunakan JSON merupakan cara yang paling mudah untuk mengirim data.

Untuk memastikan bahwa backend REST API membalas dengan response dalam format JSON, request dan response yang dikirimkan ke REST API harus memiliki set Content-Type di header request nya berupa “application/json”. Dengan menggunakan Content-type tersebut baik framework front end maupun backend dapat memproses data JSON tersebut dengan mudah.

Sedikit pengecualian dalam menggunakan JSON sebagai format komunikasi data antara backend dan front end adalah Ketika menggunakan JSON kita tidak dapat melakukan pengiriman data berupa file, untuk mengirim data dalam format file harus menggunakan form data dari backend ke front end.

### Gunakan Kata Benda Di Endpoint

Ketika membuat endpoint pada backend REST API, sebaiknya menggunakan kata benda karena kata benda mewakili sebuah entitas dari endpoint tersebut. Menggunakan kata benda sebagai endpoint mempermudah tim dalam memahami REST API karena tidak ambigu jika dibandingkan dengan menggunakan kata kerja.

Kenapa menggunakan kata benda ? karena di HTTP request pada endpoint yang dibuat sudah memiliki kata kerja. Di HTTP request kata kerja yang sudah disediakan adalah GET, POST, PUT, DELETE. Selain itu kata kerja memiliki sifat ambigu karena antara satu developer dengan developer yang lain dapat memiliki penamaan yang berbeda terhadap satu

proses yang sama, misalnya getBook dengan retriveBook antara kata get dan retrive memiliki makna yang sama akan lebih baik jika get dan retrive dihilangkan dari endpoint.

Untuk membuat sebuah endpoint dapat menggunakan format berikut ini HTTP request sebagai kata kerja (verb) dan kata benda untuk mewakili entitas. Beberapa HTTP verb yang umum digunakan adalah :

HTTP VERB	TUJUAN
GET	Mengambil resource dari server
POST	Menyimpan data baru ke server
PUT	Mengupdate data yang ada di server
DELETE	Menghapus data dari server

Contoh Endpoint yang baik :

HTTP VERB	Endpoint	Tujuan
GET	/categories/:id	Mengambil data category berdasarkan id
GET	/categories/	Mengambil semua data category
POST	/categories	Membuat category baru
PUT	/categories/:id	Mengedit category berdasarkan id
DELETE	/categories/:id	Menghapus category berdasarkan id
DELETE	/categories/	Menghapus semua category

#### Gunakan Logical Nesting Di Endpoint

Ketika mendesain endpoint sebaiknya kita menggunakan “logical nesting” dimana kita mengelompokkan data yang sesuai dengan hierarkinya. Maksudnya jika sebuah objek memiliki objek lain maka endpoint sebaiknya di desain mengikuti pola yang sama. Ini merupakan best practice yang biasanya diterapkan di REST API walaupun bisa jadi desain database nya tidak mengikuti konsep tersebut.

Contohnya jika kita kan membuat endpoint untuk mengambil komentar dari sebuah artikel berita, sebaiknya kita menambahkan endpoint /comments di akhir dari endpoint article. Contoh detail nya dapat dilihat pada tabel berikut ini.

HTTP Verb	Endpoint	Tujuan
-----------	----------	--------

GET	/articles/	Mengambil semua artikel
GET	/articles/:id	Mengambil satu artikel
GET	/articles/:id/comments	Mengambil semua komentar dari artikel dengan id x
GET	/articles/:id/comments/:id	Mengambil satu komentar dari artikel dengan id x

Meskipun logical nesting ini baik namun perlu dipertimbangkan dalam mendesain endpoint agar nesting ini tidak terlalu dalam dan juga tidak merefleksikan desain database yang kita gunakan, karena Ketika nesting dilakukan mengikuti semua desain database akan mempermudah seorang hacker untuk memahami desain database yang kita buat.

#### Handle Error Dengan Status Code Standar

Salah satu kesalahan yang paling sering ditemukan ketika mendesain REST API adalah kesalahan dalam memberikan response error. HTTP secara default sudah memiliki beberapa response code yang dapat digunakan untuk mengidentifikasi error apa yang terjadi di backend. Dengan menggunakan HTTP error message yang benar akan mempermudah tim frontend dalam menggunakan REST API yang dibuat. Berikut ini error message yang umum digunakan di HTTP

Error Code	Error Message	Keterangan
400	Bad Request	Gagal karena validasi input.
401	Unauthorized	Gagal karena user mengakses route yang tidak boleh di akses jika tidak terauthentikasi.
403	Forbidden	User nya terauthentikasi tapi tidak memiliki akses terhadap route tersebut
404	Not Found	Resource yang diinginkan tidak ditemukan
500	Internal Server Error	General error dari server, sebaiknya error ini tidak dikeluarkan secara langsung harus dilakukan try catch.
502	Bad Gateway	Gagal karena invalid response dari backend

503	Service Unavailable	Gagal karena ada error dari sisi server bisa jadi karena overload.
-----	---------------------	--

#### Berikan Fitur Filter, Sorting Dan Pagination

Database yang digunakan pada REST API bisa jadi memiliki data yang banyak. Ketika menampilkan data di REST API tidak bijak jika ditampilkan semua data dari database. Olehkarena itu diperlukan cara untuk menyajikan data dengan aman dan efisien. Salah satu cara yang dapat digunakan adalah filtering, sorting dan pagination.

Untuk melakukan filtering dapat ditambahkan parameter baru dibelakang endpoint standar yang sudah dibuat. Berikut ini contoh yang dapat digunakan.

HTTP VERB	Endpoint	Tujuan
GET	/categories/?name=soko	Mengambil category dengan filter name = soko
GET	/categories/?name=soko&type=important	Mengambil category dengan filter name = soko dan type = important

Filter juga dapat digabung dengan sorting, biasanya bisa digunakan dengan notasi + dan -, notasi + digunakan untuk sorting ascending dan – digunakan untuk sorting descending.

HTTP VERB	Endpoint	Tujuan
GET	/categories/?name=soko&sort=-name	Mengambil category dengan filter name = soko sorting name descending
GET	/categories/?name=soko&type=important&sort=+name	Mengambil category dengan filter name = soko dan type = important dan sorting name ascending

Selain itu juga dapat ditambahkan pagination yang membatasi jumlah response yang diberikan.

HTTP VERB	Endpoint	Tujuan

GET	/categories/?name=soko&sort=-name&page=1	Mengambil category dengan filter name = soko sorting name descending, di paging ke halaman pertama
GET	/categories/?name=soko&type=important&sort=+name&page=1	Mengambil category dengan filter name = soko dan type = important dan sorting name ascending, dipaging ke halaman pertama

#### Mengikuti Best Practice Dalam Security

Pada umumnya komunikasi antara backend dan front end harus terproteksi, jadi penggunaan SSL / TLS (HTTPS) pada request REST API menjadi sesuatu yang wajib digunakan. Untuk menggunakan HTTPS kita dapat memanfaatkan sebuah SSL certificate, untuk mendapatkan certificate ini dapat menggunakan certbot dimana dapat diperoleh dengan gratis. Oleh karena itu ketika melakukan deploy rest api wajib menggunakan HTTPS.

Selain security dari sisi SSL dan TLS kita juga perlu melakukan pembatasan akses, contohnya seorang user tidak boleh mengakses data dari user yang lain baik di aplikasi biasa maupun dari REST API, topik ini termasuk dalam pembahasan Role Base Access Management. Pada Role Based Access Management seorang user harus diberikan role se minimal mungkin agar tidak dapat mengakses resource dari user lain.

Pembatasan ini penting karena jika seorang user dapat mengakses resource milik user lain dapat terjadi pencurian data dan kebocoran data jika dibiarkan lebih lanjut.

#### Membuat Data Cache Untuk Meningkatkan Performa

Ketika mendesain REST API kita juga perlu mempertimbangkan proses caching untuk meningkatkan performa dari REST API kita, terutama untuk data yang sering di akses dan jarang berubah. Kita dapat melakukan cache dengan menyimpan response ke sebuah local memory daripada harus melakukan query ke database. Dengan menggunakan pendekatan ini request yang dilakukan oleh user akan di response dengan lebih cepat. Beberapa teknologi yang dapat digunakan sebagai cache adalah Redis, in memory caching dan lain lain.

## Membuat Versi API Agar Mudah Dimigrasi

Ketika mendesain REST API perlu diperhatikan agar membuat versioning, kenapa perlu dilakukan versioning ?. Karena ketika kita melakukan perubahan tanpa menggunakan versioning akan membuat client yang menggunakan REST API yang kita buat tidak dapat berjalan. Contoh ketika kita membuat perubahan pada endpoint yang tidak diberikan versioning misalkan di endpoint /categories, client sudah menggunakan response yang ada di /categories ketika kita merubah response JSON nya ada kemungkinan aplikasi client tidak dapat berjalan dengan normal.

Teknis versioning pada REST API yang baik adalah menyediakan endpoint yang sesuai dengan version yang disediakan contohnya endpoint v1/categories, ketika ada client yang menggunakan endpoint v1/categories kita dapat membuat endpoint categories baru tapi di endpoint v2/categories. Ketika kita mengembangkan endpoint baru dengan cara ini user yang menggunakan endpoint v1 tetap dilayani. Sembari melayani endpoint v1 kita dapat menginformasikan kepada client untuk melakukan update ke endpoint v2.

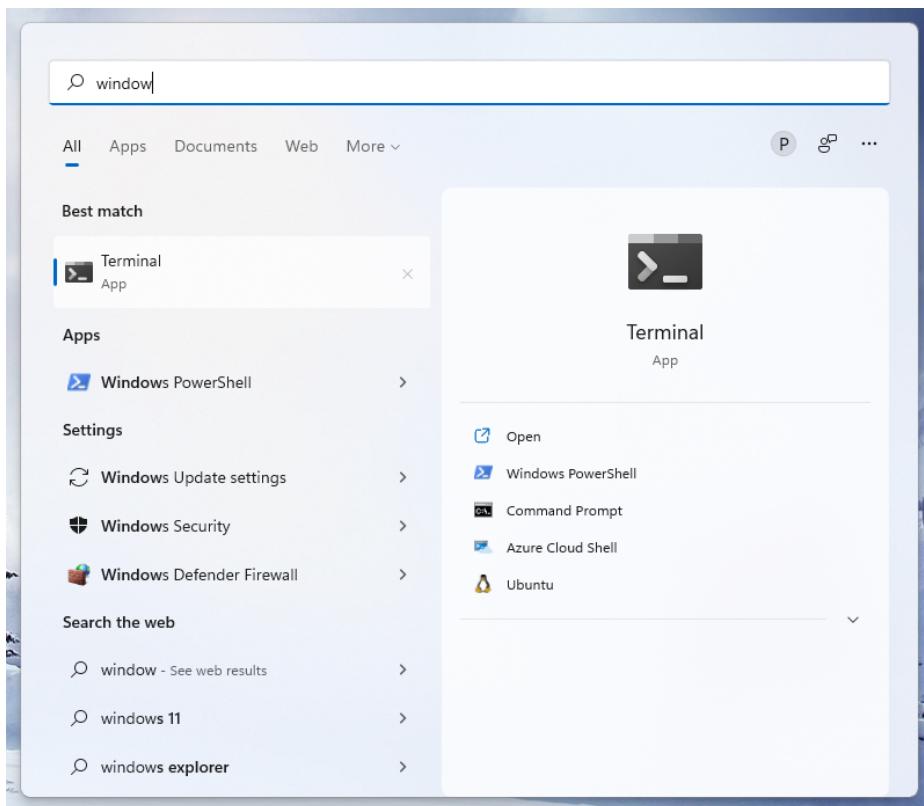
## Install Software Pendukung

Untuk memulai mempelajari database ada beberapa perangkat lunak yang harus di instal terlebih dahulu. Pada buku ini akan dijelaskan bagaimana cara melakukan installasi perangkat lunak pendukung menggunakan chocolatey. Chocolatey adalah sebuah software package manager yang dikhususkan untuk sistem operasi windows, package manager ini mirip dengan apt yang ada di linux dan brew yang ada di sistem operasi mac os. Dengan menggunakan chocolatey kita tidak perlu melakukan download software secara manual, chocolatey akan melakukan otomasi proses install perangkat lunak pada device anda.

## Install Chocolatey

Untuk mulai menginstall chocolatey ikutilah langkah langkah berikut ini :

1. Buka terminal windows dengan menggunakan administrative shell atau klik kanan dan run as administrator.



2. Jalankan perintah “Get-ExecutionPolicy” pada terminal untuk memeriksa level execution policy jika execution policy masih “Restricted” gantilah agar menjadi “Bypass” atau “AllSigned” dengan menjalankan perintah berikut “Set-ExecutionPolicy AllSigned”
3. Selanjutnya jalankan command berikut dengan mengcopy dan paste ke windows terminal.

```
Set-ExecutionPolicy      Bypass      -Scope      Process      -Force;
[System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.p  
s1'))
```

4. Tungguhlah proses installasi chocolatey dan setelah proses install selesai anda dapat menginstall perangkat lunak yang di ada di halaman <https://community.chocolatey.org/packages>

Mencari Perangkat Lunak / Package pada chocolatey

Untuk mencari perangkat lunak / pacakage pada chocolatey dapat dilakukan dengan dua cara yaitu melalui terminal atau melalui website official chocolatey. Kali ini akan dibahas cara

mencari perangkat lunak dengan menggunakan terminal. Bukalah windows terminal kemudian gunakan perintah dibawah ini untuk mencari package yang ada di chocolatey.

```
choco search [keyword]
```

Berikut ini contoh penggunaan command search pada chocolatey

```
C:\>choco search hugo
Chocolatey v0.10.8
hugo 0.32.2 [Approved] Downloads cached for licensed users
pcwrunas 0.4.0.20161129 [Approved] Downloads cached for licensed
users
lightworks 14.0.0.20171007 [Approved]
3 packages found.
```

Untuk mengetahui informasi yang lebih lengkap mengenai perangkat lunak tersebut dapat ditambahkan flag -v pada command choco search.

```
Choco search hugo -v
```

## Install Perangkat Lunak Dengan Chocolatey

Berikut ini cara menggunakan chocolatey untuk menginstall dan update perangkat lunak yang bernama Hugo dan Python3. Hugo adalah sebuah static site generator yang ditulis menggunakan bahasa go lang.

```
choco install [packagename]
C:\WINDOWS\system32>choco install hugo
Chocolatey v0.10.8
Installing the following packages:
hugo
By installing you accept licenses for the packages.
Progress: Downloading hugo 0.32.2... 100%
hugo v0.32.2 [Approved]
hugo package files install completed. Performing other installation
steps.
The package hugo wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
```

```
Note: To confirm automatically next time, use '-y' or consider:  
choco feature enable -n allowGlobalConfirmation  
Do you want to run the script?([Y]es/[N]o/[P]rint): y  
Downloading hugo 64 bit  
...  
The install of hugo was successful.  
Software installed to 'C:\ProgramData\chocolatey\lib\hugo\tools'  
Chocolatey installed 1/1 packages.  
See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).
```

## Upgrade Perangkat Lunak Dengan Chocolatey

```
choco upgrade [packagename]  
C:\>choco upgrade hugo  
Chocolatey v0.10.8  
Upgrading the following packages:  
hugo  
By upgrading you accept licenses for the packages.  
hugo v0.32.2 is the latest version available based on your source(s).  
Chocolatey upgraded 0/1 packages.  
See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).
```

## Mengupgrade Chocolatey

Sebuah perangkat lunak tentunya mempunyai versi yang baru atau mengalami proses update, untuk mengupdate chocolatey dapat dilakukan dengan menggunakan perintah berikut ini.

```
choco upgrade chocolatey  
C:\>choco upgrade chocolatey  
Chocolatey v0.10.8  
Chocolatey detected you are not running from an elevated command shell  
(cmd/powershell).  
  
You may experience errors - many functions/packages  
require admin rights. Only advanced users should run choco w/out an  
elevated shell. When you open the command shell, you should ensure
```

that you do so with "Run as Administrator" selected. If you are attempting to use Chocolatey in a non-administrator setting, you must select a different location other than the default install location. See

<https://chocolatey.org/install#non-administrative-install> for details.

