

PEMROGRAMAN WEB BERORIENTASI SERVICES

“BACKEND”



NateSare :

Andika Wisnu Wijaya

Filllaah Al Farizi

Gilang Aji Panutan

UNIVERSITAS TEKNOKRAT INDONESIA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
INFORMATIKA
2023

Struktur Database

e_learning users <ul style="list-style-type: none">id : bigint(20) unsignedusername : varchar(255)name : varchar(255)image : textemail : varchar(255)password : varchar(255)api_token : textlevel : enum('admin','teacher','student')paid_status : tinyint(1)remember_token : varchar(100)created_at : timestampupdated_at : timestampdeleted_at : timestamp	e_learning comments <ul style="list-style-type: none">id : bigint(20) unsigneduser_id : bigint(20) unsignedpost_slug : textbody : longtextcreated_at : timestampupdated_at : timestampdeleted_at : timestamp	e_learning posts <ul style="list-style-type: none">id : bigint(20) unsignedslug : texttitle : textcategory_id : bigint(20) unsignedimage : textexcerpt : textbody : longtextstatus : enum('free','paid')publish : tinyint(1)author_id : bigint(20) unsignedpublished_at : timestampcreated_at : timestampupdated_at : timestampdeleted_at : timestamp
e_learning personal_access_tokens <ul style="list-style-type: none">id : bigint(20) unsignedtokenable_type : varchar(255)tokenable_id : bigint(20) unsignedname : varchar(255)token : varchar(64)abilities : textlast_used_at : timestampexpires_at : timestampcreated_at : timestampupdated_at : timestamp	e_learning categories <ul style="list-style-type: none">id : bigint(20) unsignedslug : textname : textcreated_at : timestampupdated_at : timestamp	

User memiliki 3 role yaitu: admin, teacher, dan student. Hanya admin dan teacher yang dapat membuat materi / postingan.

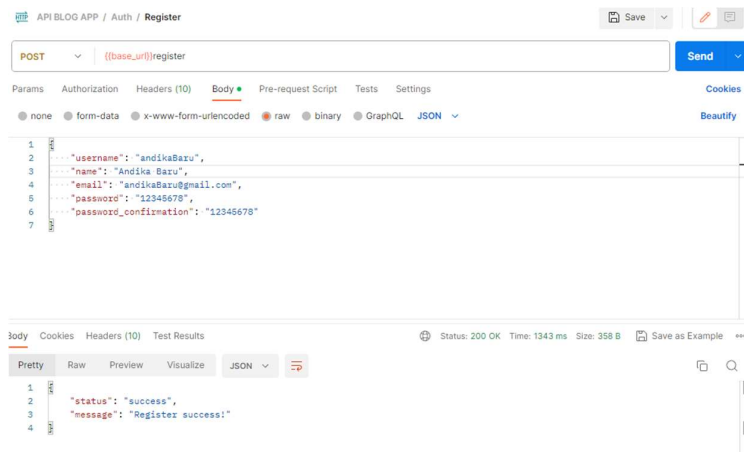
Semua user dapat membuat komentar.

Authentikasi

Register

```
1 class Authentication extends Controller
2 {
3     public function register(Request $request) {
4         $validateData = $request->validate([
5             'username' => 'required|alpha:ascii|max:255|unique:users,username',
6             'name' => 'required|max:255',
7             'email' => 'required|email:dns|unique:users,email',
8             'password' => 'required|min:8|max:255|confirmed'
9         ]);
10
11         if($request->level) {
12             $validateData['level'] = $request->level;
13         } else {
14             $validateData['level'] = 'student';
15         }
16
17         if($request->paid_status) {
18             $validateData['paid_status'] = $request->paid_status;
19         } else {
20             $validateData['paid_status'] = 0;
21         }
22
23         $validateData['password'] = Hash::make($validateData['password']);
24
25         User::create($validateData);
26
27         return response()->json([
28             'status' => 'success',
29             'message' => 'Register success!'
30         ]);
31     }
32 }
```

Controller untuk registrasi user, memerlukan inputan username, name, email, dan password dengan ketentuan tertentu. Inputan password user akan dilakukan hashing secara otomatis di backend untuk keamanan.

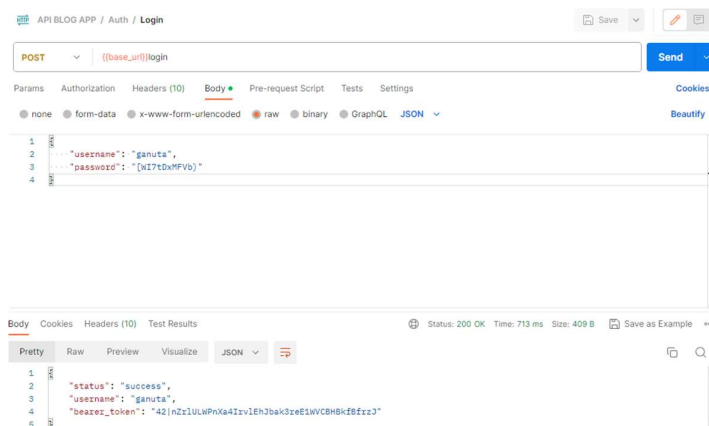


Melakukan pengecekan pada controller register menggunakan postman

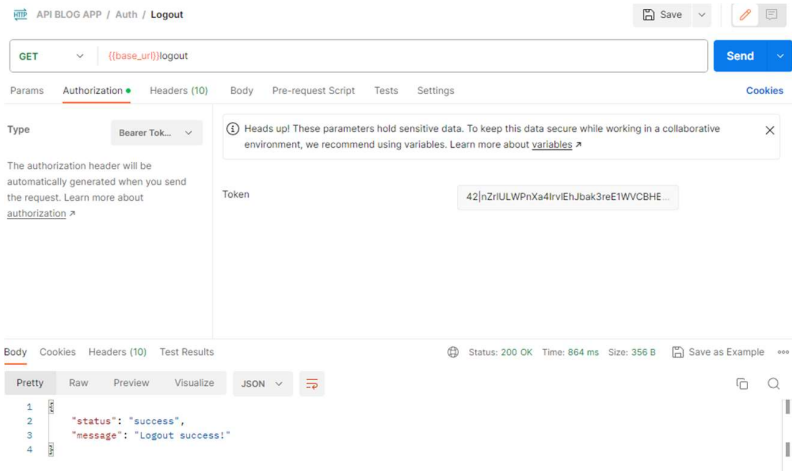
Login dan Logout

```
1 public function login(Request $request) {
2     $request->validate([
3         'username' => 'required',
4         'password' => 'required'
5     ]);
6
7     $user = User::firstWhere('username', $request->username);
8
9     if(!$user || !Hash::check($request->password, $user->password)) {
10         return response()->json([
11             'status' => 'failed',
12             'message' => 'Wrong username or password!',
13         ]);
14     }
15
16     $token = $user->createToken('user_token' . $user->username)->plainTextToken;
17
18     User::where('id', $user->id)->update([
19         'api_token' => $token
20     ]);
21
22     return response()->json([
23         'status' => 'success',
24         'username' => $user->username,
25         'bearer_token' => $token,
26     ]);
27 }
28
29 public function logout(Request $request) {
30
31     User::where('id', Auth::user()->id)->update([
32         'api_token' => null
33     ]);
34
35     $request->user()->currentAccessToken()->delete();
36
37     return response()->json([
38         'status' => 'success',
39         'message' => 'Logout success!',
40     ]);
41 }
```

