

VERSI 1.0
DESEMBER 2025



[PEMROGRAMAN WEB]
MODUL 6 - MIDDLEWARE DAN FILE STORAGE

DISUSUN OLEH:

AMINUDIN, S.KOM., M.CS.

YUSUF NUR MUHAMMAD

KRISNA BIMANTORO

TIM LABORATORIUM INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG

PENDAHULUAN

TUJUAN

Memberikan pemahaman kepada mahasiswa mengenai pengamanan REST API menggunakan JWT dan pengelolaan file di Laravel, sehingga mampu:

1. Menginstal dan mengkonfigurasi *library* JWT Auth.
2. Mengimplementasikan fitur Register, Login, dan Logout pada API.
3. Menggunakan Middleware untuk membatasi akses endpoint.
4. Mengkonfigurasi File Storage (Symlink).
5. Membuat endpoint untuk upload file (gambar/dokumen).

TARGET MODUL

Setelah menyelesaikan modul ini mahasiswa diharapkan mampu:

1. Menjelaskan konsep *Stateless Authentication* dengan JWT.
2. Mengamankan endpoint API menggunakan Middleware.
3. Mengelola file upload dan menyimpannya ke dalam direktori publik.
4. Menguji autentikasi dan upload file menggunakan Postman.

PERSIAPAN

Sebelum mengikuti praktikum, mahasiswa perlu:

1. Memahami konsep dasar HTTP Header (Authorization).
2. Koneksi internet untuk mengunduh *library* via Composer.

KEYWORDS

JWT, Middleware, Authentication, Bearer Token, File Storage, Symbolic Link.

TABLE OF CONTENTS

PENDAHULUAN.....	2
TUJUAN.....	2
TARGET MODUL.....	2
PERSIAPAN.....	2
KEYWORDS.....	2
TABLE OF CONTENTS.....	2
MATERI.....	4
MIDDLEWARE.....	4
Fungsi Utama Middleware:.....	4
Ilustrasi Alur Kerja Middleware:.....	4





Jenis -jenis Middleware.....	4
AUTHENTICATION JWT (JSON WEB TOKEN).....	5
Alur Kerja JWT:.....	6
Instalasi JWT Auth di Laravel.....	6
Implementasi JWT Auth di Laravel.....	7
Pembuatan Postman untuk endpoint Auth.....	13
Solusinya dengan menambahkan bearer token pada Authorization.....	16
Implementasi middleware untuk endpoint todo.....	17
Informasi Opsional Postman.....	19
FILE STORAGE.....	21
Implementasi File Storage Laravel.....	21
Implementasi Pada Postman.....	24
CODELAB.....	26
TUGAS.....	27
KRITERIA & DETAIL PENILAIAN.....	28



MATERI

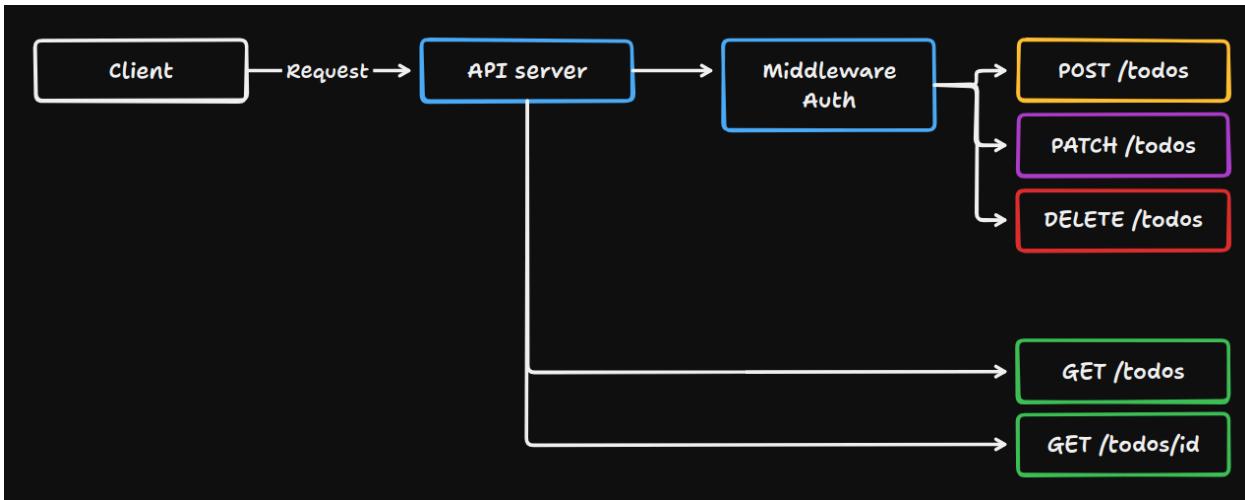
MIDDLEWARE

Middleware adalah salah satu konsep inti dalam framework Laravel. Middleware berfungsi sebagai lapisan perantara yang berada di antara Permintaan HTTP (Request) dari pengguna dan Aplikasi/Controller. Konsep dasar Middleware dapat diibaratkan sebagai (**Security Guard**) atau **Filter** yang bertugas memeriksa dan memproses setiap permintaan yang masuk sebelum diizinkan untuk melanjutkan ke *Controller* atau sebelum respons diproses untuk dikirimkan kembali ke klien.

Fungsi Utama Middleware:

- **Filtering:** Memblokir akses jika kondisi tertentu tidak terpenuhi (Misal: User belum login).
- **Processing:** Memodifikasi permintaan (Request) atau respons (Response). (Misal: Menambahkan header HTTP tertentu).

Ilustrasi Alur Kerja Middleware:



Contoh case, ketika user hit endpoint POST /todos maka memerlukan auth login dahulu agar bisa di proses pada controller, tetapi ketika user hit endpoint GET /todos maka user bisa mendapatkan responsenya ketika tidak login, karena endpoint tersebut tidak terdapat middleware

Jenis -jenis Middleware

1. Global Middleware: Dijalankan pada setiap HTTP request ke seluruh endpoint.
2. Route Middleware: Dijalankan pada route atau endpoint tertentu, contoh endpoint untuk manage resource pada admin, dan endpoint yang bisa digunakan pada user tertentu (Role-Based Access Control (RBAC))
3. Middleware Group: Sekelompok middleware yang diterapkan bersamaan pada sekelompok route.



AUTHENTICATION JWT (JSON WEB TOKEN)

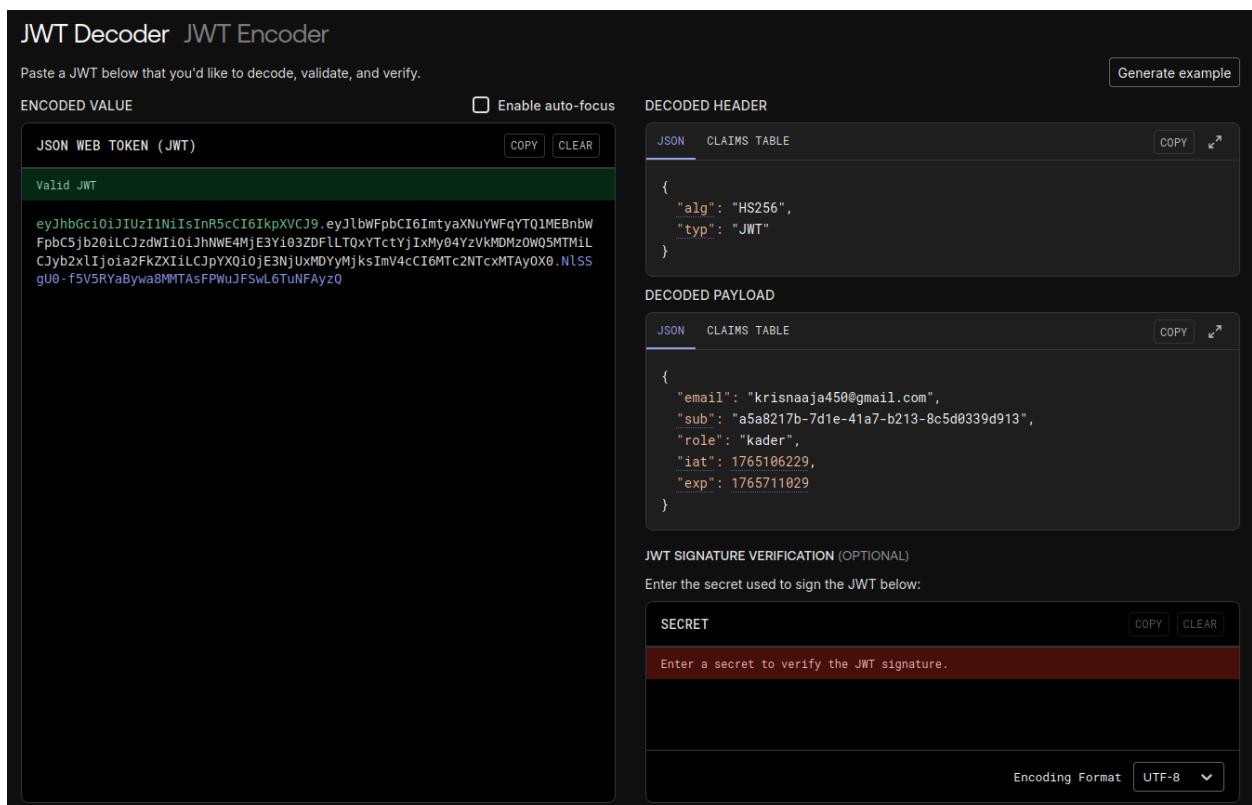
Pada aplikasi web tradisional, autentikasi biasanya menggunakan *session* dan *cookies*. Namun, untuk REST API yang bersifat *stateless* (tidak menyimpan status sesi di server), kita membutuhkan metode yang lebih fleksibel, yaitu **Token Based Authentication**.