Do you want to continue?([Y]es/[N]o): Y

Upgrading the following packages:

chocolatey

By upgrading you accept licenses for the packages.

chocolatey v0.10.8 is the latest version available based on your source(s).

Chocolatey upgraded 0/1 packages.

See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).

Menampilkan Perangkat Lunak yang diinstall dengan chocolatey

```
choco list --local-only
C:\>choco list --local-only
Chocolatey v0.10.8
7zip 16.4.0.20170506
chocolatey 0.10.8
chocolatey-core.extension 1.3.1
hugo 0.31.1
4 packages installed.
```

Contoh penggunaan chocolatey

Setelah berhasil menginstall chocolatey selanjutnya anda dapat mempelajari beberapa tips dalam menginstall perangkat lunak dengan chocolatey. Berikut ini beberapa perintah yang dapat mempermudah proses installasi.

```
Choco install chrome -yes
```

Flag -y atau -yes akan membuat chocolatey menginstall tanpa mempertanyakan apakah yakin akan menginstall atau tidak.

```
Choco search Git -ev
```

Flag -ev akan menampilkan informasi detail mengenai package yang dicari namun hanya menampilkan perangkat lunak yang tulisannya benar benar sama dengan yang dicari. Selain itu juga dapat ditambahkan flag -a atau -all yang akan menampilkan semua versi dari perangkat lunak yang dicari.

```
C:\>choco search Git -ea
Chocolatey v0.10.8
git 2.15.1.2 [Approved]
git 2.15.1 [Approved]
git 2.15.0 [Approved]
git 2.14.3 [Approved]
git 2.14.2.20170927 [Approved]
git 2.14.2 [Approved]
...
...
```

Selain itu juga ada flag untuk mengatur urutan hasil pencarian dari yang terpopuler sampai yang terkecil tingkat popularitasnya.

```
Choco search python -order-by-popularity
```

## Install Php

Untuk menginstall PHP dapat dilakukan dengan menggunakan chocolatey, berikut ini langkah langkah yang perlu dilakukan dalam menginstall chocolatey.

1. Jalankan perintah berikut di windows terminal yang di run as administrator.

```
PS C:\Users\putra> choco install php
Chocolatey v1.1.0
2 validations performed. 1 success(es), 1 warning(s), and 0 error(s).
Installing the following packages:
php
By installing, you accept licenses for the packages.
Progress: Downloading php 8.1.9... 100%

php v8.1.9 [Approved]
php package files install completed. Performing other installation
steps.

The package php wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
```

```
Note: To confirm automatically next time, use '-y' or consider:  
choco feature enable -n allowGlobalConfirmation  
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint):  
A  
  
Extracting 64-bit C:\ProgramData\chocolatey\lib\php\tools\php-8.1.9-  
nts-Win32-vs16-x64.zip to C:\tools\php81...  
C:\tools\php81  
PATH environment variable does not have C:\tools\php81 in it.  
Adding...  
Creating default php.ini  
Configuring PHP extensions directory  
Please make sure you have CGI installed in IIS for local hosting  
Environment Vars (like PATH) have changed. Close/reopen your shell to  
see the changes (or in powershell/cmd.exe just type `refreshenv`).  
The install of php was successful.  
Software installed to 'C:\tools\php81'  
  
Chocolatey installed 1/1 packages.  
See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).
```

2. Jika proses instalasi berhasil dijalankan anda dapat melanjutkan dengan menutup dan membuka lagi windows terminal dan mengetikkan command php -v untuk mengecek versi php yang di install. Perhatikan bahwa laravel 9 membutuhkan versi php minimal 8.1.

```
PS C:\Users\putra> php -v  
PHP 8.1.9 (cli) (built: Aug 2 2022 14:17:33) (NTS Visual C++ 2019  
x64)  
Copyright (c) The PHP Group  
Zend Engine v4.1.9, Copyright (c) Zend Technologies  
PS C:\Users\putra>
```

3. Selamat anda sudah berhasil menginstall php

## Install Composer

Untuk menginstall Composer dapat dilakukan dengan menggunakan chocolatey, berikut ini langkah langkah yang perlu dilakukan dalam menginstall chocolatey.

1. Jalankan perintah berikut untuk menginstall composer menggunakan chocolatey pada windows terminal yang di run as administrator.

```
PS C:\Users\putra> choco install composer
Chocolatey v1.1.0
2 validations performed. 1 success(es), 1 warning(s), and 0 error(s).

Installing the following packages:
composer

By installing, you accept licenses for the packages.
Progress: Downloading composer 6.3.0... 100%

composer v6.3.0 [Approved]
composer package files install completed. Performing other
installation steps.

The package composer wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint):
A

Installing composer...
composer has been installed.

composer can be automatically uninstalled.

Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type `refreshenv`).
The install of composer was successful.

Software installed to 'C:\Program Files (x86)\ComposerSetup\'

Chocolatey installed 1/1 packages.
```

```
See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).
```

- Untuk memvalidasi hasil installasi tutup dan buka kembali terminal kemudian ketikkan perintah berikut ini.

```
PS C:\Users\putra> composer -V  
Composer version 2.4.1 2022-08-20 11:44:50
```

- Selamat anda telah berhasil melakukan installasi Composer menggunakan chocolatey.

## Install Laravel

Untuk menginstall laravel kita dapat menggunakan perintah composer setelah sebelumnya menginstall composer. Lakukanlah langkah berikut ini untuk menginstall project laravel baru.

- Jalankan perintah composer berikut untuk menginstall laravel.

```
composer create-project laravel/laravel example-app
```

- Untuk proses installasi ini membutuhkan koneksi internet, setelah mengetik command diatas tunggu lah beberapa saat sampai proses installasi selesai.
- Jika dilakukan pertama kali akan sedikit lama karena banyak library yang di download, namun untuk langkah selanjutnya bisa lebih cepat karena library yang selanjutnya tidak di download ulang.

## Install Mysql

Untuk menginstall Mysql dapat dilakukan dengan menggunakan chocolatey berikut ini langkah langkah yang perlu dilakukan dalam menginstall mysql.

- Jalankan Perintah berikut di windows terminal yang di run as administrator.

```
PS C:\Users\putra> choco install mysql  
Chocolatey v1.1.0  
Installing the following packages:  
mysql  
By installing, you accept licenses for the packages.  
Progress: Downloading vcredist2015 14.0.24215.20170201... 100%  
Progress: Downloading mysql 8.0.28... 100%  
  
vcredist2015 v14.0.24215.20170201 [Approved]
```

```
vcredist2015 package files install completed. Performing other
installation steps.

The install of vcredist2015 was successful.

Software installed to 'C:\ProgramData\chocolatey\lib\vcredist2015'

mysql v8.0.28 [Approved]
mysql package files install completed. Performing other installation
steps.

The package mysql wants to run 'chocolateyInstall.ps1'.

Note: If you don't run this script, the installation will fail.

Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation

Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint):
```

2. Chocolatey akan menginstall dan mendownload mysql secara otomatis.

```
Adding 'C:\tools\mysql\current\bin' to the path and the current shell
path

PATH environment variable does not have C:\tools\mysql\current\bin in
it. Adding...

Downloading mysql 64 bit
  from  'https://dev.mysql.com/get/Downloads/MySQL-8.0/mysql-8.0.28-
winx64.zip'
Progress: 15% - Saving 32.66 MB of 211.74 MB
```

3. Dalam proses installasi akan muncul beberapa log yang panjang dan jika berhasil akan tampil seperti dibawah ini.

```
Only an exit code of non-zero will fail the package by default. Set
`--failonstderr` if you want error messages to also fail a script.

See
`choco -h` for details.

Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type `refreshenv`).

The install of mysql was successful.

Software installed to 'C:\tools\mysql'

Chocolatey installed 2/2 packages.
```

See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).

4. Jika berhasil anda dapat menggunakan command mysql pada windows terminal untuk login ke database mysql.
5. Sebelum menggunakan mysql tutuplah windows terminal anda terlebih dahulu.
6. Untuk login jalankan perintah berikut ini.

```
PS C:\Users\putra> mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.28 MySQL Community Server - GPL
Copyright (c) 2000, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

7. Selamat anda sudah berhasil menginstall database mysql di local machine anda.
8. Lakukan command berikut untuk melihat database yang ada pada mysql anda.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> show databases;
+-----+
| Database           |

```

```
+-----+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+-----+
4 rows in set (0.03 sec)

mysql>
```

## Install Vscode

Untuk menginstall Visual Studio Code (VsCode) dapat dilakukan dengan menggunakan chocolatey berikut ini langkah langkah yang perlu dilakukan dalam menginstall vscode.

1. Jalankan perintah berikut ini pada windows terminal yang sudah di run as administrator.

```
PS C:\WINDOWS\system32> choco install vscode
Chocolatey v1.1.0
2 validations performed. 1 success(es), 1 warning(s), and 0 error(s).
```

Validation Warnings:

Installing the following packages:

vscode

By installing, you accept licenses for the packages.

Progress: Downloading vscode.install 1.70.2... 100%

Progress: Downloading vscode 1.70.2... 100%

vscode.install v1.70.2 [Approved]

vscode.install package files install completed. Performing other installation steps.

The package vscode.install wants to run 'ChocolateyInstall.ps1'.

Note: If you don't run this script, the installation will fail.

Note: To confirm automatically next time, use '-y' or consider:

choco feature enable -n allowGlobalConfirmation

```
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint):  
A  
  
Merge      Tasks:      !runCode,      desktopicon,      quicklaunchicon,  
addcontextmenufiles,      addcontextmenufolders,      associatewithfiles,  
addtopath  
Downloading vscode.install 64 bit  
from  
'https://az764295.vo.msecnd.net/stable/e4503b30fc78200f846c62cf8091b  
76ff5547662/VSCodeSetup-x64-1.70.2.exe'  
Progress:      100%      -      Completed      download      of  
C:\Users\putra\AppData\Local\Temp\chocolatey\vscode.install\1.70.2\V  
SCodeSetup-x64-1.70.2.exe (79.04 MB).  
Download of VSCodeSetup-x64-1.70.2.exe (79.04 MB) completed.  
Hashes match.  
Installing vscode.install...  
vscode.install has been installed.  
vscode.install can be automatically uninstalled.  
Environment Vars (like PATH) have changed. Close/reopen your shell to  
see the changes (or in powershell/cmd.exe just type `refreshenv`).  
The install of vscode.install was successful.  
Software installed to 'C:\Program Files\Microsoft VS Code\'  
  
vscode v1.70.2 [Approved]  
vscode package files install completed. Performing other installation  
steps.  
The install of vscode was successful.  
Software installed to 'C:\ProgramData\chocolatey\lib\vscode'  
  
Chocolatey installed 2/2 packages.  
See      the      log      for      details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).
```

2. Untuk memvalidasi silahkan buka postman melalui menu windows.

## Install Postman

Untuk menginstall Postman dapat dilakukan dengan menggunakan chocolatey berikut ini langkah langkah yang perlu dilakukan dalam menginstall postman.

1. Jalankan perintah berikut ini pada windows terminal yang sudah di run as administrator.

```
PS C:\WINDOWS\system32> choco install postman
Chocolatey v1.1.0
2 validations performed. 1 success(es), 1 warning(s), and 0 error(s).
Installing the following packages:
postman
By installing, you accept licenses for the packages.
Progress: Downloading postman 9.27.0... 100%

postman v9.27.0 [Approved]
postman package files install completed. Performing other installation
steps.

The package postman wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint):
A

Downloading postman 64 bit
  from 'https://dl.pstmn.io/download/version/9.27.0/win64'
Progress:      100%      -      Completed      download      of
C:\Users\putra\AppData\Local\Temp\chocolatey\postman\9.27.0\Postman-
win64-9.27.0-Setup.exe (146.59 MB).
Download of Postman-win64-9.27.0-Setup.exe (146.59 MB) completed.
Hashes match.
Installing postman...
postman has been installed.

The install of postman was successful.

Software installed as 'exe', install location is likely default.
```

```
Chocolatey installed 1/1 packages.  
See the log for details  
(C:\ProgramData\chocolatey\logs\chocolatey.log).
```

2. Untuk memvalidasi silahkan buka postman melalui menu windows.

## Mulai Membuat REST Api dengan Laravel

### Create Project Laravel

Untuk membuat Project Rest API dapat dimulai dengan membuat sebuah project Laravel baru. Cara membuat project laravel baru dapat dilakukan dengan beberapa cara yaitu menggunakan composer dan menggunakan laravel installer. Pada buku ini anda akan menggunakan laravel installer untuk membuat project laravel. Berikut ini langkah langkah membuat project laravel menggunakan laravel installer :

1. Buka terminal
2. Jalankan perintah  
`laravel new rest-api`
3. Command tersebut akan mendownload dependency yang dibutuhkan dan akan membuatkan folder baru dengan nama 'rest-api'.
4. Bukanlah folder tersebut dengan perintah berikut

```
cd rest-api
```

5. Didalam folder rest-api ini jalankan command berikut

```
php artisan serve
```

6. Bukanlah browser di alamat <http://localhost:8000/> untuk melihat project laravel yang sudah di buat.

Selanjutnya setelah berhasil membuat project laravel langkah yang tidak kalah penting adalah melakukan konfigurasi agar project rest-api yang dibuat dapat terhubung ke database. (Vallejo, n.d.)