Controller untuk login dan logout user, memerlukan inputan username dan password untuk login user. Ketika logout, token yang di bawa oleh user akan dihapus.



Test login menggunakan postman



Test logout menggunakan postman

Postingan Model

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Illuminate\Database\Eloquent\Relations\HasMany;
9  use Cviebrock\EloquentSluggable\Sluggable;
10
11 class Post extends Model
12 {
13     use HasFactory, Sluggable;
14
15     protected $guarded = ['id'];
16
17     protected $with = ['author', 'comments:id,user_id,post_slug,body,created_at', 'category'];
18
19     public function getRouteKeyName(): string {
20         return 'slug';
21     }
22
23     public function author(): BelongsTo {
24         return $this->belongsTo(User::class, 'author_id', 'id');
25     }
26
27     public function comments(): HasMany {
28         return $this->hasMany(Comment::class, 'post_slug', 'slug');
29     }
30
31     public function category(): BelongsTo {
32         return $this->belongsTo(Category::class, 'category_id', 'id');
33     }
34
35     public function sluggable(): array
36     {
37         return [
38             'slug' => [
39                 'source' => 'post_title'
40             ]
41         ];
42     }
43 }
```

Model postingan ini saya menggunakan eloquent.

Resource

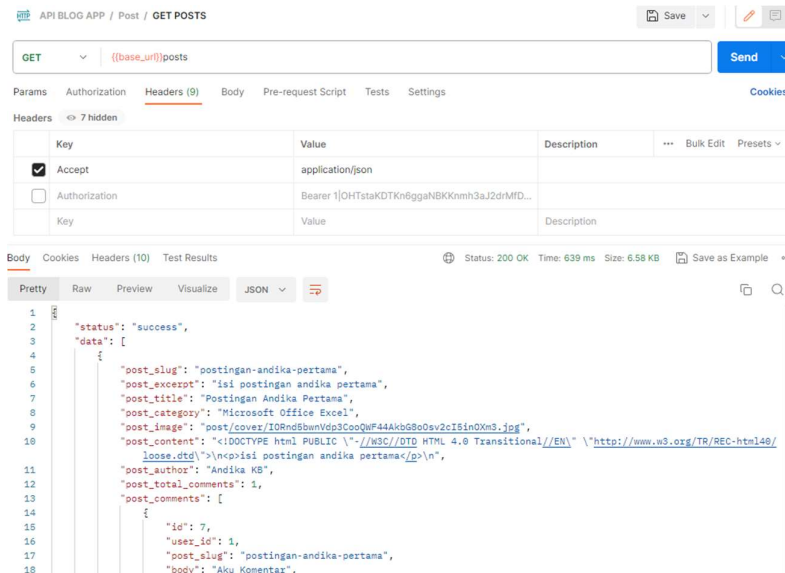
```
1 class PostResource extends JsonResource
2 {
3     /**
4      * Transform the resource collection into an array.
5      *
6      * @return array<int|string, mixed>
7      */
8     public function toArray(Request $request): array
9     {
10         // return parent::toArray($request);
11         return [
12             'post_slug' => $this->slug,
13             'post_excerpt' => $this->excerpt,
14             'post_title' => $this->title,
15             'post_category' => $this->whenLoaded('category')->name,
16             'post_image' => $this->image,
17             'post_content' => $this->body,
18             'post_author' => $this->whenLoaded('author')->name,
19             'post_total_comments' => count($this->whenLoaded('comments')),
20             'post_comments' => $this->whenLoaded('comments', function () {
21                 return collect($this->comments)->each(function ($comment) {
22                     return $comment;
23                 });
24             }),
25             'post_status' => $this->status,
26             'post_publish' => $this->publish,
27             'post_published_at' => \Carbon\Carbon::parse($this->published_at)->diffForHumans(),
28         ];
29     }
30 }
```

Controller

Menampilkan seluruh postingan

```
1 class PostController extends Controller
2 {
3     // -----
4     // GET ALL MATERIES
5     // -----
6     public function allPosts()
7     {
8         $posts = Post::latest()->get();
9
10        return response()->json([
11            'status' => 'success',
12            'data' => PostResource::collection($posts),
13        ]);
14    }
```

Menampilkan seluruh postingan yang sudah dibuat oleh admin atau teacher.



The screenshot shows an API client interface for a 'GET /posts' request. The request is a GET method to the endpoint '({base_url})posts'. The response status is 200 OK, with a time of 639 ms and a size of 6.58 KB. The response body is displayed in JSON format, showing a successful status and a collection of post data.

Key	Value	Description
Accept	application/json	
Authorization	Bearer 1jOHTstaKDTKn6ggaNBKKnmh3aJ2drMFD...	

```
1 {
2   "status": "success",
3   "data": [
4     {
5       "post_slug": "postingan-andika-pertama",
6       "post_excerpt": "isi postingan andika pertama",
7       "post_title": "Postingan Andika Pertama",
8       "post_category": "Microsoft Office Excel",
9       "post_image": "post/cover/IDRNdShanWdp3CooQwF44Ak6S8o0sv2cI5inDXn3.jpg",
10      "post_content": "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\" \"http://www.w3.org/TR/REC-html40/loose.dtd\">\n<p>isi postingan andika pertama</p>\n",
11      "post_author": "Andika KB",
12      "post_total_comments": 1,
13      "post_comments": [
14        {
15          "id": 7,
16          "user_id": 1,
17          "post_slug": "postingan-andika-pertama",
18          "body": "Aku Komentar",
19        }
20      ]
21    }
22  ]
23 }
```

Menampilkan Postingan yang Hanya Dibuat Oleh User yang Login