Apa itu JWT? JWT adalah standar terbuka (RFC 7519) yang mendefinisikan cara ringkas dan mandiri untuk mengirimkan informasi antar pihak sebagai objek JSON. Informasi ini dapat diverifikasi dan dipercaya karena ditandatangani secara digital.

Struktur JWT terdiri dari tiga bagian yang dipisahkan oleh titik (.):

1. Header: Jenis token dan algoritma penandatanganan (misal: HS256).
2. Payload: Data klaim (informasi user, expiry time, dll).
3. Signature: Tanda tangan digital untuk memverifikasi bahwa token tidak dimanipulasi.

Contoh JWT yang di decode pada website JWT IO



The screenshot shows the JWT Decoder interface with a valid JWT token pasted into the 'Encoded Value' field. The token is:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImtyaXNuYWFrYTQ1MEBnbWFpbC5jb20iLCJwdIiOiJHNWE4MjE3Yi03ZDFLTQXYTctyjIxMy04Y2VNMMDz0WQ5MTMjLCJyb2xlijoia2FkZXIiLCJpYXQiOjE3NjUxMDYyMjksImV4cCI6MTc2NTcxMTAy0x0.NLSSgu0-f5V5RYaBywa8MMTAstFPWuJFswL6TuNFayzQ
```

The 'Decoded Header' section shows:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

The 'Decoded Payload' section shows:

```
{
  "email": "krisnaaja450@gmail.com",
  "sub": "a5ab217b-7d1e-41a7-b213-8c5d0339d913",
  "role": "kader",
  "iat": 1765106229,
  "exp": 176511029
}
```

The 'JWT SIGNATURE VERIFICATION (OPTIONAL)' section contains a 'SECRET' input field with placeholder text: 'Enter a secret to verify the JWT signature.' Below it is an 'Encoding Format' dropdown set to 'UTF-8'.



Pada bagian kiri adalah token JWT nya dan pada Decoded Payload adalah hasil dari decode token dengan payload pada gambar, jangan simpan credential seperti data password atau data penting lainnya pada payload JWT.

Alur Kerja JWT:

1. Client mengirim kredensial (email & password) ke Server (Login).
2. Server memverifikasi. Jika benar, Server membuat JWT dan mengirimkannya kembali ke Client.
3. Client menyimpan JWT (biasanya di LocalStorage atau Cookie).
4. Setiap kali Client ingin mengakses *protected route* (misal: menambah data Todo), Client mengirimkan JWT di header Authorization: Bearer <token>.
5. Server memvalidasi token. Jika valid, request diproses.

Instalasi JWT Auth di Laravel

Masih pada project codelab yang sama seperti modul 5

1. Pada project codelab kemarin jalankan perintah berikut pada terminal untuk install JWT nya

```
composer require php-open-source-saver/jwt-auth
```

```
+ pemrograman-web git:(main) ✘ composer require php-open-source-saver/jwt-auth
laravel/pail ..... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
pestphp/pest-plugin-laravel ..... DONE
php-open-source-saver/jwt-auth ..... DONE

90 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
[INFO] No publishable resources for tag [laravel-assets].
No security vulnerability advisories found.
Using version "2.8" for php-open-source-saver/jwt-auth
+ pemrograman-web git:(main) ✘
```

Contoh berhasil

2. Selanjutnya publish konfigurasi JWT dengan menjalankan perintah berikut dalam satu line

```
php artisan vendor:publish
--provider="PHPOpenSourceSaver\JWTAuth\Providers\LaravelServiceProvider"
```

```
Using version "2.8" for php-open-source-saver/jwt-auth
+ pemrograman-web git:(main) ✘ php artisan vendor:publish --provider="PHPOpenSourceSaver\JWTAuth\Providers\LaravelServiceProvider"
[INFO] Publishing assets.
Copying file [vendor/php-open-source-saver/jwt-auth/config/config.php] to [config/jwt.php] ..... DONE
+ pemrograman-web git:(main) ✘
```

Contoh Berhasil

3. Selanjutnya generate JWT key dengan menggunakan secret key, jalankan perintah berikut



php artisan jwt:secret

```
● → pemrograman-web git:(main) ✘ php artisan jwt:secret  
    jwt-auth secret set successfully.
```

Perintah ini akan menambahkan JWT_SECRET pada file .env

```
JWT_SECRET=rfLYX6bJ78oX95EYZdKs10fwUtDiT2BqEKBWfDW0tK6m7zVidQb0F0r1vbaxpvDg  
JWT_ALGO=HS256
```

Implementasi JWT Auth di Laravel

Masih pada project codelab yang sama seperti modul 5

1. Update Model User seperti code dibawah
Buka file **app/Models/User.php**

```
1 <?php
2
3 namespace App\Models;
4
5 // Import JWT Subjectnya
6 use PHPOpenSourceSaver\JWTAuth\Contracts\JWTSubject;
7 use Illuminate\Database\Eloquent\Factories\HasFactory;
8 use Illuminate\Foundation\Auth\User as Authenticatable;
9 use Illuminate\Notifications\Notifiable;
10
11 //Implementasi JWTSubject
12 class User extends Authenticatable implements JWTSubject
13 {
14     /** @use HasFactory<\Database\Factories\UserFactory> */
15     use HasFactory, Notifiable;
16
17     /**
18      * The attributes that are mass assignable.
19      *
20      * @var list<string>
21      */
22     protected $fillable = [
23         'name',
24         'email',
25         'password',
26     ];
27
28     /**
29      * The attributes that should be hidden for serialization.
30      *
31      * @var list<string>
32      */
33     protected $hidden = [
34         'password',
35         'remember_token',
36     ];
37
38     /**
39      * Get the attributes that should be cast.
40      *
41      * @return array<string, string>
42      */
43     protected function casts(): array
44     {
45         return [
46             'email_verified_at' => 'datetime',
47             'password' => 'hashed',
48         ];
49     }
50
51     // Implementasi method dari JWTSubject
52     public function getJWTIdentifier()
53     {
54         return $this->getKey();
55     }
56
57     public function getJWTCustomClaims()
58     {
59         return [];
60     }
61 }
62
```

2. Konfigurasi Guard Auth

Buka file **config/auth.php**



Ubah file auth.php dengan mengganti guard dari web menjadi api

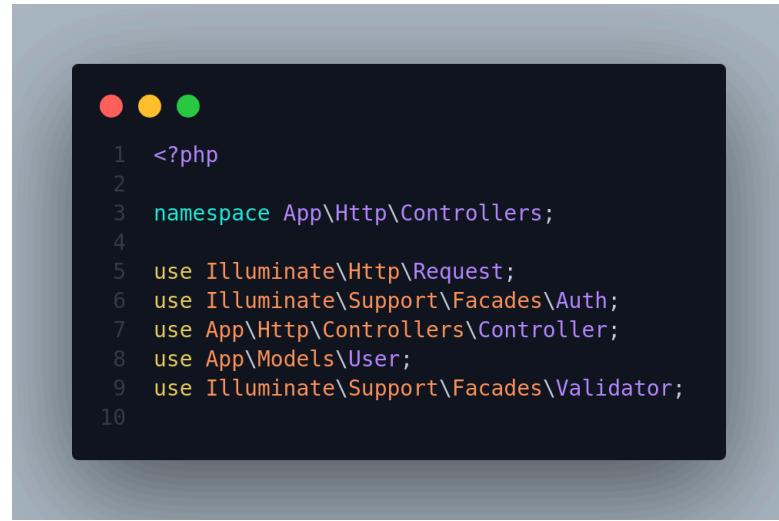
```
1 'defaults' => [
2     'guard' => env('AUTH_GUARD', 'api'),
3     'passwords' => env('AUTH_PASSWORD_BROKER', 'users'),
4 ],
```

Tambahkan api pada guard

```
1 'guards' => [
2     'web' => [
3         'driver' => 'session',
4         'provider' => 'users',
5     ],
6     'api' => [
7         'driver' => 'jwt',
8         'provider' => 'users',
9     ],
10 ],
```

3. Buat file baru AuthController dengan menjalankan perintah berikut
php artisan make:controller AuthController
4. Buka AuthController dan modifikasi seperti code berikut untuk membuat endpoint Auth
 - a. Tambahkan Use seperti berikut