### Konfigurasi Project Laravel

Berikut ini adalah langkah langkah yang perlu dilakukan untuk mengkonfigurasi project laravel yang dibuat. Semua konfigurasi dalam project laravel disimpan pada sebuah file yang bernama .env. File ini berada pada root folder dari project laravel rest-api, yang perlu

diperhatikan file .env ini kadang kala secara default disembunyikan oleh sistem operasi jika anda tidak menemukan file ini lakukan konfigurasi terlebih dahulu agar explorer sistem operasi anda dapat menampilkan file ini. Berikut ini contoh posisi file .env pada terminal.

drwxr-xr-x	27	putraprima	staff	864	Jan 13 16:41	.
drwxr-xr-x	12	putraprima	staff	384	Jan 13 16:31	..
-rw-r--r--	1	putraprima	staff	258	Dec 22 17:07	.editorconfig
-rw-r--r--	1	putraprima	staff	965	Jan 13 16:35	.env
-rw-r--r--	1	putraprima	staff	900	Jan 13 16:31	.env.example
drwxr-xr-x	12	putraprima	staff	384	Jan 16 06:50	.git
-rw-r--r--	1	putraprima	staff	111	Dec 22 17:07	.gitattributes
-rw-r--r--	1	putraprima	staff	207	Dec 22 17:07	.gitignore
-rw-r--r--	1	putraprima	staff	194	Dec 22 17:07	.styleci.yml
-rw-r--r--	1	putraprima	staff	269	Jan 16 08:20	README.md
drwxr-xr-x	7	putraprima	staff	224	Dec 22 17:07	app
-rwxr-xr-x	1	putraprima	staff	1686	Dec 22 17:07	artisan
drwxr-xr-x	4	putraprima	staff	128	Dec 22 17:07	bootstrap
-rw-r--r--	1	putraprima	staff	1745	Jan 16 06:35	composer.json
-rw-r--r--	1	putraprima	staff	289868	Jan 13 16:45	composer.lock
drwxr-xr-x	17	putraprima	staff	544	Dec 22 17:07	config
drwxr-xr-x	6	putraprima	staff	192	Dec 22 17:07	database
-rw-r--r--	1	putraprima	staff	473	Dec 22 17:07	package.json
-rw-r--r--	1	putraprima	staff	1202	Dec 22 17:07	phpunit.xml
drwxr-xr-x	6	putraprima	staff	192	Dec 22 17:07	public
drwxr-xr-x	6	putraprima	staff	192	Dec 22 17:07	resources
drwxr-xr-x	6	putraprima	staff	192	Dec 22 17:07	routes
-rw-r--r--	1	putraprima	staff	563	Dec 22 17:07	server.php
drwxr-xr-x	5	putraprima	staff	160	Dec 22 17:07	storage
drwxr-xr-x	6	putraprima	staff	192	Dec 22 17:07	tests
drwxr-xr-x	45	putraprima	staff	1440	Jan 13 16:31	vendor
-rw-r--r--	1	putraprima	staff	559	Dec 22 17:07	webpack.mix.js

Laravel juga menyertakan sebuah file .env.example sebagai template contoh file konfigurasi .env yang dapat ditiru. Jika menginstall menggunakan laravel installer file .env ini sudah secara otomatis di buatkan oleh installer sedangkan jika menginstall menggunakan

composer biasa anda harus melakukan copy dan paste secara manual untuk file konfigurasi ini. Berikut ini contoh dari isi file .env yang digunakan pada project rest-api.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:V+IM9mYi8uaDV3Fyaffyt8TPeMh763drdDlo/2yBW10=
APP_DEBUG=true
APP_URL=http://localhost:8000

...
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=rest_api
DB_USERNAME=root
DB_PASSWORD=putraprima

...
```

Perhatikan pada potongan kode dari file .env diatas ada yang diberi warna abu abu, lakukanlah modifikasi pada kode program tersebut agar sesuai dengan development environment di laptop / pc anda. Berikut ini daftar yang harus disesuaikan :

1. APP\_NAME : Disesuaikan dengan nama aplikasi yang anda inginkan
2. APP\_URL : Disesuaikan dengan url aplikasi anda dalam hal ini gunakan <http://localhost:8000> sesuai dengan output dari php artisan serve.
3. DB\_DATABASE : Gunakan database yang sudah ada pada database mysql di lokal pc / laptop anda jika belum ada buatlah database tersebut terlebih dahulu.
4. Untuk membuat database mysql pada local komputer anda gunakan perintah berikut.
5. Login ke database mysql dengan username dan password mysql anda.

```
mysql -u root -p
```

Perhatikan user yang digunakan pada perintah di atas adalah user root sesuaikan user ini dengan user yang anda miliki.

```
mysql -u root -p
```

```
Enter password:
```

Selanjutnya sistem akan meminta password, tuliskanlah password anda kemudian tekan enter. Berikut ini tampilan terminal anda jika berhasil login ke mysql.

```
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 9  
Server version: 8.0.27 Homebrew  
  
Copyright (c) 2000, 2021, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input  
statement.  
  
mysql>
```

Selanjutnya ketikkan perintah berikut pada console mysql untuk membuat database baru :

```
create database rest_api;
```

Berikut ini tampilan console mysql jika berhasil membuat database

```
mysql> create database rest_api;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql>
```

Untuk memastikan database sudah berhasil di create dapat anda lakukan perintah berikut:

```
show databases;
```

Perintah ini akan menampilkan semua database yang anda miliki pada database mysql anda.

Database
rest_api

```
| safepas          |
+-----+
2 rows in set (0.00 sec)
```

6. DB\_USERNAME : isikan dengan username dari database lokal anda (Perhatikan pada buku ini menggunakan user root tapi pada proses deploy tidak disarankan untuk menggunakan user root sebagai user database production)
7. DB\_PASSWORD : password dari user pada database mysql yang anda gunakan.

#### Penjelasan routes web.php vs api.php

Sejak laravel 8 sebuah project laravel akan memiliki beberapa routes file, file ini berada pada folder routes. Folder ini berisi beberapa file yaitu api.php, channels.php, console.php dan web.php.

```
└── resources
    ├── css
    ├── js
    ├── lang
    └── views
└── routes
    ├── api.php
    ├── channels.php
    ├── console.php
    └── web.php
```

Masing masing file routes memiliki fungsi masing masing, file web.php berisi routes yang dapat diakses melalui browser, file api.php berisi routes khusus yang dapat diakses melalui REST API sedangkan channels dan console merupakan route lain yang dapat anda pelajari fungsinya pada dokumentasi laravel. Pada buku ini akan difokuskan pada routes api.php.

#### Membuat Endpoint GET dan Mengakses dari Postman

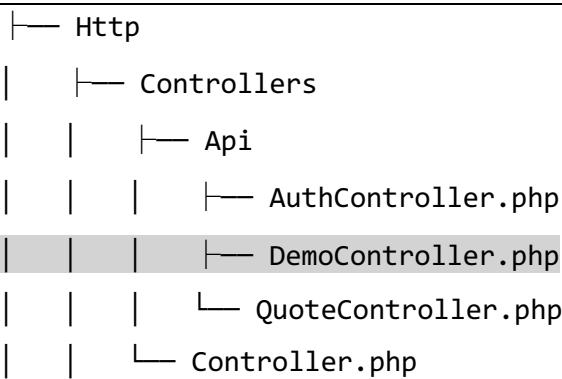
Untuk mulai membuat sebuah Endpoint GET dapat dimulai dengan menambahkan sebuah controller dan menyambungkannya ke sebuah routes dai dalam file api.php. Endpoint

GET berguna sebagai endpoint yang dapat diakses oleh aplikasi lain dimana server REST API yang dibuat akan mengembalikan hasil menggunakan format Json. Hasil yang dikembalikan oleh REST API dapat berasal dari database, hasil pertitungan atau text statis biasa. Pada sub bab ini akan kita pelajari cara membuat sebuah endpoint GET dan mengaksesnya menggunakan postman dimana endpoint tersebut mengambilkan sebuah data statis berformat json. Berikut ini langkah-langkah yang dibutuhkan untuk membuat sebuah endpoint GET pada Rest Api Laravel.

1. Bukalah project laravel menggunakan Visual Studio Code kemudian buka terminal pada Visual Studio Code.
2. Ketikkan perintah berikut ke console

```
php artisan make:controller Api/DemoController
```

3. Perintah di atas akan menggeneratekan sebuah controller di folder app/Http/Controller/Api dengan nama DemoController.



4. Berikut ini isi standar dari file Demo Controller.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class DemoController extends Controller
{
    //
}
```

5. Controller ini belum melakukan apapun, untuk bisa memberikan response terhadap sebuah request kita harus menambahkan fungsi dan kode program untuk mengembalikan sebuah data. Buatlah sebuah function dengan nama index dan isikan kode program dibawah ini.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class DemoController extends Controller
{
    //
    public function index()
    {
        $data = [
            'name' => 'Demo',
            'version' => '1.0.0',
        ];
        return $data;
    }
}
```

6. Selanjutnya agar dapat diakses melalui sebuah endpoint REST Api maka endpoint tersebut harus di registrasikan pada file api.php. bukalah file api.php kemudian tambahkan kode program berikut.

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\DemoController;
use App\Http\Controllers\Api\QuoteController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
```

```

| -----
| -----
| API Routes
| -----
| -----
| 
| Here is where you can register API routes for your application.
| These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

```

`Route::get('/demo', [DemoController::class, 'index']);`

- Kode program diatas menambahkan sebuah endpoint <http://localhost:8000/api/demo> ke aplikasi laravel dan menunjuk class DemoController dengan function index untuk menghandle request terhadap endpoint tersebut.
- Untuk menguji apakah endpoint sudah dapat bekerja dengan baik maka bukalah aplikasi postman kemudian isikan url dengan konfigurasi seperti pada gambar dibawah ini.

Method	Url	Param
GET	localhost:8000/api/demo	-

Params    Authorization    Headers (8)    Body    Pre-request Script    Tests    Settings

Headers    Hide auto-generated headers

KEY	VALUE
<input checked="" type="checkbox"/> Cache-Control ⓘ	no-cache
<input checked="" type="checkbox"/> Postman-Token ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> Host ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent ⓘ	PostmanRuntime/7.28.4
<input type="checkbox"/> Accept ⓘ	*/*
<input checked="" type="checkbox"/> Accept-Encoding ⓘ	gzip, deflate, br
<input checked="" type="checkbox"/> Connection ⓘ	keep-alive
<input checked="" type="checkbox"/> Accept	application/json

- Perhatikan pada bagian header ditambahkan header Accept yang value nya application/json.



The screenshot shows the Postman interface with the 'Body' tab selected. The response is displayed in 'Pretty' format as a JSON object:

```
1 {  
2   "name": "Demo",  
3   "version": "1.0.0"  
4 }
```

10. Berikut ini hasil dari response dengan request GET ke endpoint /api/demo

#### Membuat Endpoint POST dan Mengakses dari Postman

Untuk mulai membuat sebuah Endpoint POST prosesnya mirip dengan cara membuat endpoint GET, yang menjadi perbedaan mendasar adalah di file api.php route yang dibuat bukan lagi menggunakan method get namun menggunakan method POST. Berikut ini langkah langkah yang dibutuhkan untuk membuat sebuah endpoint POST pada Rest Api Laravel.

1. Buka file DemoController.php kemudian tambahkan sebuah function dengan nama store dimana pada function ini kita akan pura-pura menyimpan data yang dikirim melalui endpoint post ke database.

```
<?php  
  
namespace App\Http\Controllers\Api;  
  
use App\Http\Controllers\Controller;  
use Illuminate\Http\Request;  
  
class DemoController extends Controller  
{  
    ...  
    public function store(Request $request)  
    {  
        $data = [  
            'name' => $request->name,  
            'version' => $request->version,  
        ];  
        return $data;  
    }  
}
```

```
...
}
```

2. Pada function store kita menggunakan class Request untuk menyimpan parameter dari request yang dikirim ke endpoint. Kemudian membuat sebuah variabel dengan nama \$data yang akan menampung \$request->name ke array name dan \$request->version ke array version dari variabel \$data. Dibagian akhir kita me return kan \$data ke user.
3. Kemudian pada file api.php tambahkan kode program berikut untuk menambahkan endpoint post ke function store.