```
1 // -----
2 // GET ALL MATERIES BY AUTHOR
3 // -----
4 public function myPosts(String $parameter)
5 {
6     $user = User::firstWhere('username', $parameter);
7     $posts = Post::latest()->where('author_id', $user->id)->get();
8
9     return response()->json([
10        'status' => 'success',
11        'data' => PostResource::collection($posts),
12    ]);
13 }
```


GET Send

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

Headers 7 hidden

Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Accept	application/json				
<input type="checkbox"/> Authorization	Bearer 1jOHTstaKDTKn6ggaNBKKnmh3aJ2drMfD...				
Key	Value	Description			

Body Cookies Headers (10) Test Results Status: 200 OK Time: 320 ms Size: 5.8 KB Save as Example

Pretty Raw Preview Visualize JSON

```

1 {
2   "status": "success",
3   "data": [
4     {
5       "post_slug": "postingan-ke-sebelas",
6       "post_excerpt": "Postingan Ke SebelasPostingan Ke SebelasPostingan Ke SebelasPostingan Ke Sebelas",
7       "post_title": "postingan ke sebelas",
8       "post_category": "Microsoft Office Excel",
9       "post_image": "post/cover/8zVyFGEZSrj4bE5fcWw1f1e6Z2EONWwF6J1z71p.png",
10      "post_content": "<p>Postingan Ke Sebelas</p><br></p><p>Postingan Ke Sebelas</p><br></p><p>Postingan Ke Sebelas</p><br></p><p>Postingan Ke Sebelas<br></p>",
11      "post_author": "Gilang Aji Panutan",
12      "post_total_comments": 0,
13      "post_comments": [],
14      "post_status": "free",
15      "post_publish": 1,
16      "post_published_at": "2 weeks ago"
17    },
18  ]
19 }
```

Menampilkan Hanya Postingan Gratis yang sudah di publish

```

1 // -----
2 // GET ONLY FREE MATERIES
3 // -----
4 public function freePosts()
5 {
6     $posts = Post::latest()->where('publish', 1)->where('status', 'free')->get();
7
8     return response()->json([
9         'status' => 'success',
10        'data' => PostResource::collection($posts),
11    ]);
12 }
```

GET Send

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

Headers 7 hidden

Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Accept	application/json				
<input type="checkbox"/> Authorization	Bearer 1jOHTstaKDTKn6ggaNBKKnmh3aJ2drMfD...				
Key	Value	Description			

Body Cookies Headers (10) Test Results Status: 200 OK Time: 258 ms Size: 3.05 KB Save as Example

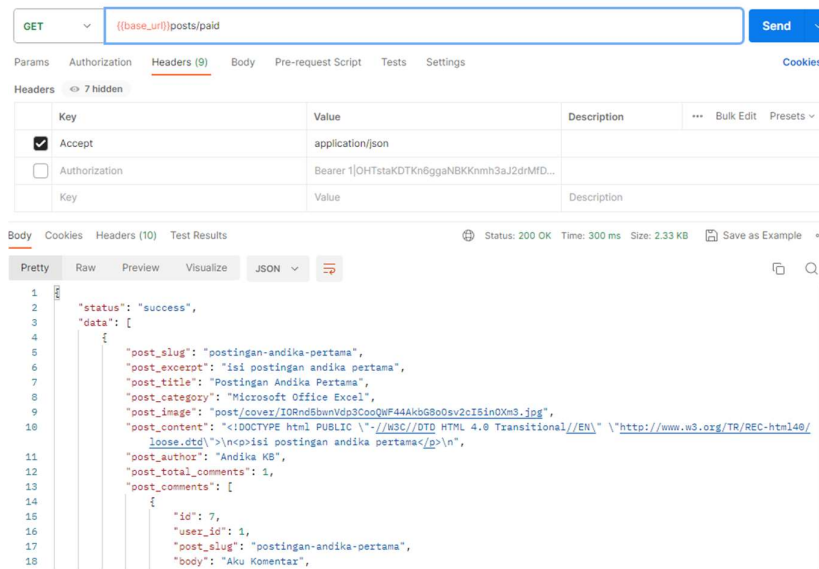
Pretty Raw Preview Visualize JSON

```

1 {
2   "status": "success",
3   "data": [
4     {
5       "post_slug": "postingan-ke-sebelas",
6       "post_excerpt": "Postingan Ke SebelasPostingan Ke SebelasPostingan Ke SebelasPostingan Ke Sebelas",
7       "post_title": "postingan ke sebelas",
8       "post_category": "Microsoft Office Excel",
9       "post_image": "post/cover/8zVyFGEZSrj4bE5fcWw1f1e6Z2EONWwF6J1z71p.png",
10      "post_content": "<p>Postingan Ke Sebelas</p><br></p><p>Postingan Ke Sebelas</p><br></p><p>Postingan Ke Sebelas</p><br></p><p>Postingan Ke Sebelas<br></p>",
11      "post_author": "Gilang Aji Panutan",
12      "post_total_comments": 0,
13      "post_comments": [],
14      "post_status": "free",
15      "post_publish": 1,
16      "post_published_at": "2 weeks ago"
17    },
18  ]
19 }
```

Menampilkan Hanya Postingan Berbayar yang sudah di publish

```
1 // -----
2 // GET ONLY PAID MATERIES
3 // -----
4 public function paidPosts()
5 {
6     $posts = Post::latest()->where('publish', 1)->where('status', 'paid')->get();
7
8     return response()->json([
9         'status' => 'success',
10        'data' => PostResource::collection($posts),
11    ]);
12 }
```



Menampilkan Hanya Postingan yang Sudah di Publish