```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7 use App\Http\Controllers\Controller;
8 use App\Models\User;
9 use Illuminate\Support\Facades\Validator;
10
```

b. Fungsi Register



```
1 public function register(Request $request)
2 {
3     $validator = Validator::make($request->all(), [
4         'name' => 'required',
5         'email' => 'required|email|unique:users',
6         'password' => 'required|min:6',
7     ]);
8
9     if ($validator->fails()) {
10         return response()->json($validator->errors(), 422);
11     }
12
13     $user = User::create([
14         'name' => $request->name,
15         'email' => $request->email,
16         'password' => bcrypt($request->password),
17     ]);
18
19     return response()->json(['message' => 'User successfully registered', 'user' => $user], 201);
20 }
```

c. Fungsi Login



```
1 public function login()
2 {
3     $credentials = request(['email', 'password']);
4
5     if (! $token = auth()->attempt($credentials)) {
6         return response()->json(['error' => 'Unauthorized'], 401);
7     }
8
9     return $this->respondWithToken($token);
10 }
```

d. Fungsi me biasanya digunakan untuk mendapatkan session pada client





```
● ● ●  
1 public function me()  
2 {  
3     return response()->json(auth()->user());  
4 }  
5
```

e. Fungsi logout



```
● ● ●  
1 public function logout()  
2 {  
3     auth()->logout();  
4  
5     return response()->json(['message' => 'Successfully logged out']);  
6 }
```

f. Fungsi respondWithToken, digunakan ketika login



```
● ● ●  
1 protected function respondWithToken($token)  
2 {  
3     return response()->json([  
4         'access_token' => $token,  
5         'token_type' => 'bearer',  
6         'expires_in' => auth()->factory()->getTTL() * 60  
7     ]);  
8 }
```

Full Code AuthController



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7 use App\Http\Controllers\Controller;
8 use App\Models\User;
9 use Illuminate\Support\Facades\Validator;
10
11 class AuthController extends Controller
12 {
13     public function register(Request $request)
14     {
15         $validator = Validator::make($request->all(), [
16             'name' => 'required',
17             'email' => 'required|email|unique:users',
18             'password' => 'required|min:6',
19         ]);
20
21         if ($validator->fails())
22             return response()->json($validator->errors(), 422);
23     }
24
25         $user = User::create([
26             'name' => $request->name,
27             'email' => $request->email,
28             'password' => bcrypt($request->password),
29         ]);
30
31         return response()->json(['message' => 'User successfully registered', 'user' => $user], 201);
32     }
33
34     public function login()
35     {
36         $credentials = request(['email', 'password']);
37
38         if (! $token = auth()->attempt($credentials)) {
39             return response()->json(['error' => 'Unauthorized'], 401);
40         }
41
42         return $this->respondWithToken($token);
43     }
44
45     public function me()
46     {
47         return response()->json(auth()->user());
48     }
49
50     public function logout()
51     {
52         auth()->logout();
53
54         return response()->json(['message' => 'Successfully logged out']);
55     }
56
57     protected function respondWithToken($token)
58     {
59         return response()->json([
60             'access_token' => $token,
61             'token_type' => 'bearer',
62             'expires_in' => auth()->factory()->getTTL() * 60
63         ]);
64     }
65 }
66 }
```

5. Routing API endpoint Auth tadi pada **routes/api.php**



```

1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\TodoController;
6 use App\Http\Controllers\AuthController;
7
8 Route::get('/user', function (Request $request) {
9     return $request->user();
10 })->middleware('auth:sanctum');
11 Route::apiResource('todos', TodoController::class);
12
13 Route::group(['middleware' => 'api', 'prefix' => 'auth'], function ($router) {
14     $router->post('register', [AuthController::class, 'register']);
15     $router->post('login', [AuthController::class, 'login']);
16     $router->post('logout', [AuthController::class, 'logout'])->middleware('auth:api');
17     $router->get('me', [AuthController::class, 'me'])->middleware('auth:api');
18 });
19

```

6. Jalankan perintah `php artisan route:list` untuk melihat apakah endpoint sudah di regis pada routes

```

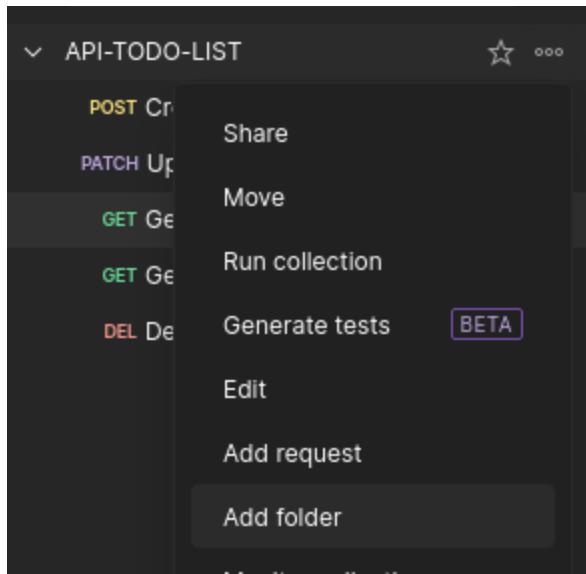
+-- pemerograman-web git:(main) ✘ php artisan route:list
GET|HEAD / ..... AuthController@login
POST api/auth/login ..... AuthController@login
POST api/auth/logout ..... AuthController@logout
GET|HEAD api/auth/ ..... AuthController@register
POST api/auth/register ..... AuthController@register
GET|HEAD api/todos ..... todos.index > TodoController@index
POST api/todos ..... todos.store > TodoController@store
GET|HEAD api/todos/{todo} ..... todos.show > TodoController@show
PUT|PATCH api/todos/{todo} ..... todos.update > TodoController@update
DELETE api/todos/{todo} ..... todos.destroy > TodoController@destroy
GET|HEAD api/user ..... sanctum/csrf-cookie
GET|HEAD sanctum/csrf-cookie ..... Laravel\Sanctum\CSRF\CookieController@show
GET|HEAD storage/{path} ..... storage.local
GET|HEAD up .....

```

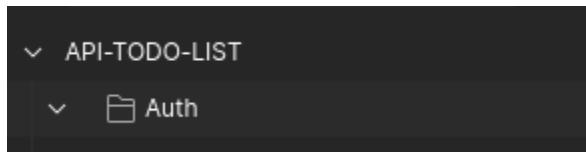
Pembuatan Postman untuk endpoint Auth

Setelah berhasil pada implementasi JWT, untuk testing apakah sudah berhasil digunakan atau tidak kita perlu testing API untuk route Auth dengan menggunakan postman

1. Tambahkan folder Auth
Klik kanan kemudian Add Folder

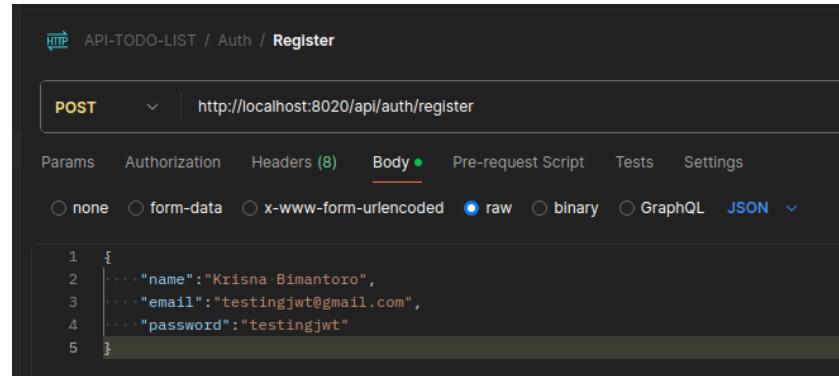


Ubah nama menjadi Auth



2. Tambahkan 4 request untuk masing-masing endpoint

a. Register [POST]



```

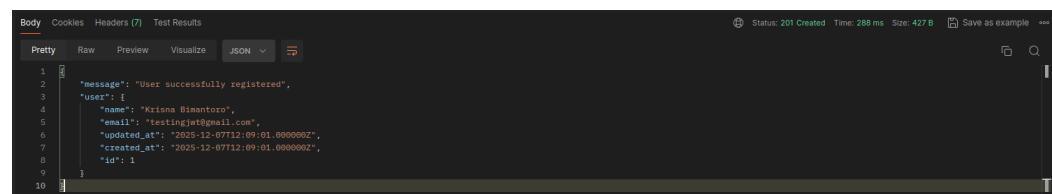
HTTP API-TODO-LIST / Auth / Register

POST http://localhost:8020/api/auth/register

Params Authorization Headers (8) Body
none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2   ...
3   "name": "Krisna Bimantoro",
4   "email": "testingjwt@gmail.com",
5   "password": "testingjwt"
6 }
```