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\DemoController;
use App\Http\Controllers\Api\QuoteController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
| -----
| API Routes
| -----
| -----
| |
| Here is where you can register API routes for your application.
| These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/
Route::get('/demo', [DemoController::class, 'index']);
Route::post('/demo', [DemoController::class, 'store']);
```

4. Untuk menguji endpoint ini pada postman buatlah sebuah request baru dengan parameter sebagai berikut.

Method	Url	Param
GET	localhost:8000/api/demo	-

Buku Laravel Rest Api / Demo / Post Request

POST localhost:8000/api/demo

Headers (10)

KEY	VALUE
Cache-Control	no-cache
Postman-Token	<calculated when request is sent>
Content-Type	multipart/form-data; boundary=<calculated when request is sent>
Content-Length	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.28.4
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Accept	application/json

Buku Laravel Rest Api / Demo / Post Request

POST localhost:8000/api/demo

Body

KEY	VALUE
tes	tes
tos	tos
name	soko
version	1.1.1
Key	Value

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON JSON

```

1 {
2   "name": "soko",
3   "version": "1.1.1"
4 }
```

5. Perhatikan konfigurasi pada postman, request yang digunakan memakai POST, endpoint dan body menggunakan x-www-form-urlencoded dengan beberapa key value yang sudah di setting.
6. Perhatikan juga di bagian body kita menambahkan key tes dan tos dimana key ini tetap dikirim namun oleh logika pada function store key ini tidak di proses dan tidak di response kan ke REST Api.

#### Membuat Endpoint PUT dan Mengakses dari Postman

Untuk mulai membuat sebuah Endpoint PUT prosesnya mirip dengan cara membuat endpoint GET, yang menjadi perbedaan mendasar adalah di file api.php route yang dibuat bukan lagi menggunakan method get namun menggunakan method PUT. Berikut ini langkah langkah yang dibutuhkan untuk membuat sebuah endpoint PUT pada Rest Api Laravel.

1. Buka file DemoController.php kemudian tambahkan sebuah function dengan nama update dimana pada function ini kita akan pura-pura mengubah data yang dikirim dari REST API kemudian meresponsekan kembali dalam format json.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class DemoController extends Controller
{
    ...
    public function update(Request $request)
    {
        $data = [
            'name' => $request->name . '-updated',
            'version' => $request->version . '-updated',
        ];
        return $data;
    }
    ...
}
```

```
}
```

2. Pada function update kita merubah input name dan version dari user menjadi \$request->name ditambahkan dengan string '-updated'.
3. Setelah function dibuat jangan lupa menambahkan route untuk function update pada api.php.

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\DemoController;
use App\Http\Controllers\Api\QuoteController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
| -----
| API Routes
| -----
| -----
| Here is where you can register API routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/
Route::get('/demo', [DemoController::class, 'index']);
Route::post('/demo', [DemoController::class, 'store']);
Route::put('/demo', [DemoController::class, 'update']);
```

4. Kali ini pada route demo ditambahkan route dengan method put ke function update dari class DemoController.
5. Untuk menguji di postman dapat digunakan konfigurasi sebagai berikut

Method	Url	Param
--------	-----	-------

PUT	localhost:8000/api/demo	-
-----	-------------------------	---

Buku Laravel Rest Api / Demo / Put Request

PUT  localhost:8000/api/demo

Params Authorization Headers (10) **Body**  Pre-request Script Tests Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> tes	tes
<input checked="" type="checkbox"/> tos	tos
<input checked="" type="checkbox"/> name	soko
<input checked="" type="checkbox"/> version	1.0.1
Key	Value

Buku Laravel Rest Api / Demo / Put Request

PUT  localhost:8000/api/demo

Params Authorization Headers (10) **Body**  Pre-request Script Tests Settings

**KEY** **VALUE**

<input checked="" type="checkbox"/> Cache-Control ⓘ	no-cache
<input checked="" type="checkbox"/> Postman-Token ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> Content-Type ⓘ	application/x-www-form-urlencoded
<input checked="" type="checkbox"/> Content-Length ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> Host ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent ⓘ	PostmanRuntime/7.28.4
<input type="checkbox"/> Accept ⓘ	*/*
<input checked="" type="checkbox"/> Accept-Encoding ⓘ	gzip, deflate, br
<input checked="" type="checkbox"/> Connection ⓘ	keep-alive
<input checked="" type="checkbox"/> Accept	application/json
Key	Value

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1  {
2      "name": "soko-updated",
3      "version": "1.0.1-updated"
4 }
```

6. Perhatikan bahwa dengan input name = soko dan version = 1.0.1 endpoint put mengembalikan response dengan tambahan string '-updated'

#### Membuat Endpoint DELETE dan Mengakses dari Postman

Proses yang sama dapat digunakan untuk membuat endpoint DELETE, pada endpoint DELETE biasanya mengembalikan response REST API berupa response no content dimana sebelum melakukan response dilakukan proses delete data. Untuk membuat endpoint DELETE ikutilah langkah langkah berikut ini :

1. Buka file DemoController.php kemudian tambahkan function Delete seperti pada potongan kode program dibawah ini.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class DemoController extends Controller
{
    ...
    public function delete(Request $request)
    {
        return response()->noContent();
    }
}
```

```
...
}
```

2. Selanjutnya bukalah file api.php kemudian tambahkan route DELETE.

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\DemoController;
use App\Http\Controllers\Api\QuoteController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
| -----
| API Routes
| -----
| -----
| Here is where you can register API routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/
Route::get('/demo', [DemoController::class, 'index']);
Route::post('/demo', [DemoController::class, 'store']);
Route::put('/demo', [DemoController::class, 'update']);
Route::delete('/demo', [DemoController::class, 'delete']);
```

3. Untuk menguji pada postman lakukan dengan konfigurasi berikut ini

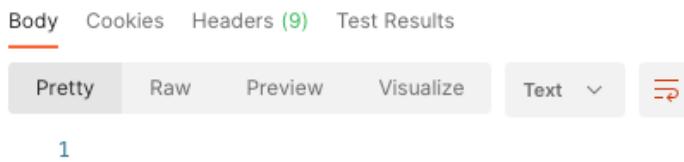
Method	Url	Param
DELETE	localhost:8000/api/demo	-

Buku Laravel Rest Api / Demo / Delete Request ⚙ ⚙

DELETE	▼	localhost:8000/api/demo				
Params	Authorization	Headers (10)	Body	Pre-request Script	Tests	Settings
KEY						VALUE
<input checked="" type="checkbox"/> Cache-Control ⓘ						no-cache
<input checked="" type="checkbox"/> Postman-Token ⓘ						<calculated when request is sent>
<input checked="" type="checkbox"/> Content-Type ⓘ						application/x-www-form-urlencoded
<input checked="" type="checkbox"/> Content-Length ⓘ						<calculated when request is sent>
<input checked="" type="checkbox"/> Host ⓘ						<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent ⓘ						PostmanRuntime/7.28.4
<input type="checkbox"/> Accept ⓘ						*/*
<input checked="" type="checkbox"/> Accept-Encoding ⓘ						gzip, deflate, br
<input checked="" type="checkbox"/> Connection ⓘ						keep-alive
<input checked="" type="checkbox"/> Accept						application/json
Key						Value

Buku Laravel Rest Api / Demo / Delete Request

DELETE	▼	localhost:8000/api/demo				
Params	Authorization	Headers (10)	Body	Pre-request Script	Tests	Settings
<input type="radio"/> none	<input type="radio"/> form-data	<input checked="" type="radio"/> x-www-form-urlencoded	<input type="radio"/> raw	<input type="radio"/> binary	<input type="radio"/> GraphQL	
KEY						VALUE
<input checked="" type="checkbox"/> tes						tes
<input checked="" type="checkbox"/> tos						tos
<input checked="" type="checkbox"/> name						soko
<input checked="" type="checkbox"/> version						1.0.1
Key						Value



4. Untuk endpoint DELETE best practice nya memang hanya mengembalikan response yang sifatnya no content jika proses delete berhasil.

### Mini Project REST API BMI Calculator

Untuk melatih pemahaman mengenai cara membuat rest api mari kita buat sebuah mini project sederhana yaitu Rest API untuk BMI Calculator.

#### Project Overview

Pada project REST API BMI Calculator ini dibutuhkan beberapa requirement sebagai berikut :

1. REST API mengembalikan data berupa nilai bmi dalam angka dan status BMI dalam bentuk underweight, normal, overweight dan obese.
2. Input yang dikirimkan ke REST API adalah dalam berat dalam satuan kilogram, dan tinggi dalam satuan centimeter.
3. Rumus perhitungan BMI menurut sumber berikut ini adalah  
[https://www.diabetes.ca/managing-my-diabetes/tools---resources/body-mass-index-\(bmi\)-calculator](https://www.diabetes.ca/managing-my-diabetes/tools---resources/body-mass-index-(bmi)-calculator)  
BMI = kg/m<sup>2</sup>
4. Input dari rest api di terima dalam method POST di endpoint /api/bmi.

#### Mendesain REST API

Untuk mempermudah desain REST API requirement tersebut dapat di translasi menjadi sebuah tabel. Berikut ini tabel translasi dari requirement.

REST Api BMI Calculator
-------------------------

Method	POST
Parameter Input	weight (dalam kg) height (dalam cm)
Response	{ “bmi”: 25.35, “status”: “Overweight” }

### Membuat Controller

Berikut ini langkah langkah yang dibutuhkan untuk membuat controller pada mini project ini.(Griffin, 2021)

1. Untuk mengakomodir kebutuhan dari REST API Bmi Calculator buatlah sebuah controller baru dengan nama BmiCalculator dengan menjalankan perintah berikut ini pada terminal.

```
php artisan make:controller Api/BmiCalculator
```

2. Setelah controller terbentuk buatlah sebuah method dengan nama index yang memiliki parameter Request dari laravel.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class BmiController extends Controller
{
    //
    public function index(Request $request)
    {
        //
    }
}
```

3. Selanjutnya tambahkan implementasi program untuk menghitung BMI berdasarkan rumus  $bmi = \frac{kg}{m^2}$ .

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class BmiController extends Controller
{
    //
    public function index(Request $request)
    {
        $result = [];
        $bmi = round($request->weight / ((($request->height / 100) *
        ($request->height / 100)), 2);
        $result['bmi'] = $bmi;
    }
}
```

4. Pada kode program diatas dibuat sebuah variabel \$result yang berupa array yang akan menampung hasil perhitungan \$bmi dan akan ditambahkan key array ‘status’ untuk menyimpan status \$bmi.
5. Pada perhitungannya digunakan round untuk menghitung sampai 2 angka dibelakang koma.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class BmiController extends Controller
{
    //
}
```

```

public function index(Request $request)
{
    ...
    if ($bmi <= 18.5) {
        $result['status'] = 'Underweight';
        return $result;
    }

    if ($bmi > 18.5 && $bmi <= 24.9) {
        $result['status'] = 'Normal';
        return $result;
    }

    if ($bmi > 24.9 && $bmi <= 29.9) {
        $result['status'] = 'Overweight';
        return $result;
    }

    if ($bmi > 29.9) {
        $result['status'] = 'Obese';
        return $result;
    }
}

```

6. Selanjutnya tambahkan pengecekan if untuk membagi response berdasarkan hasil perhitungan Bmi

#### Testing REST API

Untuk menguji REST Api yang dibuat bukalah aplikasi postman kemudian buat sebuah rest api baru dengan konfigurasi seperti pada gambar dibawah ini :

Buku Laravel Rest Api / Bmi / Calculate BMI

POST ▾ http://localhost:8000/api/bmi

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> weight	75 ↴
<input checked="" type="checkbox"/> height	172
Key	Value

Buku Laravel Rest Api / Bmi / Calculate BMI

POST ▾ http://localhost:8000/api/bmi

Params Authorization **Headers (10)** Body Pre-request Script Tests Settings

KEY	VALUE
<input checked="" type="checkbox"/> Cache-Control	no-cache
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>
<input checked="" type="checkbox"/> Content-Type	application/x-www-form-urlencoded
<input checked="" type="checkbox"/> Content-Length	<calculated when request is sent>
<input checked="" type="checkbox"/> Host	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.0
<input checked="" type="checkbox"/> Accept	text/html
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/> Connection	keep-alive
<input checked="" type="checkbox"/> Accept	application/json
Key	Value

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1  {
2      "bmi": 25.35,
3      "status": "Overweight"
4 }
```

Lakukanlah pengujian dengan beberapa nilai agar dapat menguji semua status bmi yang diberikan

#### Challenge

Ada beberapa hal yang perlu di improve pada mini project ini silahkan men challenge diri anda untuk memperbaiki rest api bmi menjadi lebih baik lagi

1. Input untuk variasi ukuran karena ada negara yang menggunakan satuan berat lb dan satuan tinggi inch.
2. Validasi input untuk parameter tinggi dan berat harus berupa angka dan tidak boleh kosong.

## Mini Project REST API Quote App

### Project Overview

Pada project ini akan dibuat sebuah REST API yang dapat menampilkan quote ke rest api, pada project ini anda akan belajar bagaimana membuat sebuah public REST API endpoint yang terhubung dengan database.

#### Mendesain Database

Langkah pertama yang perlu dilakukan adalah mendesain database, untuk database pada project ini cukup sederhana. Project ini dapat diselesaikan dengan menggunakan satu tabel berisi data Quote yang ingin ditampilkan. Berikut ini desain tabel pada REST API Quote.

Kolom	Tipe Data
-------	-----------

Id	Int(Primary key)
text	text
author	string
created_at	timestamp
updated_at	timestamp

Untuk mempermudah proses development anda perlu menerjemahkan tabel ini ke migration di laravel. Untuk melakukan migration dapat dibuat dengan menggunakan beberapa cara, salah satu caranya adalah mengenerate file migration sekalian dengan mengenerate model, factory dan seeder nya. Jalankan perintah berikut pada terminal untuk mengenerate semua file yang dibutuhkan.

```
php artisan make:model Quote -mfs
```

Perintah diatas akan memerintahkan laravel mengenerate beberapa file yaitu file model, model, factory, seeder dan migration dengan nama sebagai berikut :

Model	App/Models/Quote.php
Factory	Database/Factories/QuoteFactory.php
Seeder	Database/Seeders/QuoteSeeder.php
Migration	Database/Migrations/xxxx_xx_xxx_create_quote_tables.php

#### Membuat Model

File yang digenerate ini adalah file template yang perlu kita modifikasi, untuk melakukan modifikasi dapat dimulai dengan model pada folder app/models/Quote.php. File model pada laravel adalah salah satu file yang penting untuk dikuasai karena akan menjadi dasar dari data pada aplikasi, seperti defenisi relasi dan proses object relational mapping lainnya. Pada buku ini kita tidak fokus membahas tentang pemodelan data pada laravel, namun anda perlu mengetahui apa isi dari file model ini. Berikut ini isi dari file Quote.php pada saat awal dibuat.

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Quote extends Model
{
    use HasFactory;
}
```

Untuk dapat menggunakan model ini dengan benar kita perlu menambahkan properties fillable pada file model sesuai dengan desain yang kita buat.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Quote extends Model
{
    use HasFactory;
    protected $fillable = ['text', 'author'];
}
```

Perhatikan pada attribute \$fillable ditambahkan array dengan dua value yaitu text dan author sesuai dengan nama kolom yang di desain.