```
1 // -----
2 // GET ONLY PUBLISHED MATERIES
3 // -----
4 public function publishedPosts()
5 {
6     $posts = Post::latest()->where('publish', 1)->get();
7
8     return response()->json([
9         'status' => 'success',
10        'data' => PostResource::collection($posts),
11    ]);
12 }
13
```

GET `{{base_url}}posts/published` Send

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

Headers 7 hidden

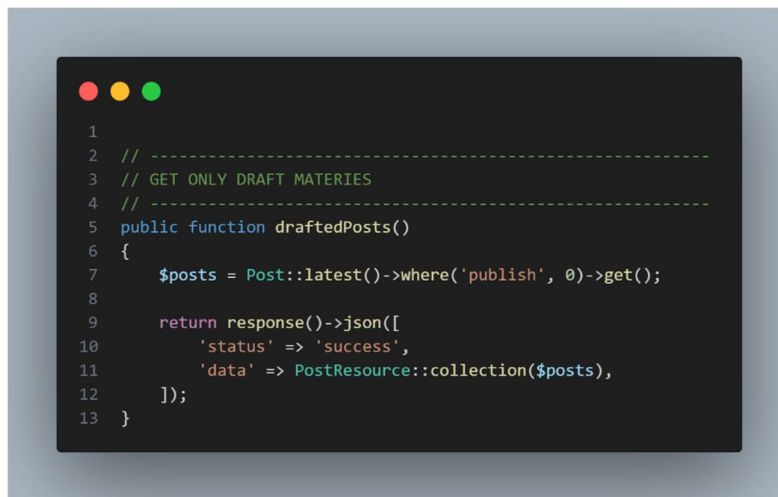
Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/> Accept	application/json			
<input type="checkbox"/> Authorization	Bearer 1 OHTstaKDTKn6ggaNBKKnmh3aJ2drMFD...			
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 344 ms Size: 5.05 KB Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "data": [
4     {
5       "post_slug": "postingan-andika-pertama",
6       "post_excerpt": "isi postingan andika pertama",
7       "post_title": "Postingan Andika Pertama",
8       "post_category": "Microsoft Office Excel",
9       "post_image": "post/cover/10md8bmVd3CooQwF44Ak6G8o0sv2cI5in0xm3.jpg",
10      "post_content": "<DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\" \"http://www.w3.org/TR/REC-html40/loose.dtd\">\n<nc>isi postingan andika pertama<p>\n",
11      "post_author": "Andika KB",
12      "post_total_comments": 1,
13      "post_comments": [
14        {
15          "id": 7,
16          "user_id": 1,
17          "post_slug": "postingan-andika-pertama",
18          "body": "Aku komentaz",
```

Menampilkan Hanya Postingan yang Masih di Draft



GET `{{base_url}}posts/drafted` Send

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

Headers 7 hidden

Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/> Accept	application/json			
<input type="checkbox"/> Authorization	Bearer 1 OHTstaKDTKn6ggaNBKKnmh3aJ2drMFD...			
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 294 ms Size: 1.85 KB Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "data": [
4     {
5       "post_slug": "postingan-kedelapan",
6       "post_excerpt": "Konten postingan ke delapan",
7       "post_title": "Postingan Kedelapan",
8       "post_category": "Microsoft Office Excel",
9       "post_image": "post/cover/MdujpmnQCEW3n19shL8656DfyxGQIyIGK9QzVTIX.jpg",
10      "post_content": "<DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\" \"http://www.w3.org/TR/REC-html40/loose.dtd\">\n<nc>Konten postingan ke delapan<p>\n",
11      "post_author": "Gilang Aji Panutan",
12      "post_total_comments": 0,
13      "post_comments": [],
14      "post_status": "psid",
15      "post_publish": 0,
16      "post_published_at": "1 second ago"
17     },
18   ]
```

Menampilkan Detail Postingan

```
1 // -----
2 // GET DETAIL MATERY
3 // -----
4 public function show(Post $post)
5 {
6     $post = Post::firstWhere('slug', $post->slug);
7
8     return response()->json([
9         'status' => 'success',
10        'data' => new PostResource($post),
11    ]);
12 }
```

GET ({{base_url}})posts/postingan-pertama Send

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

Headers <> 7 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Accept	application/json				
<input type="checkbox"/>	Authorization	Bearer 1jOHTstaKDTKn6ggaNBKKnmh3aJ2drMFD...				
	Key	Value	Description			

body Cookies Headers (10) Test Results Status: 200 OK Time: 306 ms Size: 954 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "data": {
4     "post_slug": "postingan-pertama",
5     "post_excerpt": "Ini adalah postingan pertama",
6     "post_title": "Postingan Pertama",
7     "post_category": "Microsoft Office Excel",
8     "post_image": null,
9     "post_content": "Ini adalah isi dari postingan pertama",
10    "post_author": "Gilang Aji Panutan",
11    "post_total_comments": 1,
12    "post_comments": [
13      {
14        "id": 1,
15        "user_id": 1,
16        "post_slug": "postingan-pertama",
17        "body": "Isi dari komentar pertama",
18        "created_at": "2023-05-27T12:06:19.000000Z",
19        "user": {
```

Membuat Postingan Baru

```
1 // -----  
2 // CREATE MATERY  
3 // -----  
4 public function store(Request $request)  
5 {  
6     // -----  
7     // VALIDATION RULES  
8     // -----  
9     $validateData = $request->validate([  
10         'post_slug' => 'required|max:255|unique:posts,slug',  
11         // 'post_excerpt' => 'required',  
12         'post_title' => 'required',  
13         'post_category' => 'required',  
14         'post_image' => 'nullable|image|file|mimes:jpg,jpeg,png|max:512',  
15         'post_content' => 'required',  
16         'post_status' => 'required',  
17         'post_publish' => 'required',  
18     ]);  
19  
20     $validateData['post_author'] = Auth::user()->id;  
21  
22     if($request->post_publish == 1) {  
23         $validateData['post_published_at'] = \Carbon\Carbon::now()->toDateTimeString();  
24     }  
25  
26     if($request->file('post_image')) {  
27         $validateData['post_image'] = $request->file('post_image')->store('post/cover');  
28     }  
29  
30     $storageImageContent = "post/content/".$request->post_slug;  
31     Storage::makeDirectory($storageImageContent);  
32  
33     $dom = new \DOMDocument();  
34     libxml_use_internal_errors(true);  
35     $dom->loadHTML($request->post_content, LIBXML_HTML_NOIMPLIED | LIBXML_HTML_NOIMPLIED);  
36     libxml_clear_errors();  
37     $contentImages = $dom->getElementsByTagName('img');  
38     foreach ($contentImages as $contentImage) {  
39         $src = $contentImage->getAttribute('src');  
40         if(preg_match('/data:image\/(?<mime>.*?)\//', $src, $groups)) {  
41             $mimetype = $groups['mime'];  
42             $fileNameContent = uniqid();  
43             $fileNameContentRandom = substr(md5($fileNameContent), 6, 6).'.'.time();  
44             $filePath = ("storage/".$storageImageContent.'/'.$fileNameContentRandom.$mimetype");  
45             $image = Image::make($src)  
46                 ->encode($mimetype, 100)  
47                 ->save(public_path($filePath));  
48             $new_src = asset($filePath);  
49             $contentImage->removeAttribute('src');  
50             $contentImage->setAttribute('src', $new_src);  
51             $contentImage->setAttribute('class', 'img-responsive');  
52         }  
53     }  
54 }  
55  
56 $validateData['post_excerpt'] = Str::limit(strip_tags($request->post_content), 160);  
57 $validateData['post_content'] = $dom->saveHTML();  
58  
59 // -----  
60 // CREATE POST  
61 // -----  
62 Post::create([  
63     'slug' => $validateData['post_slug'],  
64     'title' => $validateData['post_title'],  
65     'category_id' => $validateData['post_category'],  
66     'image' => ($request->file('post_image')) ? $validateData['post_image'] : null,  
67     'excerpt' => $validateData['post_excerpt'],  
68     'body' => $validateData['post_content'],  
69     'status' => $validateData['post_status'],  
70     'author_id' => $validateData['post_author'],  
71     'publish' => $validateData['post_publish'],  
72     'published_at' => ($request->post_publish == 1) ? $validateData['post_published_at'] : null,  
73 ]);  
74  
75 // -----  
76 // RESPONSE  
77 // -----  
78 return response()->json([  
79     'status' => 'success',  
80     'data' => "New matery has been added!",  
81 ]);  
82 }
```