Klik send dan pastikan response seperti di bawah

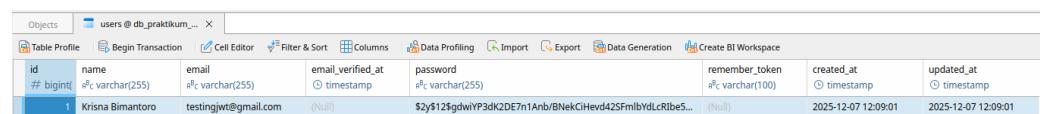


```

Body Cookies Headers (7) Test Results
Pretty Raw Preview Visualize JSON

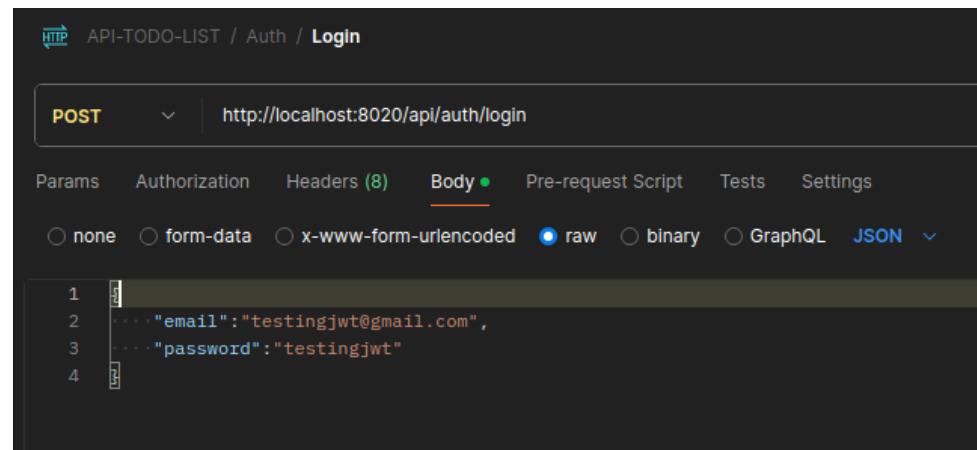
1 {
2   "message": "User successfully registered",
3   "user": {
4     "name": "Krisna Bimantoro",
5     "email": "testingjwt@gmail.com",
6     "updated_at": "2025-12-07T12:09:01.000000Z",
7     "created_at": "2025-12-07T12:09:01.000000Z",
8     "id": 1
9   }
10 }
```

Dan pada database seperti berikut



id	name	email	email_verified_at	password	remember_token	created_at	updated_at
1	Krisna Bimantoro	testingjwt@gmail.com	(Null)	\$2y\$12\$gdwiYP3dKE7n1AnB/NekCiHvd42SFmIbYdLcRibe5...	(Null)	2025-12-07 12:09:01	2025-12-07 12:09:01

b. Login [POST]



```

1 {
2   "email": "testingjwt@gmail.com",
3   "password": "testingjwt"
4 }

```

Perhatikan responsenya



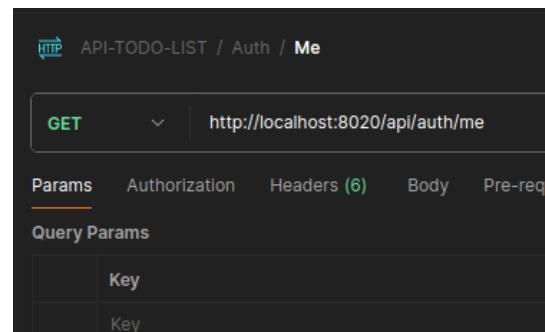
```

1 {
2   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiDjd0dGhhdG9tLGVb6j9jYwcb03N60jgwMjA/yXbpL2F1dgqvG9naW41LCjpyXQ1OjE3NjUwMGk1NzUsImV4cC16MtC2NTExM23NSw1bm3mIjoxNzY1MTAT9Tc1LC3qdGk10132Z1M2cWvVGZ6Yw9tbGM5IiwiJ3ViIjoiMSisInBydiI6IjIzIw0jYzg5MD01MjkyNwRRIwz1ln2AxYz0wmwBg5MhRiIN2E10Tc22yciEQ.ksnPi1IDzU5-880oh1Vgrq351_cxeCtkg9hywRzxZc",
3   "token_type": "bearer",
4   "expires_in": 3668
5 }

```

Bisa kalian coba acces tokennya di decode pada website <https://www.jwt.io/>

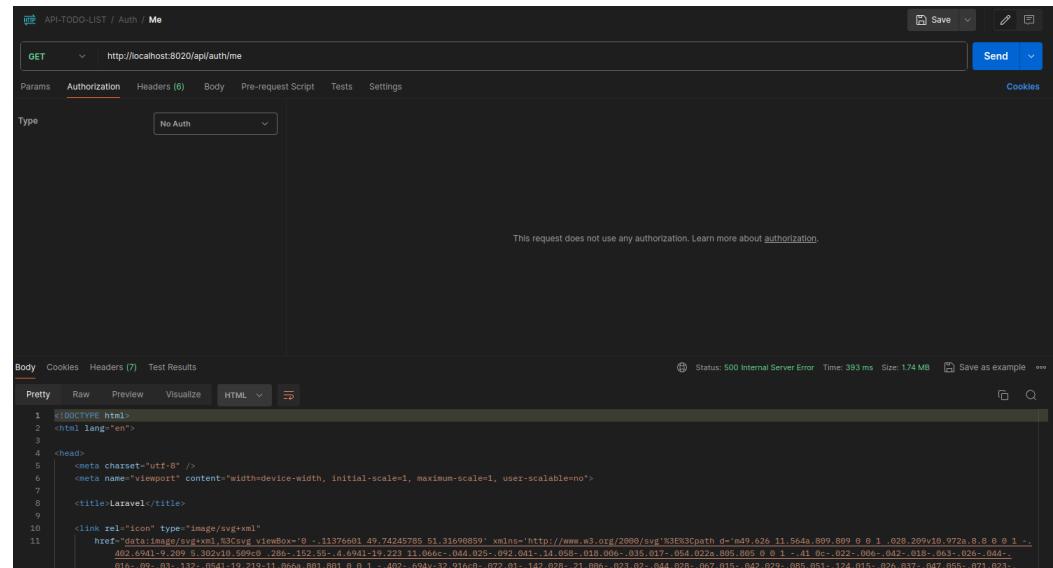
c. Me [GET]



Key
Key

Kalau langsung di send pasti akan muncul error seperti di bawah





HTTP API-TODO-LIST / Auth / Me

GET http://localhost:8020/api/auth/me

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Type No Auth

This request does not use any authorization. Learn more about [authorization](#).

Status: 500 Internal Server Error Time: 393 ms Size: 1.74 MB Save as example

Body Cookies Headers (7) Test Results

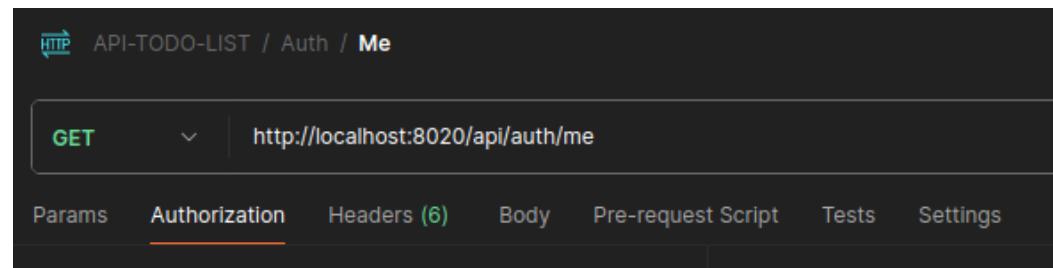
Pretty Raw Preview Visualizer HTML

```

1 <!DOCTYPE html>
2 <html lang="en">
3 
4 <head>
5   <meta charset="utf-8" />
6   <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
7 
8   <title>Laravel</title>
9 
10  <link rel="icon" type="image/svg+xml" href="data:image/svg+xml,%3Csvg viewBox='0 0 113756691 49.74245785 51.31696859' xmlns='http://www.w3.org/2000/svg'%3E%3Cpath d='m9.626 11.56d.899.899 0 0 1 .028.299v18.972a.8.8 0 0 1 -.402.694 9.299 5.302v10.596c0 .286-.152 .55-.4.694l-19.223 11.666-.044.025-.092.041-.14.058-.018.006-.035.017-.054.022a.805.805 0 0 1 -.41.0c-.022-.006-.042-.018-.063-.026-.044-.016-.09-.03.132-.054l-19.219-11.066a.891.891 0 0 1 -.402-.694v-32.916a.072.072 142.028-.21.086-.023.02-.044.028-.067.015-.042.029-.085.051-.124.015-.076.037-.047.055-.071.023'/%3E%3C/svg>" data-bbox="278 328 868 359"/>