#### Membuat Migration

File yang digenerate ini adalah file template yang perlu kita modifikasi, untuk melakukan modifikasi dapat dimulai dengan membuka file migration yaitu xxxx\_xx\_xxx\_create\_quote\_table.php. berikut ini isi default dari file migration nya.

```
<?php

use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

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

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

Kita sebagai programmer laravel harus menambahkan secara manual kolom apa saja yang akan kita butuhkan pada kode program migration tersebut. Berdasarkan desain yang sudah dibuat maka akan dibutuhkan dua kolom tambahan yaitu text dan author.

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateQuotesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('quotes', function (Blueprint $table) {
            $table->id();
            $table->string('text');
            $table->string('author');
            $table->timestamps();
        });
    }

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

Untuk menguji apakah migration ini berhasil dan bekerja dengan baik maka lakukanlah perintah berikut untuk menjalankan migration pada project Laravel.

```
php artisan migrate:fresh
```

Perintah ini akan mengulang kembali semua migration dan mengenerate ulang tabel yang sudah di desain pada database. Bukalah client mysql anda kemudian perhatikan apakah sudah terdapat tambahan tabel Quote yang sesuai dengan spesifikasi yang kita inginkan.

The screenshot shows the MySQL Workbench interface with the following details:

- Database:** rest\_api
- Table:** quotes
- Structure Tab:** The current tab is 'Structure'. Other tabs include 'Data', 'Structure', '+ Index', '+ Column', 'Triggers', and 'Info'.
- Columns:** The table has five columns: id, text, author, created\_at, and updated\_at. The 'id' column is defined as bigint unsigned, primary key, and auto-increment.
- Indexes:** A primary index is defined on the 'id' column using a BTREE algorithm.

The screenshot shows the MySQL Workbench interface. The title bar reads "MySQL 8.0.27 : local : rest\_api : quotes". The left sidebar shows the database structure with "simakre" and "rest\_api" databases. Under "rest\_api", there are "Functions", "Tables" (with "failed\_jobs", "migrations", "password\_resets", "personal\_access\_tokens", "quotes", and "users" listed), and a search bar "Search for item...". The main area displays the "quotes" table with columns: id, text, author, created\_at, and updated\_at. The table has 0 rows. At the bottom, there are buttons for "Data", "Structure", "+ Row", and a timestamp "2022-02-03 05:53:53.2960". The SQL query "SELECT \* FROM `rest\_api`.`quotes` ORDER BY `id` LIMIT 300 OFFSET 0;" is shown at the bottom.

## Membuat Factory

Perhatikan pada proses migration yang dilakukan saat ini terlihat bahwa tabel yang di generate belum memiliki data apapun, maka untuk mengenerate data yang dapat digunakan ketika membuat aplikasi kita perlu menambahkan sebuah file yang bernama factory, dimana file ini akan menjadi pabrik yang dapat mengenerate data untuk tabel quotes secara otomatis. Berikut ini langkah langkah yang diperlukan untuk membuat sebuah factory.

1. Buka file database/factories/QuoteFactory.php
2. Perhatikan isi awal dari file Quote Factory

```
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

class QuoteFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            //
        ];
    }
}
```

3. Kita akan menambahkan beberapa baris pada function definition() agar dapat menambah input sesuai dengan yang kita inginkan.

```
<?php

namespace Database\Factories;
```

```

use Illuminate\Database\Eloquent\Factories\Factory;

class QuoteFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            //
            'text' => $this->faker->sentence,
            'author' => $this->faker->name,
        ];
    }
}

```

4. Pada function definition() kita sudah menambahkan array ‘text’ akan berisi faker yang berupa kalimat, dan pada array author akan berisi data faker yang berupa nama.

Mengupdate Quote Seeder.

Agar tabel quote dapat diisi oleh data dari faker ini kita harus mengupdate file quote seeder agar menggunakan factory yang sudah kita edit. Berikut ini langkah-langkah yang diperlukan untuk mengupdate quote seeder.

1. Bukalah file database/seeders/QuoteSeeder.php
2. Berikut ini tampilan awal dari file QuoteSeeder.php

```

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class QuoteSeeder extends Seeder
{

```

```
/**  
 * Run the database seeds.  
 *  
 * @return void  
 */  
public function run()  
{  
    //  
}  
}
```

3. Kita perlu memanggil Quote factory di function run().

```
<?php  
  
namespace Database\Seeders;  
  
use App\Models\Quote;  
use Illuminate\Database\Seeder;  
  
class QuoteSeeder extends Seeder  
{  
    /**  
     * Run the database seeds.  
     *  
     * @return void  
     */  
    public function run()  
    {  
        //  
        Quote::factory()->count(10)->create();  
    }  
}
```

4. Dengan menambahkan pemanggilan factory pada function run ini kita akan mentrigger factory ketika dilakukan migration yang membutuhkan seeder.

Memanggil QuoteSeeder di Database Seeder.

Langkah terakhir dalam proses mempersiapkan database dan factory nya ini adalah memanggil quote seeder di file database seeder.

1. Buka file DatabaseSeeder pada folder database/seeders/DatabaseSeeder.php
2. Berikut ini isi default dari file DatabaseSeeder.

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // \App\Models\User::factory(10)->create();
    }
}
```

3. Untuk memanggil Quote Seeder kita perlu melakukan modifikasi pada function run.

```
<?php

namespace Database\Seeders;

use App\Models\Quote;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
```

```

*
* @return void
*/
public function run()
{
    \App\Models\User::factory(10)->create();
    $this->call([
        QuoteSeeder::class
    ]);
}

```

4. Pada function run kita mengenable generate 10 user dan memanggil class Quote Seeder.
5. Untuk melakukan validasi apakah kode program yang kita buat sudah berjalan dengan benar maka lakukanlah perintah berikut untuk mereset database dan mengisi kembali data ke dalam tabel.

```
php artisan migrate:fresh --seed
```

```

Dropped all tables successfully.

Migration table created successfully.

Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (95.91ms)

Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (72.91ms)

Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (54.69ms)

Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table
(52.61ms)

Migrating: 2022_01_13_095748_create_quotes_table
Migrated: 2022_01_13_095748_create_quotes_table (12.23ms)

Seeding: Database\Seeders\QuoteSeeder
Seeded: Database\Seeders\QuoteSeeder (47.18ms)

Database seeding completed successfully.

```

Berikut ini isi dari tabel Quote jika proses modifikasi seeder dan factory berhasil (perhatikan data yang digenerate bisa berbeda dari data yang ada pada gambar ini).

The screenshot shows the MySQL Workbench interface with the database 'rest\_api' selected. In the left sidebar, under the 'Tables' section, the 'quotes' table is highlighted. The main pane displays a table with the following columns: id, text, author, created\_at, and updated\_at. There are 10 rows of data, each representing a quote with its author and timestamp.

	id	text	author	created_at	updated_at
1	1	Distinctio illum velit temporibus excepturi.	Dr. Josue Kerluke	2022-02-02 23:09:52	2022-02-02 23:09:52
2	2	Repellendus amet autem nihil amet et quas aut.	Mariile DuBuque	2022-02-02 23:09:52	2022-02-02 23:09:52
3	3	Quidem sint veniam qui et doloremque.	Maryam Cassin	2022-02-02 23:09:52	2022-02-02 23:09:52
4	4	Optio id dolor placeat voluptatum.	Kaley Schmeler	2022-02-02 23:09:52	2022-02-02 23:09:52
5	5	Quia omnis quam et doloremque non explicabo.	Ms. Shyann Heg..	2022-02-02 23:09:52	2022-02-02 23:09:52
6	6	Dignissimos quae porro nisi.	Gina Fadel PhD	2022-02-02 23:09:52	2022-02-02 23:09:52
7	7	Tempora exercitationem accusamus sint.	Sabryna Rau	2022-02-02 23:09:52	2022-02-02 23:09:52
8	8	Ipsa dolor dolorem neque similique.	Miss Naomie Con..	2022-02-02 23:09:52	2022-02-02 23:09:52
9	9	Suscipit sit eligendi velit dolorem nihil doloremque a...	Angelica Hayes	2022-02-02 23:09:52	2022-02-02 23:09:52
10	10	Minus sint deleniti atque consectetur eveniet.	Prof. Malvina Bei..	2022-02-02 23:09:52	2022-02-02 23:09:52

Pelajarilah langkah langkah pembuatan migration seeder dan factory ini dengan seksama sehingga kedepannya akan memudahkan anda dalam membuat aplikasi laravel yang dapat memiliki data awal secara otomatis. Hal ini akan sangat membantu dalam proses development.

### Membuat Route di Api.php

Langkah selanjutnya adalah membuat route yang akan diakses pada saat user menggunakan REST API yang kita buat. Untuk membuat route ini kita akan menggunakan apiResource dimana route yang menggunakan apiResource akan membuatkan route yang dibutuhkan untuk REST API. Buka file api.php kemudian tambahkan route berikut ini:

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\BmiController;
use App\Http\Controllers\Api\DemoController;
use App\Http\Controllers\Api\QuoteController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
```

```

/*
| -----
| API Routes
| -----
|
| Here is where you can register API routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

```

```

Route::get('/demo', [DemoController::class, 'index']);
Route::post('/demo', [DemoController::class, 'store']);
Route::put('/demo', [DemoController::class, 'update']);
Route::delete('/demo', [DemoController::class, 'delete']);

Route::post('/bmi', [BmiController::class, 'index']);
Route::apiResource('quote', QuoteController::class);

```

Dengan menambahkan route apiResource diatas kita mendapatkan route baru yang dapat digunakan untuk proses REST Api terhadap resource quote. Untuk melakukan pengecekan jalankan perintah berikut ini :

```
php artisan route:list --path=quote
```

Perintah ini akan menampilkan semua route yang memiliki path quote.

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	api/quote	quote.index	App\Http\Controllers\Api\QuoteController@index	api
	POST	api/quote	quote.store	App\Http\Controllers\Api\QuoteController@store	api
	GET HEAD	api/quote/{quote}	quote.show	App\Http\Controllers\Api\QuoteController@show	api
	PUT PATCH	api/quote/{quote}	quote.update	App\Http\Controllers\Api\QuoteController@update	api
	DELETE	api/quote/{quote}	quote.destroy	App\Http\Controllers\Api\QuoteController@destroy	api

## Membuat Controller

Sesuai dengan route yang kita buat bahwa route ini akan membutuhkan sebuah controller dengan nama QuoteController maka untuk membuat controller nya jalankan perintah berikut di terminal.

```
php artisan make:controller Api/QuoteController --resource --api --model=Quote
```

Perintah artisan diatas akan mengeneratekan sebuah file dengan nama QuoteController pada folder Api yang memiliki resource dan menggunakan model Quote. Berikut ini hasil file yang di generate.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Quote;
use Illuminate\Http\Request;

class QuoteController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
    }
```

```
        */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     *
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function show(Quote $quote)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Quote $quote)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function destroy(Quote $quote)
```

```
{  
    //  
}  
}
```

Perlu diperhatikan pada file controller yang digenerate ini akan berbeda dengan controller biasa dimana pada controller ini terdapat method yang dibutuhkan untuk meghandle suatu resource tapi hanya untuk proses REST Api. Function yang dibuatkan adalah index,store,show,update dan delete tanpa function create dan edit.

Pada file ini juga sudah menggunakan Quote sebagai route model binding nya sehingga tidak perlu lagi menggunakan binding manual pada routing. Selain itu laravel juga menambahkan parameter Request pada routingnya. Hal ini sangat baik karena mempermudah kita dalam membuat sebuah controller.

#### Membuat Endpoint GET untuk menampilkan list quote

Untuk endpoint get yang menampilkan list dari sebuah resource pada umumnya dilakukan di method index. Dimana method index ini akan di akses melalui endpoint /api/quote yang response nya adalah list dari resource quote baik dibentuk dalam sebuah pagination atau tidak. Pada mini project ini endpoint /api/quote akan kita buat untuk mengembalikan semua data dari tabel quote. Berikut ini langkah langkah yang perlu dilakukan untuk membuat REST API pada endpoint tersebut.

1. Buka file QuoteController.php berikut ini adalah isi dari function index di awal sebelum diedit.

```
class QuoteController extends Controller  
{  
    /**  
     * Display a listing of the resource.  
     *  
     * @return \Illuminate\Http\Response  
     */  
    public function index()  
    {  
        //
```

```

        return Quote::all();
    }
}

```

2. Pada function index dapat dilihat kode program untuk mengembalikan semua data dari tabel Quote.

Setelah membuat function index ini ujilah endpoint ini dengan menggunakan postman.

Berikut ini konfigurasi dari postman untuk mengakses endpoint /api/quote/

KEY	VALUE
Authorization	Bearer 1jjEujpypnOqHRSE0pttV3lVNq7tGBUJ442ZcJot8a
Cache-Control	no-cache
Postman-Token	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.0
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Accept	application/json
Key	Value

```

[
  {
    "id": 1,
    "text": "Distinctio illum velit temporibus excepturi.",
    "author": "Dr. Josue Kerluke",
    "created_at": "2022-02-02T23:09:52.000000Z",
    "updated_at": "2022-02-02T23:09:52.000000Z"
  },
  {
    "id": 2,
    "text": "Repellendus amet autem nihil amet et quas aut.",
    "author": "Marilie DuBuque",
    "created_at": "2022-02-02T23:09:52.000000Z",
    "updated_at": "2022-02-02T23:09:52.000000Z"
  }
]

```

```
    },  
]
```

Perhatikan dengan kode program di atas kita berhasil mengembalikan response REST API sesuai dengan data yang ada di database. Namun demikian ada baiknya kita tidak menampilkan semua kolom ke hasil REST API. Karena ada kolom kolom yang bisa jadi sifatnya rahasia sehingga tidak perlu kita tampilkan seperti created\_at, updated\_at dan kolom kolom lain yang perlu di sembunyikan atau di konversi. Untuk membuatnya dapat digunakan fitur Resource pada Laravel.