POST

{{base_url}}posts

Send

Params

Authorization

Headers (12)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	post_slug	postingan-empat-puluh			
<input checked="" type="checkbox"/>	post_title	Postingan Ke Empat Puluh			
<input checked="" type="checkbox"/>	post_category	2			
<input checked="" type="checkbox"/>	post_image	Screenshot (2).png			
<input checked="" type="checkbox"/>	post_content	Isi Postingan Ke Empat Puluh			
<input checked="" type="checkbox"/>	post_status	free			
<input checked="" type="checkbox"/>	post_publish	1			
<input checked="" type="checkbox"/>	_method	POST			
	Key	Value	Description		

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Time: 1192 ms

Size: 364 B

Save as Example

...

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

START

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

Update Postingan

```
1 // -----
2 // UPDATE MATERY
3 // -----
4 public function update(Request $request, Post $post) {
5
6     // return $request->all();
7
8     // -----
9     // VALIDATION RULES
10    // -----
11    $rules = [
12        // 'post_slug' => 'required|max:255|unique:posts,slug',
13        // 'post_excerpt' => 'required',
14        'post_title' => 'required',
15        'post_category' => 'required',
16        'post_image' => 'nullable|image|file|mimes:jpg,jpeg,png|max:512',
17        'post_content' => 'required',
18        'post_status' => 'required',
19        'post_publish' => 'required',
20    ];
21
22    if($request->post_slug != $post->slug) {
23        $rules['post_slug'] = 'required|max:255|unique:posts,slug';
24    }
25
26    $validateData = $request->validate($rules);
27
28    if($request->file('post_image')) {
29        if($request->old_post_image) {
30            Storage::delete($request->old_post_image);
31        }
32        $validateData['post_image'] = $request->file('post_image')->store('post/cover');
33    }
34
35    $validateData['post_author'] = Auth::user()->id;
36
37    $validateData['post_excerpt'] = Str::limit(strip_tags($request->post_content), 160);
38    // $validateData['post_content'] = $dom->saveHTML();
39
40    if($request->post_publish == 1 && $request->old_post_publish == 0) {
41        $validateData['post_published_at'] = \Carbon\Carbon::now()->toDateTimeString();
42    }
43
44    // -----
45    // UPDATE RULES
46    // -----
47    $updateRules = [
48        'title' => $validateData['post_title'],
49        'category_id' => $validateData['post_category'],
50        'excerpt' => $validateData['post_excerpt'],
51        'body' => $validateData['post_content'],
52        'status' => $validateData['post_status'],
53        'author_id' => $validateData['post_author'],
54        'publish' => $validateData['post_publish'],
55    ];
56
57    if($request->post_slug != $post->slug) {
58        $updateRules['slug'] = $validateData['post_slug'];
59
60        Comment::where('post_slug', $post->slug)->update([
61            'post_slug' => $validateData['post_slug'],
62        ]);
63    }
64
65    if($request->file('post_image')) {
66        $updateRules['image'] = $validateData['post_image'];
67    }
68
69    if($request->post_publish == 1 && $request->old_post_publish == 0) {
70        $updateRules['published_at'] = $validateData['post_published_at'];
71    }
72
73    // -----
74    // UPDATE POST
75    // -----
76    Post::where('slug', $post->slug)->update($updateRules);
77
78    // -----
79    // RESPONSE
80    // -----
81    return response()->json([
82        'status' => 'success',
83        'data' => "Materi has been updated!",
84    ]);
85 }
```

POST Send

Params Authorization Headers (12) **Body** Pre-request Script Tests Settings Cookies

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

<input checked="" type="checkbox"/>	post_title	postingan ke sebelas	
<input checked="" type="checkbox"/>	post_slug	postingan-ke-sebelas	
<input checked="" type="checkbox"/>	post_category	1	
<input checked="" type="checkbox"/>	post_image	TTD DEDI DARWIS.png	
<input checked="" type="checkbox"/>	post_content	Postingan Ke Sebelas	
<input checked="" type="checkbox"/>	post_status	free	
<input checked="" type="checkbox"/>	post_publish	1	
<input checked="" type="checkbox"/>	old_post_publish	1	
<input checked="" type="checkbox"/>	old_post_image	post/cover/UxwC5NEREH12ldZXw0K6LLqSpIQJ7...	
	Key	Value	Description

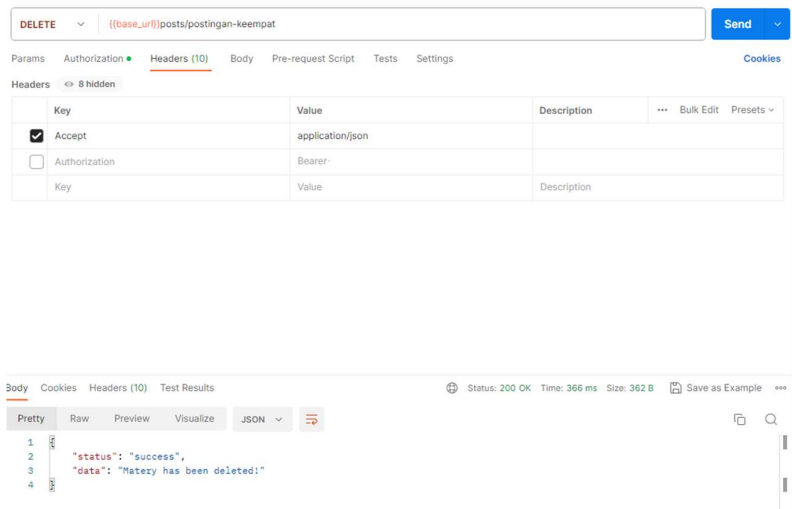
Body Cookies Headers (10) Test Results Status: 200 OK Time: 549 ms Size: 362 B Save as Example