```

Solusinya dengan menambahkan bearer token pada Authorization

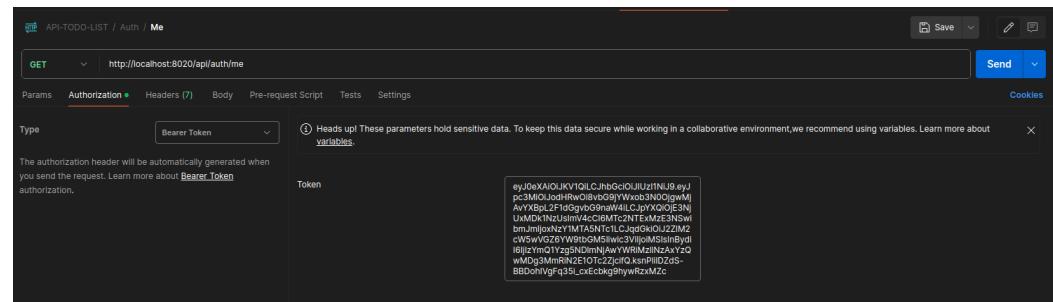


HTTP API-TODO-LIST / Auth / Me

GET http://localhost:8020/api/auth/me

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Pada Type ubah menjadi bearer token dan tambahkan token yang di hasilkan dari response login tadi



HTTP API-TODO-LIST / Auth / Me

GET http://localhost:8020/api/auth/me

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

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

Token

eyJ0eXAiOiJKV1QiLCJhbGciOiUzI1NiJ9.eyJpc3MiUlcsdIiRwQiblc0c9yfWxzb3N0QipgMjAvYXBpL2FdfG9yv09naW4lLCJpjtXQjOjE3NjUxMDk1NzUsImV4cCI6MTc2NTExMzE3NmwibmJmjmxNzYtMTA5MS11LCJqdGkiOiIxZDmZ2dwIiwidmVyc2lvbiI6IjIwMjAxMjIwMjIwMjIwMSIsIngtIjIyLzmtQ1YzsnDmNjawyWRlMzlnAxYz0wMDg3MmRn2E1OTc2ZjcfQ.knPliIDZdS-BBDDohVgFq35_cxEcbkg9hywRzxMzC

Contoh response yang berhasil



Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "id": 1,
3   "name": "Krisna Bimantoro",
4   "email": "testingjwt@gmail.com",
5   "email_verified_at": null,
6   "created_at": "2025-12-07T12:09:01.000000Z",
7   "updated_at": "2025-12-07T12:09:01.000000Z"
8

```

d. Logout

HTTP API-TODO-LIST / Auth / Logout

POST http://localhost:8020/api/auth/logout

Tambahkan Authorization pada endpoint logout

POST http://localhost:8020/api/auth/logout

Authorization: Bearer Token

Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eJ...

Klik send

POST http://localhost:8020/api/auth/logout

Authorization: Bearer Token

Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eJ...

Status: 200 OK Time: 76 ms Size: 258 B Save as example

```

1 {
2   "message": "Successfully logged out"
3

```

Implementasi middleware untuk endpoint todo

Mari kita coba untuk memodifikasi endpoint untuk POST, PATCH dan DELETE implementasi middleware dan untuk GET bisa di get public.

1. Modifikasi route di routes/api.php



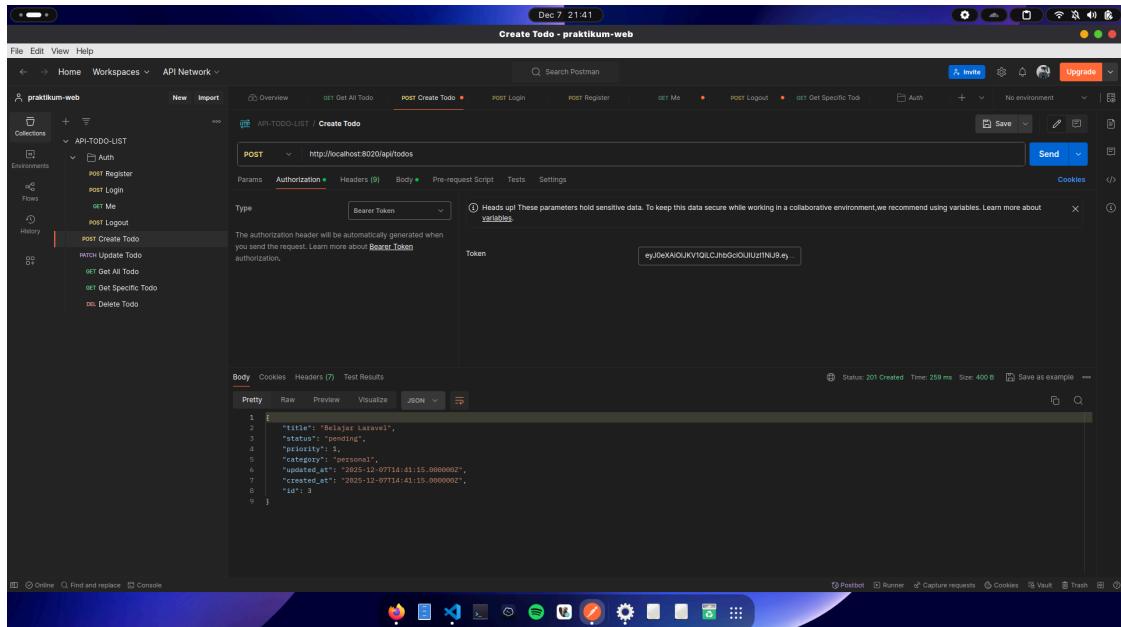
```

1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\TodoController;
6 use App\Http\Controllers\AuthController;
7
8 // Auth routes
9 Route::group(['middleware' => 'api', 'prefix' => 'auth'], function ($router) {
10     $router->post('register', [AuthController::class, 'register']);
11     $router->post('login', [AuthController::class, 'login']);
12     $router->post('logout', [AuthController::class, 'logout'])->middleware('auth:api');
13     $router->get('me', [AuthController::class, 'me'])->middleware('auth:api');
14 });
15
16 // Todo routes with authentication
17 Route::middleware(['auth:api'])->group(function () {
18     $router->post('todos', [TodoController::class, 'store']);
19     $router->put('todos/{todo}', [TodoController::class, 'update']);
20     $router->delete('todos/{todo}', [TodoController::class, 'destroy']);
21 });
22
23 // Public todo routes
24 Route::get('todos', [TodoController::class, 'index']);
25 Route::get('todos/{todo}', [TodoController::class, 'show']);
26

```

Kita mengimplementasikan middleware untuk endpoint POST, PATCH dan DELETE untuk method GET akan digunakan untuk public.

2. Tambahkan bearer token untuk POST, PATCH dan DELETE

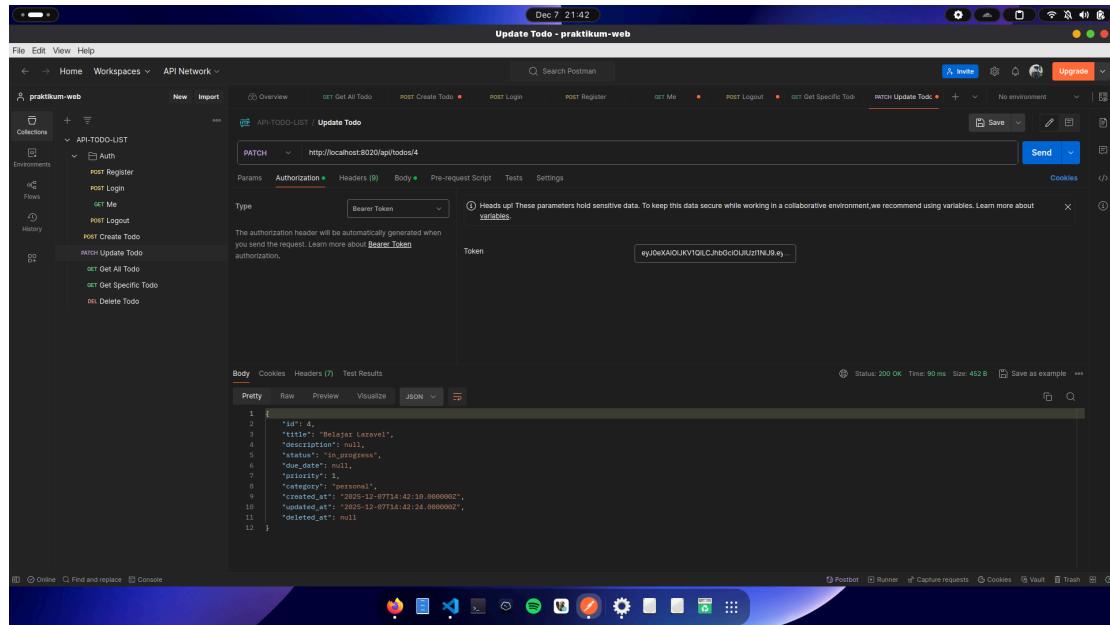


The screenshot shows the Postman interface with a collection named "praktikum-web". A POST request is being made to the endpoint `http://localhost:8020/api/todos`. In the "Authorization" field under "Headers", the type is set to "Bearer Token" and a token is provided: `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eJ`. The response status is 201 Created, and the response body is a JSON object representing a new todo item:

```

1 {
2     "title": "Belajar Laravel",
3     "status": "pending",
4     "priority": 1,
5     "category": "personal",
6     "created_at": "2025-12-07T14:41:15.000000Z",
7     "updated_at": "2025-12-07T14:41:15.000000Z",
8     "id": 3
9 }

```



The screenshot shows the Postman interface with the following details:

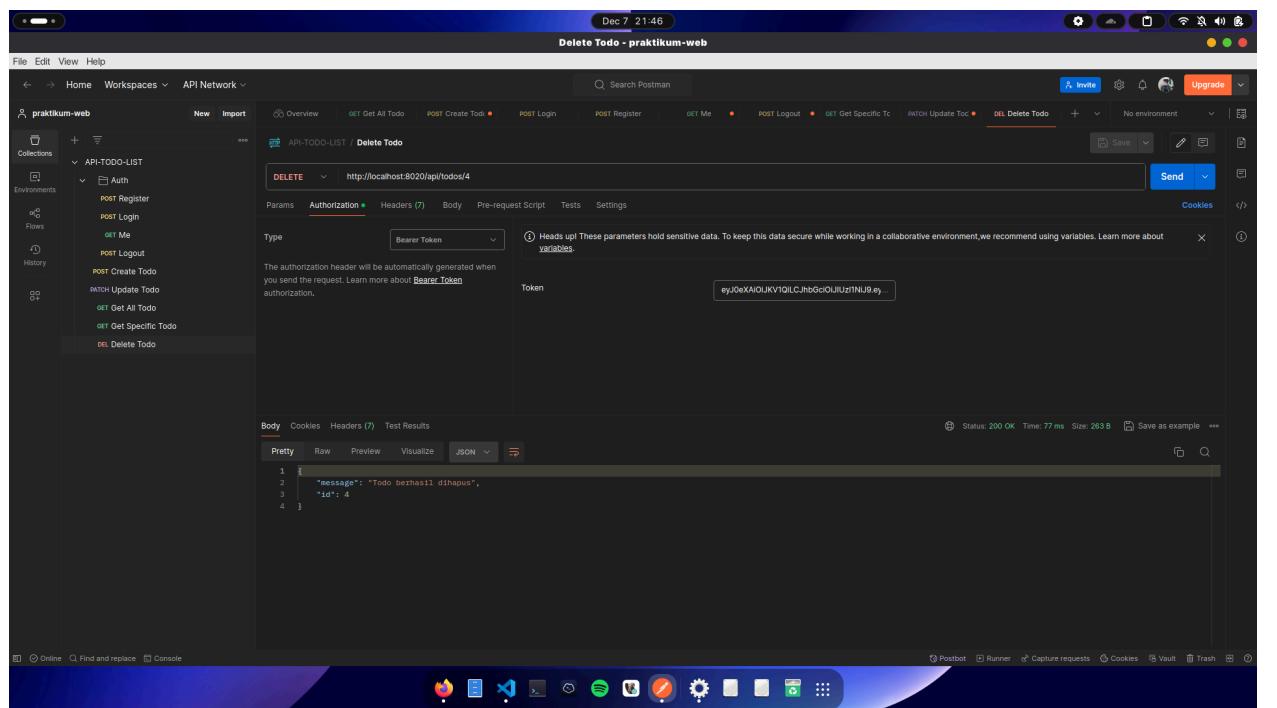
- Collection:** praktikum-web
- Endpoint:** API-TODO-LIST / Update Todo
- Type:** PATCH
- URL:** http://localhost:9020/api/todos/4
- Authorization:** Bearer Token
- Body (JSON):**

```

1 {
2     "id": 4,
3     "title": "Travel to Bali",
4     "description": null,
5     "status": "in progress",
6     "due_date": null,
7     "priority": 1,
8     "completed": false,
9     "created_at": "2025-12-07T14:42:10.000000Z",
10    "updated_at": "2025-12-07T14:42:14.000000Z",
11    "deleted_at": null
12 }

```

- Status:** 200 OK | Time: 90 ms | Size: 452 B



The screenshot shows the Postman interface with the following details:

- Collection:** praktikum-web
- Endpoint:** API-TODO-LIST / Delete Todo
- Type:** DELETE
- URL:** http://localhost:9020/api/todos/4
- Authorization:** Bearer Token
- Body (JSON):**

```

1 {
2     "message": "Todo berhasil dihapus",
3     "id": 4
4 }

```

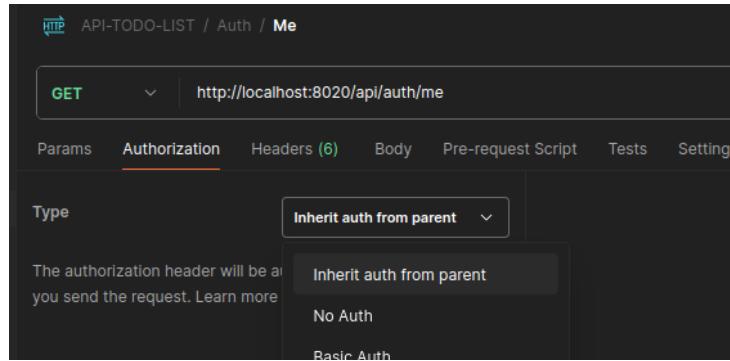
- Status:** 200 OK | Time: 77 ms | Size: 283 B

Informasi Opsiional Postman

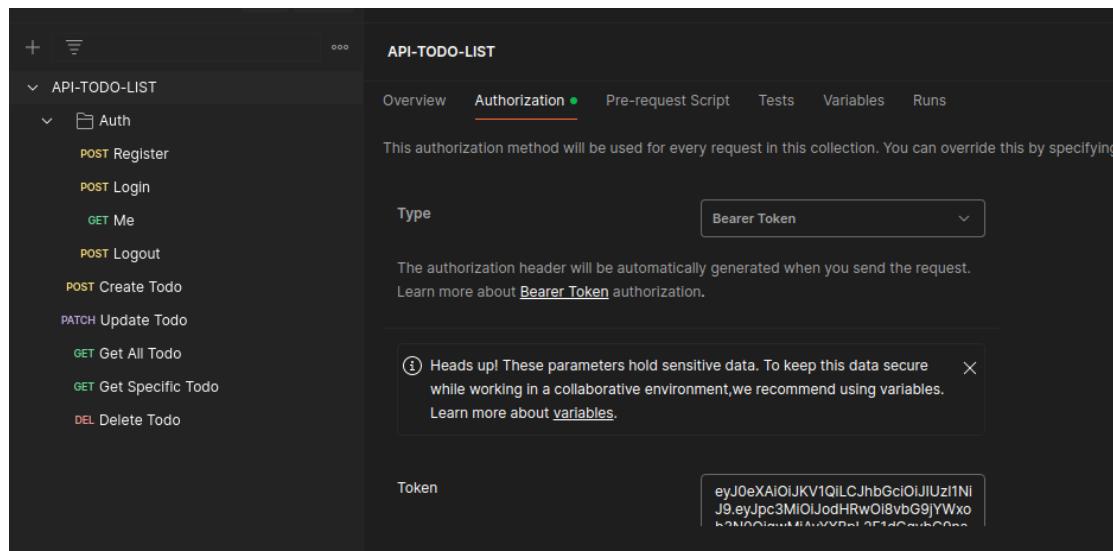
- Melakukan penambahan bearer token secara manual akan mengakibatkan hal yang repetitif terdapat satu fitur pada postman yaitu *inherit auth from parent* yaitu kita cuma mengubah bearer token yang ada pada parent, parent disini adalah collection kita.

Ganti tipe endpoint yang memerlukan auth menjadi *inherit auth from parent*

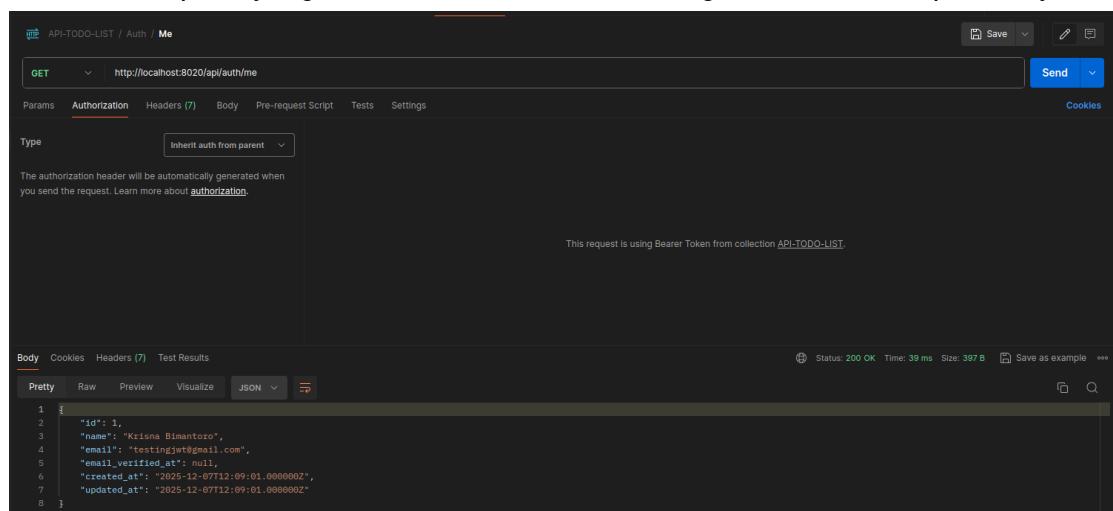




Klik pada collection kemudian pindah ke tab authorization, ubah type menjadi bearer token dan masukan token dari hasil return loginnya.



Otomatis endpoint yang memerlukan token akan mengambil token dari parent nya