#### Membuat Resource Quote Untuk Merapikan Response

Untuk membuat resource quote dapat dilakukan dengan menggunakan perintah artisan di terminal. Ketikkanlah perintah ini pada terminal :

```
php artisan make:resource QuoteResource
```

Perintah ini akan membuatkan sebuah file baru dengan nama QuoteResource pada folder app\Http\Resource\QuoteResource.

```
├── Controllers  
│   └── Api  
│       └── Controller.php  
└── Kernel.php  
└── Middleware  
└── Requests  
└── Resources  
    └── QuoteResource.php
```

Berikut ini isi dari file QuoteResource.php pada saat awal di generate :

```
<?php  
  
namespace App\Http\Resources;
```

```
use Illuminate\Http\Resources\Json\JsonResource;

class QuoteResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable
     */
    public function toArray($request)
    {
        return [
    ];
}
```

Pada QuoteResource ini kita dapat memilih akan menampilkan resource yang mana dan dapat melakukan konversi terhadap response dari REST API nya. Berikut ini contoh dimana QuoteResource akan menampilkan id, text, dan author tapi tidak menampilkan created\_at dan updated\_at.

```
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class QuoteResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
```

```

    *
    @return
array|\Illuminate\Contracts\Support\Arrayable\JsonSerializable
    */
public function toArray($request)
{
    return [
        'id' => $this->id,
        'quote' => $this->text,
        'author' => $this->author,
    ];
}
}

```

Perhatikan pada return array dari function toArray yang di isikan kita dapat melakukan modifikasi key array nya dan modifikasi content dari key yang dikembalikan. Setelah membuat QuotereResource ini kita akan melanjutkan dengan menggunakan quote resource pada kode program di controller.

Bukalah file QuoteController.php kemudian modifikasi function index agar menjadi seperti berikut ini.

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;
use Illuminate\Http\Request;

class QuoteController extends Controller
{
    /**
     * Display a listing of the resource.

```

```

*
* @return \Illuminate\Http\Response
*/
public function index()
{
    //
    return QuoteResource::collection(Quote::all());
}

```

Pada modifikasi ini kita menggunakan QuoteResource sebagai return value dari REST API dengan menggunakan QuoteResource sekarang response dari rest api yang dibuat berubah menjadi sebagai berikut:

```
{
  "data": [
    {
      "id": 1,
      "quote": "Distinctio illum velit temporibus excepturi.",
      "author": "Dr. Josue Kerluke"
    },
    {
      "id": 2,
      "quote": "Repellendus amet autem nihil amet et quas aut.",
      "author": "Marilie DuBuque"
    },
    {
      "id": 3,
      "quote": "Quidem sint veniam qui et doloremque.",
      "author": "Maryam Cassin"
    },
    {
      "id": 4,
      "quote": "Optio id dolor placeat voluptatum.",
      "author": "Kaley Schmeler"
    }
  ]
}
```

```
        "id": 5,
        "quote": "Quia omnis quam et doloremque non explicabo.",
        "author": "Ms. Shyann Hegmann"
    },
]
}
```

Jika anda perhatikan response yang dikembalikan oleh sistem saat ini terdapat sebuah wrapper yang bernama `data[]`. Ada beberapa developer yang berargumen bahwa penambahan wrapper ini tidak diperlukan pada pembuatan REST API. Untuk memfasilitasi hal ini kita dapat menggunakan fitur laravel yaitu `JsonResource`. `JsonResource` ini dapat di setting di file `AppServiceProvider.php`. Bukalah file tersebut dan ubah pada function `boot` dengan menambahkan kode program berikut ini.

```
<?php

namespace App\Providers;

use Illuminate\Http\Resources\Json\JsonResource;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    ...
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        //
        JsonResource::withoutWrapping();
    }
}
```

Jika menggunakan `JsonResource::withoutWrapping();` kita akan mendapatkan response json yang tidak di wrapping dengan tag `data[]`. Berikut ini hasil dari response `/api/quote` jika tidak menggunakan wrapping.

```
[  
 {  
   "id": 1,  
   "quote": "Distinctio illum velit temporibus excepturi.",  
   "author": "Dr. Josue Kerluke"  
 },  
 {  
   "id": 2,  
   "quote": "Repellendus amet autem nihil amet et quas aut.",  
   "author": "Marilie DuBuque"  
 },  
 {  
   "id": 3,  
   "quote": "Quidem sint veniam qui et doloremque.",  
   "author": "Maryam Cassin"  
 },  
 {  
   "id": 4,  
   "quote": "Optio id dolor placeat voluptatum.",  
   "author": "Kaley Schmeler"  
 },  
 {  
   "id": 5,  
   "quote": "Quia omnis quam et doloremque non explicabo.",  
   "author": "Ms. Shyann Hegmann"  
 }  
 ]
```

Membuat Endpoint POST untuk membuat quote baru di database

Endpoint post pada url `/api/quote` biasanya digunakan untuk membuat sebuah resource baru, dalam hal ini adalah sebuah data baru di tabel quote. Endpoint ini menerima parameter

berupa text dan author dari Request dan akan menyimpan data tersebut ke database. Proses menerima, memvalidasi dan menyimpan data ini akan dilakukan pada function store di file QuoteController. Endpoint ini akan mengembalikan response json berupa data dari quote yang berhasil dibuat. Berikut ini langkah langkah yang dilakukan untuk membuat function store pada QuoteController.

1. Buka file QuoteController.php.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;
use Illuminate\Http\Request;

class QuoteController extends Controller
{

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        //
    }
}
```

2. Pada kali ini kita akan menggunakan Class Request dari Laravel untuk menerima input dari user dan melakukan validasi data sebelum menyimpan data ke database.

3. Contoh format request postman untuk endpoint POST pada /api/quote adalah sebagai berikut (Perhatikan pada bagian body digunakan x-www-form-urlencoded).

Buku Laravel Rest Api / Quote / Create Quote

POST localhost:8000/api/quote

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	Halo soko
<input checked="" type="checkbox"/> author	soko
Key	Value

4. Tambahkanlah proses validasi pada function store() dengan mengedit function store menjadi seperti kode program dibawah ini.

```
public function store(Request $request)
{
    //
    $validated = $request->validate([
        'text' => 'required|unique:quotes',
        'author' => 'required'
    ]);
    return new QuoteResource(Quote::create($validated));
}
```

5. Perhatikan pada proses validasi kita memvalidasi bahwa 'text' harus di isi dan unique pada tabel quotes, serta author juga harus di isi dan tidak harus unique.
6. Untuk mencoba validasi lakukan setup berikut pada postman.

POST  localhost:8000/api/quote

Params Authorization  Headers (11)  Body  Pre-request Script Tests Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	
<input checked="" type="checkbox"/> author	soko
Key	Value

7. Perhatikan pada input text value nya dikosongkan.

```
{
  "message": "The given data was invalid.",
  "errors": {
    "text": [
      "The text field is required."
    ]
  }
}
```

8. Untuk pengujian selanjutnya kosongkan field author.

POST  localhost:8000/api/quote

Params Authorization  Headers (11)  Body  Pre-request Script Tests Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	isi quote
<input checked="" type="checkbox"/> author	
Key	Value

9. Perhatikan error validasinya sekarang berubah

```
{
  "message": "The given data was invalid.",
  "errors": {
    "author": [
      "The author field is required."
    ]
  }
}
```

```
        ]
    }
}
```

10. Khusus untuk data text yang harus unique cobalah memasukkan data yang sama dua kali dan perhatikan error yang muncul.

Buku Laravel Rest Api / Quote / Create Quote

POST localhost:8000/api/quote

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	Halo soko
<input checked="" type="checkbox"/> author	soko
Key	Value

11. Perhatikan response dari input dengan text quote yang sudah pernah ada.

```
{
    "message": "The given data was invalid.",
    "errors": {
        "text": [
            "The text has already been taken."
        ]
    }
}
```

12. Perhatikan pada function store juga ada perintah untuk meresponse kan data dari quote yang dibuat dalam format quote resource. Perhatikan cara penulisan response resource untuk mengembalikan satu resource berbeda dengan cara mengembalikan collection pada method index().

```
public function store(Request $request)
{
    //
    $validated = $request->validate([
        'text' => 'required|unique:quotes',
        'author' => 'required'
    ])
```

```
    ]);

    return new QuoteResource(Quote::create($validated));
}
```

13. Pada proses create juga digunakan variabel \$validated dimana variabel \$validated ini berisi request yang sukses divalidasi.

#### Menambahkan Form Request Pada Function Store

Sejauh ini function store yang kita buat sudah berhasil menyimpan data ke database dengan benar namun function store ini dapat diperbaiki lagi dengan memindahkan proses validasi ke class request yang berdiri sendiri. Laravel sudah mendukung proses ini dengan menyediakan fitur yang bernama FormRequest. Untuk membuat form request pada function store ikutilah langkah langkah berikut ini.

1. Jalankan perintah berikut ini di terminal

```
php artisan make:request StoreQuoteRequest
```

2. Perintah tersebut akan membuat sebuah class baru bernama StoreQuoteRequest yang berfungsi sebagai pengganti dari class Request biasa yang digunakan pada langkah sebelumnya.
3. StoreQuoteRequest ini dapat digunakan untuk menambahkan validation rule sehingga menyederhanakan isi dari function store().
4. Berikut ini isi file StoreQuoteRequest pada saat awal dibuat.

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreQuoteRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
}
```

```

public function authorize()
{
    return false;
}

/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        //
    ];
}

```

5. Kita perlu memodifikasi file ini dengan mengubah beberapa variabel pada function authorize dan menambahkan rule di dalam function rules()
6. Perhatikan perubahan yang dilakukan pada kode program dibawah ini.

```

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreQuoteRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {

```

```

        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            //
            'text' => 'required|unique:quotes',
            'author' => 'required'
        ];
    }
}

```

7. Perubahan yang dilakukan pada file ini adalah mengubah return true pada function authorize yang artinya kita tidak melakukan pengecekan authorisasi pada validasi form ini.
8. Selanjutnya dilakukan dengan menambahkan rule required dan unique pada kolom text untuk tabel quotes.
9. Tidak lupa menambahkan rule required pada kolom author.
10. Selanjutnya gantilah kode program pada QuoteController di function store agar tidak lagi menggunakan class Request bawaan dari laravel melainkan menggunakan class StoreQuoteRequest.

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;

```

```

use Illuminate\Http\Request;

class QuoteController extends Controller
{
    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(StoreQuoteRequest $request)
    {
        return new QuoteResource(Quote::create($request->validated()));
    }
}

```

11. Selanjutnya dapat kembali dilakukan testing via postman untuk data yang valid.

Buku Laravel Rest Api / Quote / Create Quote

POST    localhost:8000/api/quote

Params    Authorization    Headers (11)    **Body**    Pre-request Script    Tests    Settings

none     form-data     x-www-form-urlencoded     raw     binary     GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	Quote Baru
<input checked="" type="checkbox"/> author	soko
Key	Value

```
{
    "id": 13,
    "quote": "Quote Baru",
    "author": "soko"
}
```

12. Coba lah kembali untuk data yang sama agar dapat menguji proses validasi unique pada kolom quote.

POST

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	Quote Baru
<input checked="" type="checkbox"/> author	soko
Key	Value

```
{
  "message": "The given data was invalid.",
  "errors": {
    "text": [
      "The text has already been taken."
    ]
  }
}
```

### 13. Cobalah kosongkan author untuk mengecek validasi kolom author.

POST

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	Quote Baru Sekali
<input checked="" type="checkbox"/> author	
Key	Value

```
{
  "message": "The given data was invalid.",
  "errors": {
    "author": [
      "The author field is required."
    ]
  }
}
```

#### 14. Demikian juga lakukan pengujian dengan data yang kosong.



The screenshot shows a Postman interface with a POST request to `localhost:8000/api/quote`. The `Body` tab is selected, showing two fields: `text` and `author`, both of which have checkboxes checked. The response body is a JSON object:

```
{  
    "message": "The given data was invalid.",  
    "errors": {  
        "text": [  
            "The text field is required."  
        ],  
        "author": [  
            "The author field is required."  
        ]  
    }  
}
```

Sekarang dengan menggunakan custom form request `StoreQuoteRequest` kita sudah berhasil memisahkan proses validasi dan proses penyimpanan data. Hal ini baik karena mengikuti prinsip single responsibility.

Membuat Endpoint GET untuk menampilkan satu Quote.

Untuk endpoint get yang menampilkan detail satu resource pada umumnya dilakukan di method `show`. Dimana method `show` ini akan di akses melalui endpoint `/api/quote/{idquote}` yang response nya adalah detail dari resource quote yang memiliki id `{idquote}`. Pada mini project ini endpoint `/api/quote/{idquote}` akan kita buat untuk mengembalikan detail quote dengan id sesuai parameter yang dikirim. Berikut ini langkah langkah yang perlu dilakukan untuk membuat REST API pada endpoint tersebut.

1. Buka file `QuoteController.php` perhatikan kode program saat ini setelah diubah adalah sebagai berikut:

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;

class QuoteController extends Controller
{
    /**
     * Display the specified resource.
     *
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function show(Quote $quote)
    {
        //
    }
}

```

2. Perhatikan function show pada controller ini menerima input berupa instance dari kelas Quote. Dalam hal ini laravel menggunakan fitur yang di sebut Route model binding. Dengan menggunakan route model binding laravel akan secara otomatis melakukan pencarian data terhadap tabel quote sesuai dengan id yang diberikan oleh user. Route model binding ini akan menghandle secara otomatis jika user mengirimkan id yang tidak ada di database. Dengan fitur ini kita tidak usah melakukan pengecekan secara manual.
3. Selanjutnya adalah mengembalikan data quote berdasarkan id dengan memanfaatkan quote resource yang sebelumnya sudah pernah dibuat. Perhatikan dengan seksama bagaimana cara menggunakan QuoteResource untuk menampilkan satu Quote.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;

class QuoteController extends Controller
{
    /**
     * Display the specified resource.
     *
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function show(Quote $quote)
    {
        //
        return new QuoteResource($quote);
    }
}
```

4. Pada kode program diatas sudah terlihat bahwa function show dengan menggunakan route model binding akan lebih clean dan lebih mudah dipahami.
5. Selanjutnya lakukan pengujian terhadap route tersebut dengan menggunakan postman. Berikut ini konfigurasi postman dan response nya.