Pretty Raw Preview Visualize JSON

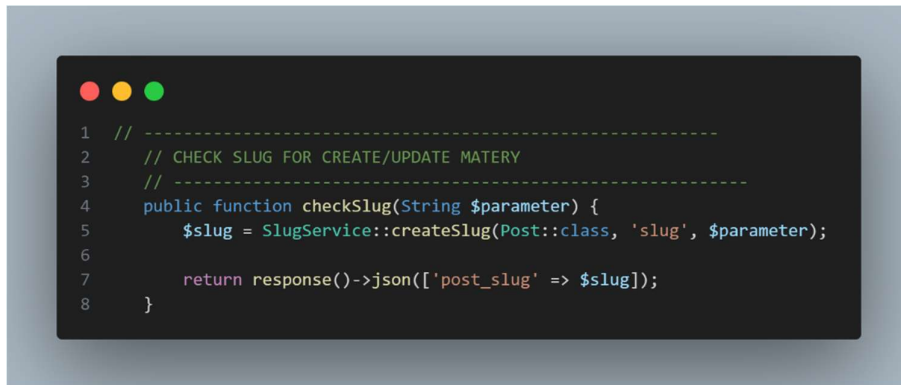
```
1  {
2    "status": "success",
3    "data": "Matezy has been updated!"
4  }
```

Menghapus Postingan

```
1 // -----
2 // DELETE MATERY
3 // -----
4 public function destroy(Post $post) {
5     // -----
6     // DELETE POST IMAGE IF EXISTS
7     // -----
8     if($post->image) {
9         Storage::delete($post->image);
10    }
11
12    // -----
13    // DELETE FOLDER POST
14    // -----
15    $storageImageContent = "post/content/" . $post->slug;
16    Storage::deleteDirectory($storageImageContent);
17
18    // -----
19    // DELETE POST
20    // -----
21    Post::destroy($post->id);
22
23    // -----
24    // RESPONSE
25    // -----
26    return response()->json([
27        'status' => 'success',
28        'data' => "Matezy has been deleted!",
29    ]);
30 }
```

Pembuatan Slug Otomatis untuk Tambah dan Update Postingan



Kategori Model

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Cviebrock\EloquentSluggable\Sluggable;
8
9  class Category extends Model
10 {
11     use HasFactory, Sluggable;
12
13     protected $guarded = ['id'];
14
15     public function getRouteKeyName(): string {
16         return 'slug';
17     }
18
19     public function posts() {
20         return $this->hasMany(Post::class, 'category_id', 'id');
21     }
22
23     public function sluggable(): array
24     {
25         return [
26             'slug' => [
27                 'source' => 'category_name'
28             ]
29         ];
30     }
31 }
```

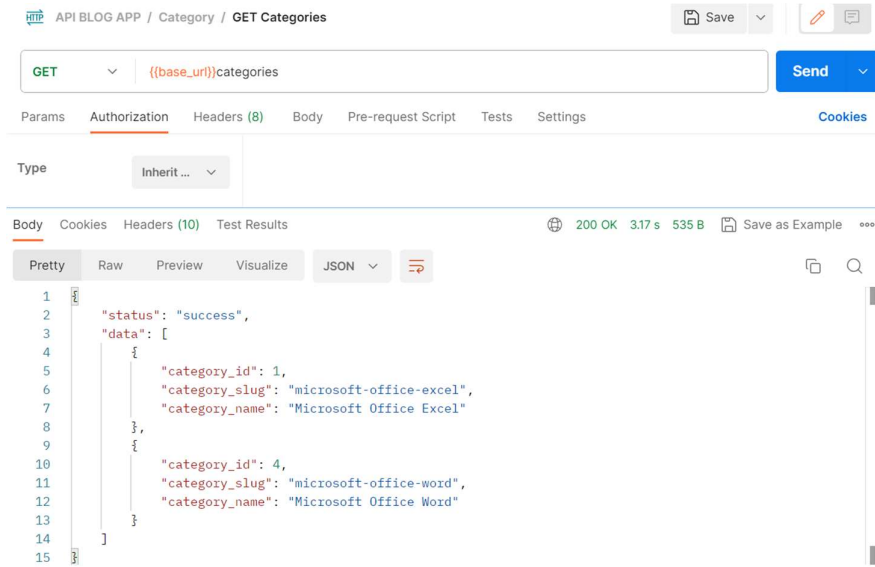
Resource

```
1 class CategoryResource extends JsonResource
2 {
3     /**
4      * Transform the resource collection into an array.
5      *
6      * @return array<int|string, mixed>
7      */
8     public function toArray(Request $request): array
9     {
10         // return parent::toArray($request);
11         return [
12             'category_id' => $this->id,
13             'category_slug' => $this->slug,
14             'category_name' => $this->name,
15         ];
16     }
17 }
```

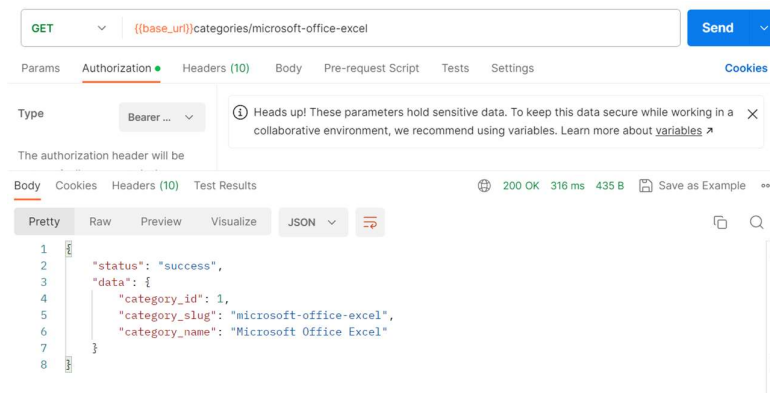
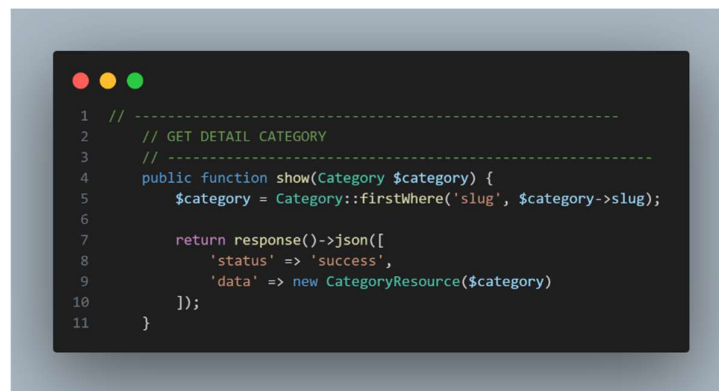
Controller

Mendapatkan semua kategori