```

1 {
2   "id": 1,
3   "name": "Kriska Bimantoro",
4   "email": "testingpyt@gmail.com",
5   "email_verified_at": null,
6   "created_at": "2025-12-07T12:09:01.000000Z",
7   "updated_at": "2025-12-07T12:09:01.000000Z"
8 }

```

Jangan lupa di save adik-adik



FILE STORAGE

Laravel File Storage merupakan salah satu bagian dari fungsi form yang dipergunakan untuk mengunggah atau mengupload file pada server. Penggunaan fitur ini misalnya pada halaman product, maka admin dapat mengisikan form untuk berbagai data produk termasuk mengupload gambar produk yang akan difungsikan sebagai foto produk.

Laravel menyediakan sistem abstraksi *file* yang kuat berkat integrasi dengan *library Flysystem*. Ini memungkinkan Anda bekerja dengan berbagai *driver* penyimpanan (disk) seperti penyimpanan lokal, S3 (Amazon Web Services), atau Google Cloud Storage dengan menggunakan sintaks yang seragam dan mudah.

Disk local vs. Disk public:

- local (driver: local): Defaultnya menunjuk ke folder storage/app/. File di sini tidak dapat diakses langsung dari web (browser). Cocok untuk data sensitif.
- public (driver: local): Defaultnya menunjuk ke folder storage/app/public/. File di sini dimaksudkan untuk diakses publik (gambar, dokumen). Namun, folder ini juga tidak bisa diakses langsung dari URL tanpa langkah tambahan.

Masalah utama dari Disk public adalah meskipun file disimpan di storage/app/public/, server web (browser) hanya dapat mengakses folder yang berada di dalam direktori public/ utama Laravel (tempat file index.php berada). Untuk mengatasi ini, kita perlu membuat **Symbolic Link (Symlink)**.

Apa itu Symlink? Symlink adalah shortcut atau jalan pintas yang menghubungkan folder public/storage ke folder fisik storage/app/public.

Implementasi File Storage Laravel

1. Jalankan perintah berikut pada terminal

php artisan storage:link

```
→ pemrograman-web git:(main) php artisan storage:link
```

```
[INFO] The [public/storage] link has been connected to [storage/app/public].
```

Bisa dilihat untuk base path file terletak di storage/app/public

2. Buat migrasi database baru untuk menambahkan kolom file_path pada database dengan perintah di bawah

php artisan make:migration add_file_path_to.todos_table --table=todos

```
→ pemrograman-web git:(main) php artisan make:migration add_file_path_to.todos_table --table=todos
```

```
[INFO] Migration [database/migrations/2025_12_07_152801_add_file_path_to.todos_table.php] created successfully.
```



3. Modifikasi file

database/migrations/2025_12_07_152801_add_file_path_to.todos_table.php



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::table('todos', function (Blueprint $table) {
15             $table->string('file_path')->nullable()->after('category');
16         });
17     }
18
19     /**
20      * Reverse the migrations.
21      */
22     public function down(): void
23     {
24         Schema::table('todos', function (Blueprint $table) {
25             $table->dropColumn('file_path');
26         });
27     }
28 };
29
```

4. Ubah file app/Models/Todo.php



```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6 use Illuminate\Database\Eloquent\SoftDeletes;
7
8 class Todo extends Model
9 {
10     use SoftDeletes;
11
12     protected $fillable = [
13         'title', 'description', 'status', 'due_date', 'priority', 'category', 'file_path'
14     ];
15
16     protected $casts = [
17         'due_date' => 'date',
18         'priority' => 'integer',
19     ];
20 }
```



5. Ubah file app/Http/Controllers/TodoController.php

a. Store

```

1
2 public function store(Request $request)
3 {
4     $data = $request->validate([
5         'title'      => 'required|string|max:150',
6         'description' => 'nullable|string',
7         'status'      => 'nullable|in:pending,in_progress,done',
8         'due_date'    => 'nullable|date',
9         'priority'    => 'nullable|integer|min:1|max:3',
10        'category'   => 'nullable|in:personal,work,study,others',
11        'file'        => 'nullable|file|max:10240', // max 10MB
12    ]);
13
14    // Handle file upload
15    if ($request->hasFile('file')) {
16        $file = $request->file('file');
17        $fileName = time() . '_' . $file->getClientOriginalName();
18        $filePath = $file->storeAs('todos', $fileName, 'public');
19        $data['file_path'] = $filePath;
20    }
21
22    $todo = Todo::create($data);
23    return response()->json($todo, 201);
24 }
25

```

b. Update

```

1 public function update(Request $request, Todo $todo)
2 {
3     $data = $request->validate([
4         'title'      => 'sometimes|required|string|max:150',
5         'description' => 'sometimes|nullable|string',
6         'status'      => 'sometimes|in:pending,in_progress,done',
7         'due_date'    => 'sometimes|nullable|date',
8         'priority'    => 'sometimes|integer|min:1|max:3',
9         'category'   => 'sometimes|in:personal,work,study,others',
10        'file'        => 'nullable|file|max:10240', // max 10MB
11    ]);
12
13    // Handle file upload
14    if ($request->hasFile('file')) {
15        // Delete old file if exists
16        if ($todo->file_path && \Storage::disk('public')->exists($todo->file_path)) {
17            \Storage::disk('public')->delete($todo->file_path);
18        }
19
20        $file = $request->file('file');
21        $fileName = time() . '_' . $file->getClientOriginalName();
22        $filePath = $file->storeAs('todos', $fileName, 'public');
23        $data['file_path'] = $filePath;
24    }
25
26    $todo->update($data);
27    return response()->json($todo);
28 }

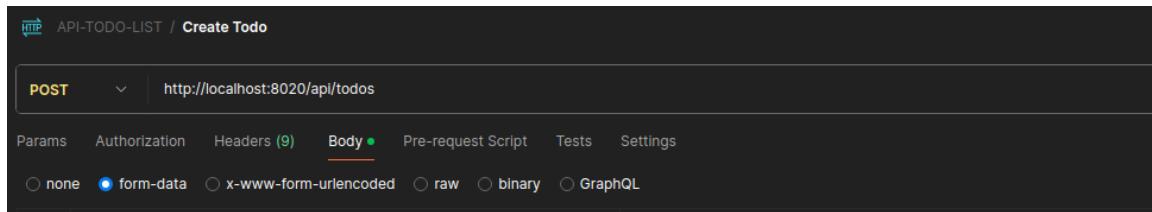
```

c. Delete

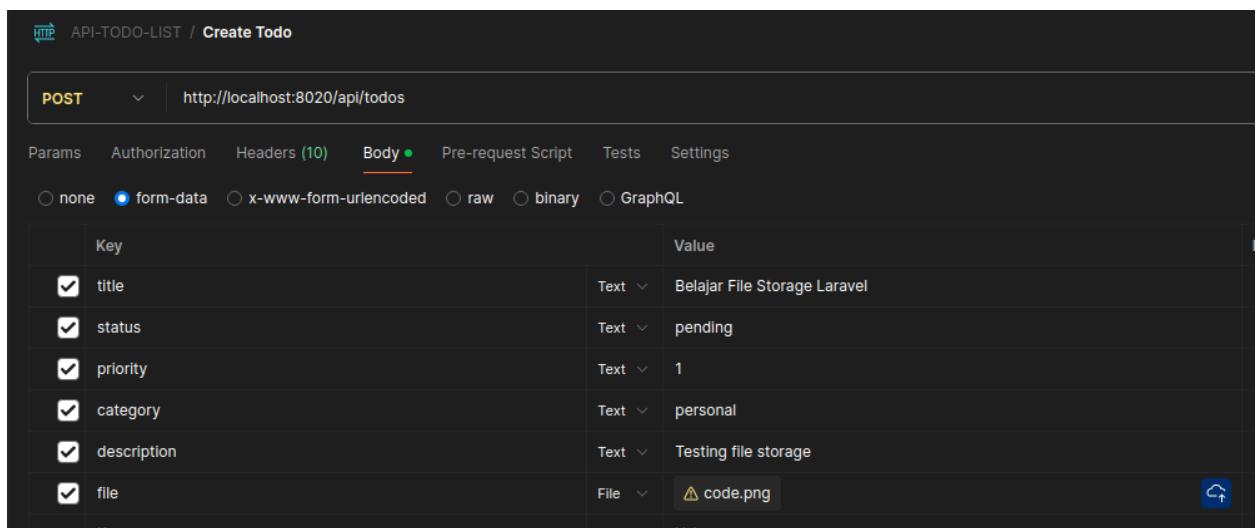
Untuk method ini sebenarnya tidak perlu diimplementasikan delete file storage, karena soft delete, jika kita merestore dan menghapus file maka file tidak akan terlihat.