KEY	VALUE
<input checked="" type="checkbox"/> Authorization	(i) Bearer 1jjEujpypnOqHRSE0pttV3IVNq7tGBUJ442ZcJot8a
<input checked="" type="checkbox"/> Cache-Control	(i) no-cache
<input checked="" type="checkbox"/> Postman-Token	(i) <calculated when request is sent>
<input checked="" type="checkbox"/> Host	(i) <calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent	(i) PostmanRuntime/7.29.0
<input type="checkbox"/> Accept	(i) */*
<input checked="" type="checkbox"/> Accept-Encoding	(i) gzip, deflate, br
<input checked="" type="checkbox"/> Connection	(i) keep-alive
<input checked="" type="checkbox"/> Accept	application/json

Key Value

```
{
  "id": 1,
  "quote": "Distinctio illum velit temporibus excepturi.",
  "author": "Dr. Josue Kerluke"
}
```

## Membuat Endpoint PUT untuk mengedit Quote

Untuk endpoint dengan verb PUT biasanya digunakan untuk mengedit data dari suatu resource biasanya method yang digunakan untuk menghandle request PUT ini adalah method update. Dimana method update ini akan di akses melalui endpoint /api/quote/{idquote} dengan method PUT yang response nya adalah detail dari resource quote yang di edit. Pada mini project ini endpoint /api/quote/{idquote} akan kita buat untuk mengembalikan detail quote yang sudah di edit. Berikut ini langkah langkah yang perlu dilakukan untuk membuat REST API pada endpoint tersebut.

1. Buka file QuoteController Perhatikan isi dari method update sebelum diubah.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
```

```

use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;
use Illuminate\Http\Request;

class QuoteController extends Controller
{
    /**
     * Update the specified resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Quote $quote)
    {
        //
    }
}

```

2. Perhatikan bahwa pada function update masih menggunakan class Request bawaan dari laravel.
3. Berdasarkan langkah yang sudah kita pelajari pada proses membuat endpoint POST untuk create Quote kita akan mengganti class Request ini menjadi custom Request yang khusus untuk mengedit data. Untuk membuat Request ini dapat dilakukan dengan menjalankan perintah berikut ini pada terminal.

```
php artisan make:request UpdateQuoteRequest
```

4. Perintah diatas akan membuatkan sebuah file dengan nama UpdateQuoteRequest pada folder app/http/request/UpdateQuoteRequest.php.

```

├── Controllers
│   ├── Api
│   └── Controller.php
└── Kernel.php
├── Middleware
└── Requests
    └── StoreQuoteRequest.php

```

```
|   └── UpdateQuoteRequest.php  
└── Resources  
    └── QuoteResource.php
```

5. Berikut ini isi awal dari UpdateQuoteRequest.php

```
<?php  
  
namespace App\Http\Requests;  
  
use Illuminate\Foundation\Http\FormRequest;  
use Illuminate\Validation\Rule;  
  
class UpdateQuoteRequest extends FormRequest  
{  
    /**  
     * Determine if the user is authorized to make this request.  
     *  
     * @return bool  
     */  
    public function authorize()  
    {  
        return false;  
    }  
  
    /**  
     * Get the validation rules that apply to the request.  
     *  
     * @return array  
     */  
    public function rules()  
    {  
        return [  
            //  
        ];  
    }  
}
```

6. Modifikasilah file UpdateQuoteRequest pada function authorize dan rules.

```

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class UpdateQuoteRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            //
            'text' => 'required|'. Rule::unique('quotes')-
>ignore($this->quote),
            'author' => 'required'
        ];
    }
}

```

7. Ubahlah return dari function authorize menjadi true dan khusus pada bagian validasi kita harus menambahkan custom validation rule untuk kolom text.

8. Khusus kolom text akan ditambahkan validation rule berikut ini

```
'text' => 'required| ' . Rule::unique('quotes')->ignore($this->quote),
```

9. Khusus pada rule validasi ini kita menambahkan ignore untuk quote saat ini agar validasi unique nya tidak ter trigger saat kita memasukkan quote yang sama ketika mengedit data.
10. Selanjutnya kita akan menggunakan UpdateQuoteRequest ini pada controller QuoteController.php

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;
use Illuminate\Http\Request;

class QuoteController extends Controller
{

    /**
     * Update the specified resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function update(UpdateQuoteRequest $request, Quote $quote)
    {
        //
        $quote->update($request->validated());
        return new QuoteResource($quote);
    }
}
```

11. Selanjutnya kita dapat menguji endpoint ini pada postman dengan mengubah satu resource dan mengeceknya di endpoint get. Berikut ini response ketika kita mengakses endpoint localhost:8000/api/quote/1.

Buku Laravel Rest Api / Quote / Show Quote

GET    localhost:8000/api/quote/1

Params    Authorization (green)    Headers (9)    Body    Pre-request Script    Tests    Settings

KEY	VALUE
<input checked="" type="checkbox"/> Authorization	Bearer 1 jEujpypnOqHRSE0pttV3lVNq7tGBUJ442ZcJot8a
<input checked="" type="checkbox"/> Cache-Control	no-cache
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>
<input checked="" type="checkbox"/> Host	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.0
<input type="checkbox"/> Accept	*/*
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/> Connection	keep-alive
<input checked="" type="checkbox"/> Accept	application/json

{  
    "id": 1,  
    "quote": "Distinctio illum velit temporibus excepturi.",  
    "author": "Dr. Josue Kerluke"  
}

12. Kita akan mengubah data tersebut dengan mengakses endpoint localhost:8000/api/quote/1 dengan method PUT.

```
{  
    "id": 1,  
    "quote": "new quote",  
    "author": "new author"  
}
```

KEY	VALUE
<input checked="" type="checkbox"/> text	new quote
<input checked="" type="checkbox"/> author	new author
Key	Value

```
{
  "id": 1,
  "quote": "new quote",
  "author": "new author"
}
```

13. Jika diakses lagi di endpoint <http://localhost:8000/api/quote/1> data sudah berubah.

```
{
  "id": 1,
  "quote": "new quote",
  "author": "new author"
}
```

### Membuat Endpoint DELETE untuk menghapus Quote

Untuk endpoint dengan verb DELETE biasanya digunakan untuk menghapus data dari suatu resource biasanya method yang digunakan untuk menghandle request PUT ini adalah method destroy. Dimana method destroy ini akan di akses melalui endpoint /api/quote/{idquote} dengan method DELETE yang jika data sukses di hapus adalah no content. Pada mini project ini endpoint /api/quote/{idquote} akan kita buat untuk mengembalikan response no content jika berhasil menghapus resource dan not found jika tidak menemukan resource. Berikut ini langkah-langkah yang perlu dilakukan untuk membuat REST API pada endpoint tersebut.

1. Buka file QuoteController.php dan perhatikanlah method delete pada saat ini.

```
<?php
```

```

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;
use Illuminate\Http\Request;

class QuoteController extends Controller
{

    /**
     * Remove the specified resource from storage.
     *
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function destroy(Quote $quote)
    {
        //
    }
}

```

2. Pada method destroy ini kita sudah menggunakan route model binding pada route nya oleh karena itu paramter yang dikirim adalah instance dari Model Quote.
3. Selanjutnya untuk proses delete tambahkan kode program berikut untuk menghapus data dari database.

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreQuoteRequest;
use App\Http\Requests\UpdateQuoteRequest;
use App\Http\Resources\QuoteResource;
use App\Models\Quote;

```

```

use Illuminate\Http\Request;

class QuoteController extends Controller
{

    /**
     * Remove the specified resource from storage.
     *
     * @param \App\Models\Quote $quote
     * @return \Illuminate\Http\Response
     */
    public function destroy(Quote $quote)
    {
        $quote->delete();
        return response()->noContent();
    }
}

```

4. Pada implementasi dari function destroy ini kita perlu menambahkan proses hapus data dan mengembalikan response no content.
5. Selanjutnya ujilah endpoint tersebut menggunakan postman.

Buku Laravel Rest Api / Quote / **Destroy Quote**

The screenshot shows a Postman interface with the following details:

- Method:** DELETE
- URL:** rest-api.test/api/quote/1
- Headers:** (9) (highlighted)
- Params:** None
- Authorization:** Bearer 1|jEujppnOqHRSE0pttV3lVNq7tGBUJ442ZcJot8a
- Headers (Visible):**

KEY	VALUE
Authorization	Bearer 1 jEujppnOqHRSE0pttV3lVNq7tGBUJ442ZcJot8a
Cache-Control	no-cache
Postman-Token	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.0
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Accept	application/json
- Body:** None
- Pre-request Script:** None
- Tests:** None
- Settings:** None

6. Response dari function ini memang tidak ada karena response nya adalah no content.

The screenshot shows the Postman interface with the 'Body' tab selected. Below it, there are tabs for 'Cookies', 'Headers (8)', and 'Test Results'. Under the 'Headers' tab, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', 'Text', and a copy icon. A single item is listed below the buttons.

- Untuk memastikan cobalah membuka endpoint <http://localhost:8000/api/quote/1> untuk mengecek data yang didelete.

The screenshot shows the Postman interface with a 'GET' request to 'localhost:8000/api/quote/1'. The 'Headers' tab is selected, showing the following configuration:

Key	Value
Authorization	Bearer 1 jEujpypnOqHRSE0pttV3IVNq7tGBUJ442ZcJot8a
Cache-Control	no-cache
Postman-Token	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.0
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Accept	application/json

- Namun demikian ketika kita menghapus data yang id nya tidak ada di dalam database seperti <http://localhost:8000/api/quote/1000> maka error response nya akan menjadi seperti berikut ini:

```
{
    "message": "No query results for model [App\\Models\\Quote] 1",
    "exception": "Symfony\\Component\\HttpKernel\\Exception\\NotFoundHttpException",
    "file": "/Users/putraprima/Documents/Stuffs/laravel/rest-api/vendor/laravel/framework/src/Illuminate/Foundation/Exceptions/Handler.php",
```

```
    "line": 385,  
    ...  
}
```

9. Untuk menghandle permasalahan ini kita perlu memperbaiki tampilan error dengan mengubah file app/exeption/handler.php

```
<?php  
  
namespace App\Exceptions;  
  
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;  
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;  
use Throwable;  
  
class Handler extends ExceptionHandler  
{  
    /**  
     * A list of the exception types that are not reported.  
     *  
     * @var array<int, class-string<Throwable>>  
     */  
    protected $dontReport = [  
        //  
    ];  
  
    /**  
     * A list of the inputs that are never flashed for validation  
     * exceptions.  
     *  
     * @var array<int, string>  
     */  
    protected $dontFlash = [  
        'current_password',  
        'password',  
        'password_confirmation',
```

```

];
/** 
 * Register the exception handling callbacks for the application.
 *
 * @return void
 */
public function register()
{
    $this->renderable(function (NotFoundHttpException $e) {
        //
        return response()->json(['message' => 'Object Not
Found'], 404);
    });
}
}

```

10. Dengan melakukan modifikasi diatas data yang dikembalikan oleh rest api ketika ada exception not found adalah sebagai berikut :

```
{
    "message": "Object Not Found"
}
```

Dengan selesainya membuat endpoint dete ini kita telah berhasil membuat sebuah CRUD yang clean untuk REST API pada resource Quote.

### [Improve Mini Project REST API Quote App Dengan Admin Akses](#)

Untuk saat ini semua endpoint yang dibuat masih dapat diakses secara publik, untuk melindungi endpoint endpoint lain kita perlu membuat sebuah mekanisme authentikasi yang dapat digunakan untuk memproteksi endpoint spesifik agar tidak dapat diakses oleh sembarang user.

Misalkan pada kasus ini kita ingin membuat endpoint GET quote di /api/quote sebagai endpoint public dan sisanya untuk proses create edit dan delete quote di haruskan login

terlebih dahulu sebagai admin. Untuk memenuhi fitur tersebut kita harus menggunakan library tambahan untuk authentikasi library ini bernama laravel sanctum.

Laravel sanctum menyediakan beberapa skenario untuk authentikasi yaitu Api token authentication, Spa authentication dan Mobile App authentication. Pada buku ini kita akan menggunakan Mobile App Authentication.

#### Install dan Config Laravel Sanctum

Laravel sanctum dapat di install melalui composer dengan menjalankan perintah berikut pada terminal di project rest-api.

```
rest-api on □ master via ☈ v14.15.5 via 🐘 v7.4.27
> composer require laravel/sanctum
```

Laravel sanctum memiliki beberapa file configurasi yang harus di publish agar dapat dimodifikasi dan di setting sesuai kebutuhan project kita untuk mempublish nya gunakanlah perintah berikut di terminal.

```
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
```

File configurasi ini terdiri dari beberapa file salah satunya adalah file migration yang berisi proses pembuatan tabel tambahan yang digunakan oleh laravel sanctum untuk proses authentikasi oleh karena itu kita harus melakukan perintah migration sebelum mulai menggunakan Laravel Sanctum. Jalankan perintah ini di terminal untuk menjalankan migration.

```
php artisan migrate
```

Pada project ini kita menggunakan User model sebagai basis proses authentikasi yang kita gunakan oleh karena itu file models/user.php harus dimodifikasi agar menggunakan trait HasApiToken dari laravel sanctum. Berikut ini hasil modifikasi pada file user.php

```
<?php

namespace App\Models;
```

```

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
    ...
}

```

### Protect Endpoint Menggunakan Laravel Sanctum

Untuk memproteksi endpoint menggunakan laravel sanctum kita dapat menggunakan middleware pada file route api.php. Middleware sanctum ini baru dapat kita gunakan jika proses installasi sudah berhasil. Route di api.php akan kita sesuaikan dengan kebutuhan berikut ini :

Method	Access
show	Public
index	Public
update	Authenticated
destroy	Authenticated
store	Authenticated

Untuk memproteksi endpoint kita modifikasi file api.php agar mengikuti kode program dibawah ini.