```
1 class CategoryController extends Controller
2 {
3     // -----
4     // GET CATEGORIES
5     // -----
6     public function index() {
7         $categories = Category::all();
8
9         return response()->json([
10             'status' => 'success',
11             'data' => CategoryResource::collection($categories),
12         ]);
13     }
```

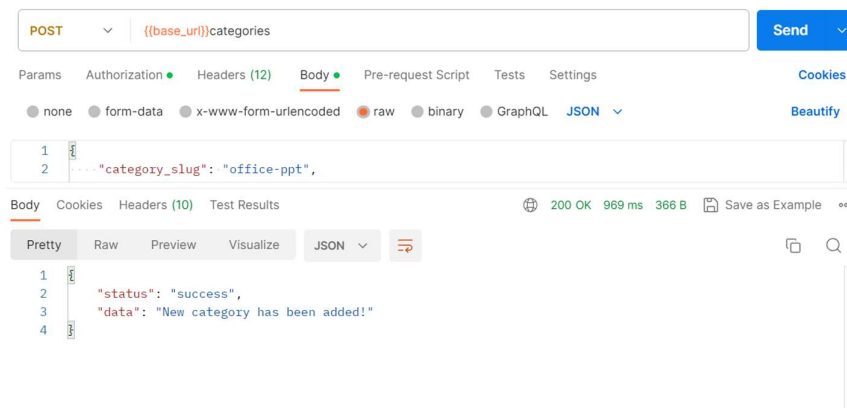


Mendapatkan detail kategori



Tambah Kategori

```
1 // -----
2 // CREATE CATEGORY
3 // -----
4 public function store(Request $request) {
5 // -----
6 // VALIDATION RULES
7 // -----
8 $validateData = $request->validate([
9     'category_slug' => 'required|max:255|unique:categories,slug',
10    'category_name' => 'required',
11 ]);
12
13 // -----
14 // CREATE CATEGORY
15 // -----
16 Category::create([
17     'slug' => $validateData['category_slug'],
18     'name' => $validateData['category_name'],
19 ]);
20
21 // -----
22 // RESPONSE
23 // -----
24 return response()->json([
25     'status' => 'success',
26     'data' => "New category has been added!",
27 ]);
28 }
```



Update Kategori

```
1 // -----
2 // UPDATE CATEGORY
3 // -----
4 public function update(Request $request, Category $category) {
5 // -----
6 // VALIDATION RULES
7 // -----
8 $rules = [
9     'category_name' => 'required',
10 ];
11
12 if($request->category_slug != $category->slug) {
13     $rules['category_slug'] = 'required|max:255|unique:categories,slug';
14 }
15
16 $validateData = $request->validate($rules);
17
18 // -----
19 // UPDATE RULES
20 // -----
21 $updateRules = [
22     'name' => $validateData['category_name'],
23 ];
24
25 if($request->category_slug != $category->slug) {
26     $updateRules['slug'] = $validateData['category_slug'];
27 }
28
29 // -----
30 // UPDATE CATEGORY
31 // -----
32 Category::where('slug', $category->slug)->update($updateRules);
33
34 // -----
35 // RESPONSE
36 // -----
37 return response()->json([
38     'status' => 'success',
39     'data' => "Category has been updated!",
40 ]);
41 }
```

PUT [Send](#)

Params Authorization Headers (12) **Body** Pre-request Script Tests Settings [Cookies](#)

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#) [Beautify](#)

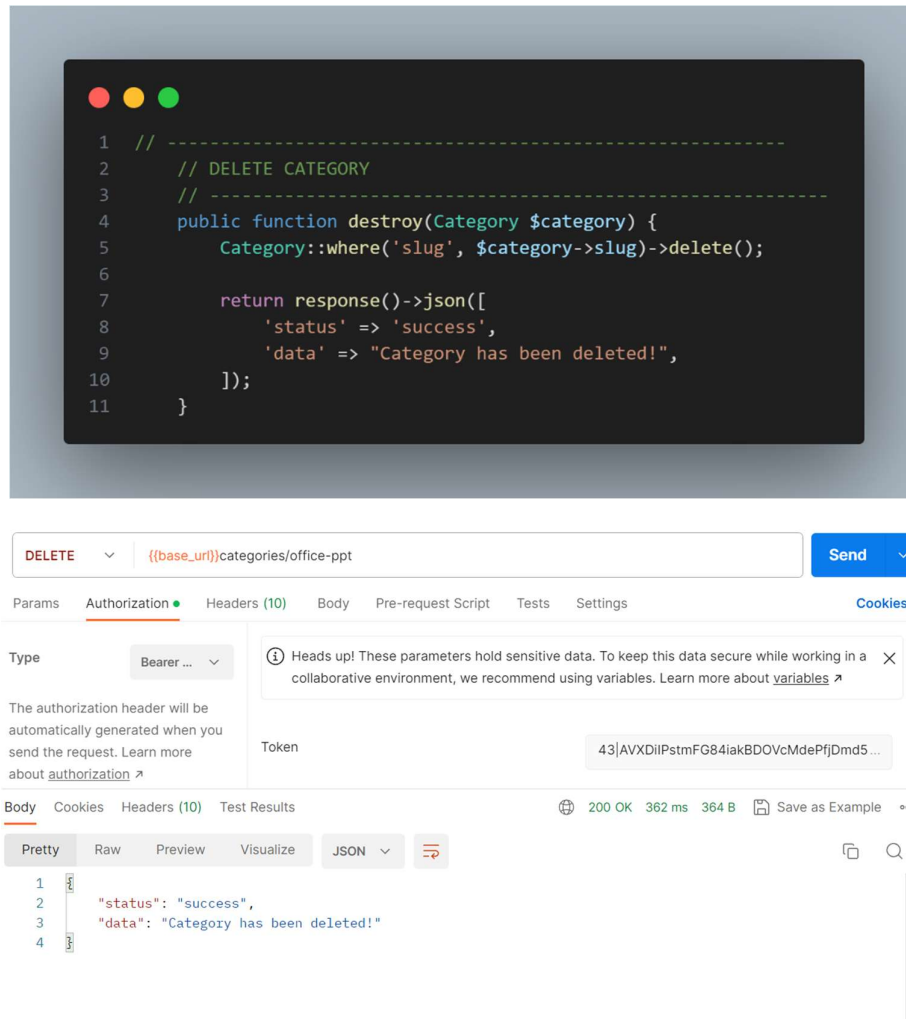
```
1 {
2   "category_slug": "office-ppt",
3   "category_name": "Microsoft Office Power Point"
4 }
```

Body Cookies Headers (10) Test Results 200 OK 381 ms 364 B [Save as Example](#)

Pretty Raw Preview Visualize [JSON](#) [Copy](#)

```
1 {
2   "status": "success",
3   "data": "Category has been updated!"
4 }
```

Hapus Kategori



The image shows a code editor with a PHP function `destroy` and an API client interface. The code defines a function that deletes a category by slug and returns a success message. The API client shows a DELETE request to `{{base_url}}categories/office-ppt` with a Bearer token. The response is a JSON object indicating success.