Implementasi Pada Postman

- Ubah tipe requestnya menjadi form data

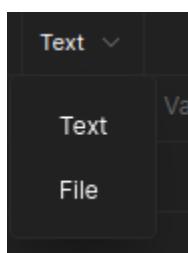


- Isi key sesuai dengan pada json sebelumnya, dan tambahkan key file

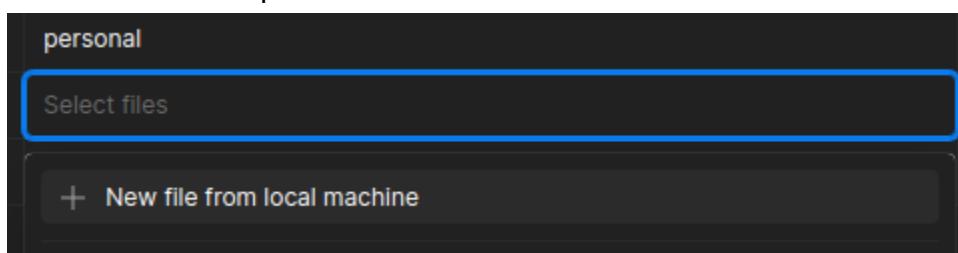


Key	Value
title	Belajar File Storage Laravel
status	pending
priority	1
category	personal
description	Testing file storage
file	code.png

Ubah Text menjadi File



Klik select files dan pilih new file from local machine



- Klik send

API-TODO-LIST / Create Todo

POST http://localhost:8020/api/todos

Params Authorization Headers (10) Body Tests Settings Cookies

Body (10) form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> title	Text Belajar File Storage Laravel		
<input checked="" type="checkbox"/> status	Text pending		
<input checked="" type="checkbox"/> priority	Text 1		
<input checked="" type="checkbox"/> category	Text personal		
<input checked="" type="checkbox"/> description	Text Testing file storage		
<input checked="" type="checkbox"/> file	File 		
Key	Value	Description	

Body Cookies (2) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

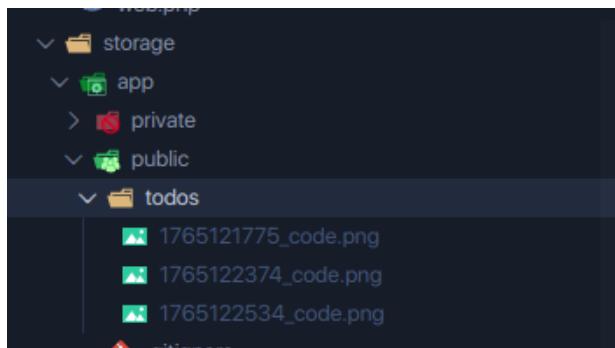
1
2   "title": "Belajar File Storage Laravel",
3   "description": "Testing file storage",
4   "status": "pending",
5   "priority": 1,
6   "category": "personal",
7   "file_path": "todos/1765122534_code.png",
8   "updated_at": "2025-12-07T15:48:54.000000Z",
9   "created_at": "2025-12-07T15:48:54.000000Z",
10  "id": 7
11

```

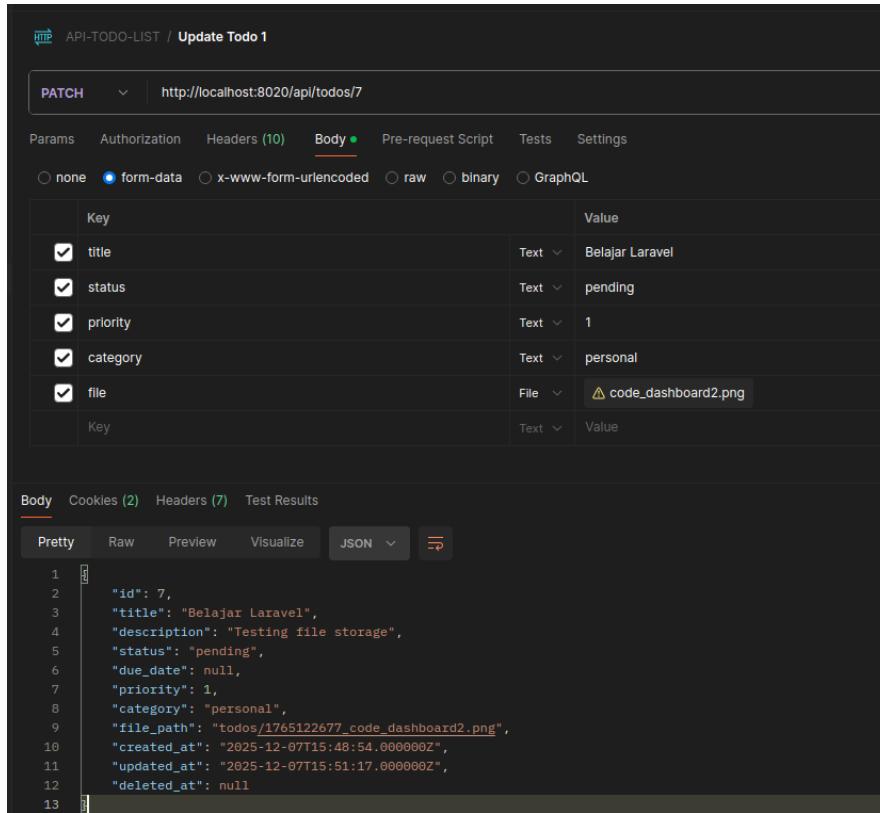
Status: 201 Created Time: 111 ms Size: 491 B Save as example

Bisa dilihat file_path memiliki ke path directory di public

4. Validasi pada local path



5. Implementasikan juga pada patch



The screenshot shows a Postman interface for updating a todo item. The URL is `http://localhost:8020/api/todos/7`. The body is set to `form-data` with the following fields:

Key	Value
<input checked="" type="checkbox"/> title	Text: Belajar Laravel
<input checked="" type="checkbox"/> status	Text: pending
<input checked="" type="checkbox"/> priority	Text: 1
<input checked="" type="checkbox"/> category	Text: personal
<input checked="" type="checkbox"/> file	File: code_dashboard2.png

The response body is:

```

1
2   "id": 7,
3   "title": "Belajar Laravel",
4   "description": "Testing file storage",
5   "status": "pending",
6   "due_date": null,
7   "priority": 1,
8   "category": "personal",
9   "file_path": "todos/1765122677_code_dashboard2.png",
10  "created_at": "2025-12-07T15:48:54.000000Z",
11  "updated_at": "2025-12-07T15:51:17.000000Z",
12  "deleted_at": null
13

```

Cara akses image melalui url local

http://localhost:8020/storage/todos/file_path (pada response)

Contoh

http://localhost:8020/storage/todos/1765122677_code_dashboard2.png

CODELAB

1. Lakukan modifikasi pada codelab sebelumnya untuk mengimplementasikan middleware dan file storage.
2. Lakukan instalasi dan implementasi middleware auth seperti pada modul ([Instalasi JWT Auth di Laravel](#) dan [Implementasi JWT Auth di Laravel](#))
3. Lakukan instalasi dan implementasi file storage seperti pada modul ([Implementasi File Storage Laravel](#))
4. Modifikasi collection postman dengan contoh seperti pada modul.
5. Penilaian Codelab:
 - a. Jika memungkinkan selesai pada pekan materi akan mendapatkan nilai 100
 - b. Jika tidak memungkinkan selesai pada pekan materi, silahkan ditunjukkan pada pekan demo dan mendapatkan nilai 70.



TUGAS

Implementasikan middleware auth dan file storage dengan tema yang kalian gunakan dan implementasikan dengan ketentuan seperti berikut:

1. Buat endpoint untuk Login, Register, Logout dan Me, seperti pada modul.
2. Modifikasi 5 endpoint yang sudah dibuat sebelumnya dengan ketentuan berikut
 - a. Get All -> Public/tanpa middleware
 - b. Get Detail > Public/tanpa middleware
 - c. Create -> middleware auth
 - d. Update dengan param id -> middleware auth
 - e. Delete dengan param id -> middleware auth
3. Implementasi file storage pada endpoint Create dan Update, ketentuan file
 - a. File harus dibawah <5 MB
4. Modifikasi Collection Postman dan request http untuk masing-masing endpoint.



KRITERIA & DETAIL PENILAIAN

Kriteria Penilaian	Persentase Penilaian
Codelab	Total 15%
Tugas	
Mengimplementasikan Middleware Auth	20%
Mengimplementasikan File Storage	15%
Modifikasi Collection Postman	10%
Pemahaman	
Kemampuan menjelaskan tugas dan menjawab pertanyaan	40%
Total	Total 100%