```

Route::get('quote', [QuoteController::class, 'index'])->name('quote.index');
Route::get('quote/{Quote}', [QuoteController::class, 'show'])->name('quote.show');
Route::group(['middleware' => 'auth:sanctum'], function () {
    Route::resource('quote', QuoteController::class)->only(['store', 'update',
    'destroy']);
});

```

Pada kode routing diatas kita membuat sebuah route resource yang hanya menggunakan store, update dan destroy. Untuk route yang public kita membuat dua route dengan route model binding yang bersifat public.

### Test Protected dan Open Endpoint

Kita dapat menggunakan postman untuk mengetes apakah endpoint dengan route baru yang kita buat dapat bekerja dengan baik. Untuk pengujian buka lah kembali aplikasi postman dan cobalah melakukan proses store, update atau destroy pada resource quote.

Buku Laravel Rest Api / Quote / Create Quote

POST localhost:8000/api/quote

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> text	
<input checked="" type="checkbox"/> author	

{  
    "message": "Unauthenticated."  
}

Buku Laravel Rest Api / Quote / Destroy Quote

DELETE rest-api.test/api/quote/1

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers Hide auto-generated headers

KEY	VALUE
<input checked="" type="checkbox"/> Cache-Control	no-cache
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>
<input checked="" type="checkbox"/> Host	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.0
<input type="checkbox"/> Accept	*/*
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/> Connection	keep-alive
<input checked="" type="checkbox"/> Accept	application/json

{  
    "message": "Unauthenticated."  
}

Perhatikan response yang diberikan yaitu unauthenticated ini artinya kita sudah bisa memberikan proteksi terhadap route store,destroy dan update. Selanjutnya kita perlu membuatkan sebuah endpoint untuk proses login dan logout serta bagimana memanfaatkan token agar dapat melakukan proses login melalui rest api dan mengakses endpoint yang terproteksi.

#### Membuat Controller dan Method Register

Sementara ini kita masih menggunakan user yang di generate melalui file DatabaseSeeder.php namun untuk mempermudah proses registrasi user kita harus membuat sebuah controller untuk authentikasi agar dapat mendukung fitur register login dan logout melalui rest api. Berikut ini langkah langkah untuk membuat proses registrasi pada Rest Api dengan menggunakan Laravel sanctum

1. Buka terminal kemudian generate sebuah controller di folder api dengan nama AuthController dengan menggunakan command berikut ini.

```
rest-api on □ master [!] via ☎ v14.15.5 via 🐘 v7.4.27
> php artisan make:controller Api/AuthController
```

2. Pastikan file AuthController berhasil dibuat di dalam folder api.
3. Buka file routing api.php kemudian tambahkan route untuk register login dan logout dimana route logout berada di dalam middleware yang di cek menggunakan auth:sanctum.

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\BmiController;
use App\Http\Controllers\Api\DemoController;
use App\Http\Controllers\Api\QuoteController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

...
Route::post('/auth/login', [AuthController::class, 'login']);
Route::post('/auth/register', [AuthController::class, 'register']);
```

```

Route::get('quote', [QuoteController::class, 'index'])->name('quote.index');

Route::get('quote/{Quote}', [QuoteController::class, 'show'])->name('quote.show');

Route::group(['middleware' => 'auth:sanctum'], function () {
    Route::post('/auth/logout', [AuthController::class, 'logout']);
    Route::resource('quote', QuoteController::class)->only(['store', 'update', 'destroy']);
});

```

4. Pada bagian ini kita akan fokus pada function register di file auth controller. Modifikasi auth controller sehingga dapat memiliki function login register dan logout.

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Illuminate\Validation\ValidationException;

class AuthController extends Controller
{
    //

    public function login(Request $request)
    {

    }

    public function register(Request $request)
    {
    }
}

```

```
    public function logout(Request $request)
    {
    }
}
```

5. Selanjutnya kita akan fokus pada fungsi register, pada function ini kita akan menambahkan logika registrasi dan mengembalikan token yang dapat digunakan untuk proses selanjutnya.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Illuminate\Validation\ValidationException;

class AuthController extends Controller
{
    //

    public function register(Request $request)
    {
        $request->validate([
            'name' => 'required|string',
            'email' => 'required|string|email|unique:users',
            'password' => ['required', 'string', 'min:8'],
            'confirmed', Password::defaults(),
            'device_name' => 'required',
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
    }
}
```

```

    ]);
}

return $user->createToken($request->device_name)-
>plainTextToken;
}
}

```

6. Pada kode program diatas kita function register akan memvalidasi input nama,email dan password kemudian jika validasi berhasil menyimpan data user ke database dan mengembalikan token sebagai result dari response endpoint ini.
7. Selanjutnya ujilah proses registrasi ini menggunakan postman, berikut ini konfigurasinya.

Buku Laravel Rest Api / Auth / User Register ⚙ ⚙

**POST** rest-api.test/api/auth/register

Params	Authorization	Headers (10)	Body	Pre-request Script	Tests	Settings													
none			<input checked="" type="radio"/> form-data <input checked="" type="radio"/> x-www-form-urlencoded <input type="radio"/> raw <input type="radio"/> binary <input type="radio"/> GraphQL																
			<table border="1"> <thead> <tr> <th>KEY</th> <th>VALUE</th> </tr> </thead> <tbody> <tr> <td>email</td> <td>putraprima@gmail.com</td> </tr> <tr> <td>password</td> <td>password</td> </tr> <tr> <td>name</td> <td>prima</td> </tr> <tr> <td>password_confirmation</td> <td>password</td> </tr> <tr> <td>device_name</td> <td>android</td> </tr> <tr> <td>Key</td> <td>Value</td> </tr> </tbody> </table>	KEY	VALUE	email	putraprima@gmail.com	password	password	name	prima	password_confirmation	password	device_name	android	Key	Value		
KEY	VALUE																		
email	putraprima@gmail.com																		
password	password																		
name	prima																		
password_confirmation	password																		
device_name	android																		
Key	Value																		

8. Response dari rest api ini adalah sebuah token yang terenkripsi

Body Cookies Headers (12) Test Results

Pretty Raw Preview Visualize HTML

1 |mrJkUTbFeo2AUK7cMVNQdsbHENLp0Xh0atWlCa0v

## Membuat Controller dan Method Login

Untuk selanjutnya kita akan membuat method login dimana dengan melakukan login user akan mendapatkan token yang dapat digunakan untuk mengakses endpoint yang di lindungi oleh middleware auth:sanctum.

1. Ubahlah function login dengan kode program berikut ini.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Illuminate\Validation\ValidationException;

class AuthController extends Controller
{
    //
    public function login(Request $request)
    {
        $request->validate([
            'email' => 'required|email',
            'password' => 'required',
            'device_name' => 'required',
        ]);

        $user = User::where('email', $request->email)->first();

        if (! $user || ! Hash::check($request->password, $user->password)) {
            throw ValidationException::withMessages([
                'email' => ['The provided credentials are incorrect.'],
            ]);
        }
    }
}
```

```

        }

        return $user->createToken($request->device_name)-
>plainTextToken;
    }
}

```

2. Lakukanlah pengujian dengan menggunakan konfigurasi postman sebagai berikut

POST u Laravel Rest Api / Auth / User Login

POST rest-api.test/api/auth/login

Params Authorization Headers (10) Body **Pre-request Script Tests Settings**

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> email	putraprima@gmail.com
<input checked="" type="checkbox"/> password	password
<input checked="" type="checkbox"/> device_name	android
Key	Value

Body Cookies Headers (12) Test Results

Pretty Raw Preview Visualize HTML

1 2|stLaV8zzsY1jik1iGS1VtMkEeUCMSi33RQ8rIJER

3. Hasil dari request login adalah token. Selanjutnya mari kita uji coba apakah token ini berhasil digunakan ketika akan mengakses endpoint yang dilindungi oleh auth.

Buku Laravel Rest Api / Quote / Create Quote

POST localhost:8000/api/quote

Params Authorization Headers (11) Body **Pre-request Script Tests Settings**

Type Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization ↗](#)

Token

4. Buka tab Authorization pada request, kemudian pilih type Bearer Token dan tambahkan token yang tadi didapatkan pada request login.
5. Selanjutnya tambahkan body text dan author untuk melengkapi data quote.

Buku Laravel Rest Api / Quote / Create Quote

KEY	VALUE
<input checked="" type="checkbox"/> text	Quote Super Terbaru
<input checked="" type="checkbox"/> author	sokosama
Key	Value

```
{
  "id": 14,
  "quote": "Quote Super Terbaru",
  "author": "sokosama"
}
```

6. Untuk menguji apakah authentikasi berjalan dengan benar hapuslah bearer token pada tab authorization kemudian lakukan request yang sama sekali lagi.

Buku Laravel Rest Api / Quote / Create Quote

Type: Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token: Token

```
{
  "message": "Unauthenticated."
}
```

7. Perhatikan dengan request yang sama jika tidak memiliki token atau token nya salah maka response nya akan berbeda yang memiliki token yang benar akan memberikan

response yang berhasil sementara yang tidak memiliki token akan memberikan response unauthenticated.

### Membuat Controller dan Method Logout

Bagian akhir dari proses pembuatan project Quote REST API ini adalah membuatkan function logout. Dimana pada function ini kita akan menghapus token dari database atas user yang sedang login. Ikutilah langkah berikut ini dalam memodifikasi file AuthController.

1. Bukalah file AuthController.php dan tambahkan kode program berikut pada function logout.

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Illuminate\Validation\ValidationException;

class AuthController extends Controller
{
    ...
    public function logout(Request $request)
    {
        $request->user()->currentAccessToken()->delete();
        return response()->noContent();
    }
    ...
}
```

2. Pada function logout kita akan menghapus currentAccessToken dan mengembalikan response no content.
3. Untuk menguji logout gunakanlah token yang didapat dari endpoint login.

4. Gunakan konfigurasi postman berikut ini untuk menguji proses logout.

Buku Laravel Rest Api / Auth / User Logout

POST rest-api.test/api/auth/logout

Params Authorization **Headers (10)** Body Pre-request Script Tests Settings

Type Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization ↗](#)

Token

2|stLaV8zzsY1jik1GS1VtMkEeUCMSi33RQE ...

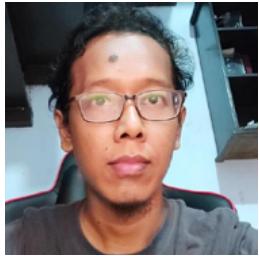
Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text

1 |

5. Pada proses logout kita tidak mengirimkan parameter apapun, cukup dengan menambahkan bearer token dari proses login. Ketika user sudah di logout bearer token yang dikirim pada proses logout tidak dapat digunakan lagi.

## Biodata Penulis

	<p><b>Putra Prima Arhandi, S.T., M.Kom</b> lahir di Pekanbaru. Penulis menyelesaikan S1 pada Fakultas Teknik di Universitas Brawijaya dan S2 di Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya. Penulis merupakan pengajar mata kuliah Pemrograman Mobile, Pemrograman Web dan Pengujian Perangkat Lunak di Jurusan Teknologi Informasi, Politeknik Negeri Malang, sejak tahun 2015 sampai sekarang. Aktif membuat konten pembelajaran di youtube melalui channel DosenNgoding dan aktif kontribusi ke open source project di github @siubie, Semangat Ngoding !!!.</p>
	<p><b>Dian Hanifudin Subhi, S.Kom., M.Kom.,</b> lahir di Gresik. Penulis menyelesaikan S1 dan S2 di Teknik Informatika Institut Teknologi Sepuluh Nopember, Surabaya. Penulis merupakan pengajar mata kuliah Pemrograman Mobile dan Pemrograman Web Lanjut di Jurusan Teknologi Informasi Politeknik Negeri Malang.</p>
	<p><b>Arie Rachmad Syulistyo, S.Kom., M.Kom.,</b> lahir di Malang. Penulis menyelesaikan S1 pada Fakultas MIPA di Universitas Brawijaya dan S2 di Fakultas Ilmu Komputer, Universitas Indonesia di Depok. Penulis merupakan pengajar mata kuliah Pemrograman Mobile, Pengolahan Citra Digital, dan Pengembangan Perangkat Lunak Berbasis Objek di Jurusan Teknologi Informasi, Politeknik Negeri Malang, sejak tahun 2016. Aktif berkontribusi pada gerakan SMK Coding dan Jong Kreatif <a href="https://jongkreatif.id/">https://jongkreatif.id/</a>.</p>
	<p><b>Annisa Taufika Firdausi ST.,MT</b> lahir di Malang. Penulis menyelesaikan S1 dan S2 di Teknik Elektro Universitas Brawijaya Malang. Penulis merupakan pengajar di Jurusan Teknologi Informasi Politeknik Negeri Malang sejak 2013. Mata Kuliah yang pernah diampu antara lain Organisasi dan Arsitektur Komputer, Dasar Multimedia, Sistem Cerdas, Sistem Operasi, Aplikasi Komputer</p>

	Perkantoran, Ilmu Organisasi dan Komunikasi, Pengantar Teknologi Informasi, Sistem Informasi, Dasar Pemrograman, Desain dan Pemrograman Web, dan Basis Data.
--	--

## Kesimpulan

Pada buku ini telah di ajarkan Langkah Langkah membuat REST API menggunakan Laravel mulai dari teori sampai dengan implementasi. Untuk teori diajarkan bagaimana mendesain REST API yang baik dan benar, sedangkan untuk implementasi diajarkan bagaimana cara melakukan installasi perangkat lunak pendukung, bagaimana membuat REST API sederhana di Laravel, bagaimana membuat REST API untuk menghitung BMI dan yang terakhir bagaimana membuat REST API yang mampu menyimpan dan mengolah data serta memiliki proses authentikasi.

## Glosarium

REST API	REST adalah akronim dari <b>REpresentational State Transfer</b> and architectural style for distributed hypermedia systems. Yang dikemukakan oleh Roy Fielding pada tahun 2000 di desertasinya dengan judul <b>Representational State Transfer (REST)</b>
Backend	Server dari REST API yang menerima request dalam format JSON dan mengembalikan response dalam format JSON.
Frontend	Client dari REST API yang mengirim dan menerima request dan response serta merubah tampilan berdasarkan response yang diterima.
Controller	File yang mengatur logika bisnis di framework Laravel
Model	File yang mengatur entitas pada framework Laravel.
Form Request	Class yang mengatur validasi pada sebuah request
Endpoint	Url yang menjadi sumber dan response dari REST API.

## [Daftar Pustaka](#)

Griffin, J. (2021). *Domain-Driven Laravel: Learn to Implement Domain-Driven Design Using Laravel*. Apress. <https://doi.org/10.1007/978-1-4842-6023-4>

Massé, M. H., & Massé, M. (2012). *REST API design rulebook: Designing consistent RESTful Web Service Interfaces*. O'Reilly.

Vallejo, P. (n.d.). *Practical Laravel: Develop clean MVC web applications*. 118.

John Au-Yeung and Ryan Donovan <https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/>

<https://chocolatey.org/install>