```
1 // -----  
2 // DELETE CATEGORY  
3 // -----  
4 public function destroy(Category $category) {  
5     Category::where('slug', $category->slug)->delete();  
6  
7     return response()->json([  
8         'status' => 'success',  
9         'data' => "Category has been deleted!",  
10    ]);  
11 }
```

DELETE `{{base_url}}categories/office-ppt` Send

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

Type Bearer ...

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)

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

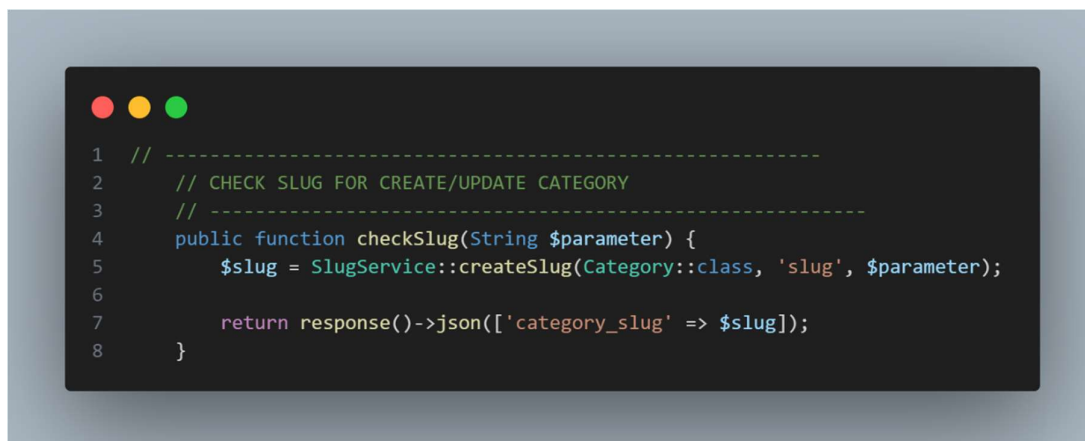
Token `43|AVXDIIpstmFG84iakBDOVcMdePfjDmd5...`

Body Cookies Headers (10) Test Results 200 OK 362 ms 364 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "status": "success",  
3   "data": "Category has been deleted!"  
4 }
```

Pembuatan Slug Otomatis untuk Tambah dan Update Kategori



The image shows a code editor with a PHP function `checkSlug`. The function takes a parameter and generates a slug using the `SlugService::createSlug` method. It returns the slug in a JSON response.

```
1 // -----  
2 // CHECK SLUG FOR CREATE/UPDATE CATEGORY  
3 // -----  
4 public function checkSlug(String $parameter) {  
5     $slug = SlugService::createSlug(Category::class, 'slug', $parameter);  
6  
7     return response()->json(['category_slug' => $slug]);  
8 }
```

Komentar Model

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8
9  class Comment extends Model
10 {
11     use HasFactory;
12
13     protected $guarded = ['id'];
14
15     protected $with = ['user:id,name,image'];
16
17     public function user(): BelongsTo {
18         return $this->belongsTo(User::class, 'user_id', 'id');
19     }
20
21     // public function post(): BelongsTo {
22     //     return $this->belongsTo(Post::class, 'post_slug', 'slug');
23     // }
24 }
25
```

Resource

```
1  <?php
2
3  namespace App\Http\Resources;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Http\Resources\Json\JsonResource;
7
8  class CommentResource extends JsonResource
9  {
10     /**
11      * Transform the resource collection into an array.
12      *
13      * @return array<int|string, mixed>
14      */
15     public function toArray(Request $request): array
16     {
17         // return parent::toArray($request);
18         return [
19             'id' => $this->id,
20             'comment_content' => $this->body,
21             'user_comment' => $this->whenLoaded('user')->name,
22             'post_comment' => $this->whenLoaded('post')->slug,
23             'comment_at' => $this->created_at->diffForHumans(),
24         ];
25     }
26 }
```


Controller

Membuat komentar

```
1 class CommentController extends Controller
2 {
3     // -----
4     // CREATE COMMENT
5     // -----
6     public function store(Request $request) {
7         $validateData = $request->validate([
8             'post_slug' => 'required|exists:posts,slug',
9             'comment_content' => 'required',
10        ]);
11
12        $validateData['user_id'] = Auth::user()->id;
13
14        Comment::create([
15            'user_id' => $validateData['user_id'],
16            'post_slug' => $validateData['post_slug'],
17            'body' => $validateData['comment_content'],
18        ]);
19
20        // RESPONSE
21        return response()->json([
22            'status' => 'success',
23            'data' => "New comment has been added!",
24        ]);
25    }
```

POST Send

Params Authorization Headers (12) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 [data]
2 {
3     "post_slug": "postingan-ketiga",
4     "comment_content": "Ini Komentar di postinga kedua User AndikaA"
5 }
```

Body Cookies Headers (10) Test Results 200 OK 365 ms 365 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 [data]
2 {
3     "status": "success",
4     "data": "New comment has been added!"
5 }
```

Update Komentar

```
1 public function update(Request $request, Comment $comment) {
2     if($comment->user_id != Auth::user()->id) {
3         return response()->json([
4             'status' => 'failed',
5             'data' => "You're not able to update this comment!",
6         ]);
7     }
8
9     $validateData = $request->validate([
10         'post_slug' => 'required|exists:posts,slug',
11         'comment_content' => 'required',
12     ]);
13
14     Comment::where('id', $comment->id)->update([
15         'post_slug' => $validateData['post_slug'],
16         'body' => $validateData['comment_content'],
17     ]);
18
19     // RESPONSE
20     return response()->json([
21         'status' => 'success',
22         'data' => "Your comment has been updated!",
23     ]);
24 }
```

PUT `{{base_url}}comment/1` [Send](#)

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings [Cookies](#)

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#) [Beautify](#)

```
1 {
2   "post_slug": "postingan-ketiga",
3   "comment_content": "Komentar Update"
4 }
```

Body Cookies Headers (10) Test Results [200 OK](#) [362 ms](#) [368 B](#) [Save as Example](#) [...](#)

Pretty Raw Preview Visualize [JSON](#) [...](#)

```
1 {
2   "status": "success",
3   "data": "Your comment has been updated!"
4 }
```