

Application Programming Interface

Pertemuan 12

Gelar Aditya Pratama, S.T., M.Kom.
gelar@stmik-amikbandung.ac.id



laravel api & mysql – **GET, POST, PUT, DELETE**

GET



ProductController.php U X

app > Http > Controllers > Api > ProductController.php > ProductController > index

```
17 public function index(Request $request)
18 {
19     // parameter
20     $where = $request->has('where') ? $request->get('where') : '{}';
21     $sort = $request->has('sort') ? $request->get('sort') : 'id:asc';
22     $per_page = $request->has('per_page') ? $request->get('per_page') : 2;
23     $page = $request->has('page') ? $request->get('page') : 1;
24     $count = $request->has('count') ? $request->get('count') : false;
25     $search = $request->has('search') ? $request->get('search') : '';
26
27     // prepare parameter
28     $sort = explode(':', $sort);
29     $where = str_replace("'", '"', $where);
30     $where = json_decode($where, true);
31 }
```

GET



ProductController.php U X

app > Http > Controllers > Api > ProductController.php > ProductController > index

```
32 .....//query.get
33 ..... $query = Product::where([[ 'id', '>', '0' ]]);
34
35 .....//query.where
36 ..... if($where){
37 .....     foreach($where as $key => $value){
38 .....         $query = $query->where([[ $key, '=', $value ]]);
39 .....     }
40 ..... }
41
42 .....//query.search
43 ..... if($search){
44 .....     $query = $query->where([[ 'name', 'like', '%'. $search . '%' ]]);
45 .....     $query = $query->orWhere([[ 'description', 'like', '%'. $search . '%' ]]);
46 .....     $query = $query->orWhere([[ 'price', 'like', '%'. $search . '%' ]]);
47 ..... }
48
```

GET



```
ProductController.php U X
app > Http > Controllers > Api > ProductController.php > ProductController > index
48
49 .....//variable data
50 ..... $datas = [];
51
52 .....//pagination
53 ..... $pagination = [];
54 ..... $pagination['page'] = (int)$page;
55 ..... $pagination['per_page'] = (int)$per_page;
56 ..... $pagination['total_data'] = $query->count('id');
57 ..... $pagination['total_page'] = ceil($pagination['total_data'] / $pagination['per_page']);
58
59 .....//count
60 ..... if($count == true)
61 ..... {
62 .....     $query = $query->count('id');
63 .....     $datas['count'] = $query;
64 ..... }
```

GET



ProductController.php U X

app > Http > Controllers > Api > ProductController.php > ProductController > index

```
65 .....//get data
66 .....else
67 .....{
68 .....    $query = $query
69 .....    .....->orderBy($sort[0], $sort[1])
70 .....    .....->limit($per_page)
71 .....    .....->offset(($page - 1) * $per_page)
72 .....    .....->get()
73 .....    .....->toArray();
74 .....
75 .....    foreach($query as $qry){
76 .....        $temp = $qry;
77 .....
78 .....        $created_at_indo = \Carbon\Carbon::parse($temp['created_at']);
79 .....        $created_at_indo->locale('id')->settings(['formatFunction'=>'translatedFormat']);
80 .....
81 .....        $temp['created_date_indo'] = $created_at_indo->format('l, d-F-Y H:i:s');
82 .....        array_push($datas, $temp);
83 .....    };
84 .....}
85 .....
86 .....return new ProductResource(true, 'Get Data Successfull', $datas, $pagination);
87 .....}
```

GET by Id



```
ProductController.php U X
app > Http > Controllers > Api > ProductController.php > ProductController > index
88
89     ....public function show($id)
90     ....{
91     ....    ....// query get by id
92     ....    ....$query = Product::find($id);
93
94     ....    ....// variable data
95     ....    ....$datas = $query;
96
97     ....    ....// pagination
98     ....    ....$pagination = [];
99
100    ....    ....return new ProductResource(true, 'Get Data By Id Successfull', $datas, $pagination);
101    ....}
102
```

POST



```
ProductController.php U X
app > Http > Controllers > Api > ProductController.php > ProductController > index
102
103     ...public function store(Request $request)
104     ...{
105     ...    ...// query insert
106     ...    ...$query = Product::create($request->all());
107
108     ...    ...// variable data
109     ...    ...$datas = $query;
110
111     ...    ...// pagination
112     ...    ...$pagination = [];
113
114     ...    ...return new ProductResource(true, 'Insert Data Successfull', $datas, $pagination);
115     ...}
```


PUT by Id



ProductController.php U X

app > Http > Controllers > Api > ProductController.php > ProductController > index

```
116
117     ...public function update(Request $request, $id)
118     ...{
119     ...    ...// query insert
120     ...    ...$query = Product::findOrFail($id);
121     ...    ...$query = $query->update($request->all());
122
123     ...    ...// get after update
124     ...    ...$query = Product::findOrFail($id);
125
126     ...    ...// variable data
127     ...    ...$datas = $query;
128
129     ...    ...// pagination
130     ...    ...$pagination = [];
131
132     ...    ...return new ProductResource(true, 'Update Data Successfull', $datas, $pagination);
133     ...}
```

DELETE by Id



```
ProductController.php U X
app > Http > Controllers > Api > ProductController.php > ProductController > index
134
135     ...public function destroy($id)
136     ...{
137     ...    ...// query insert
138     ...    ...$query = Product::findOrFail($id);
139     ...    ...$query = $query->delete();
140
141     ...    ...// variable data
142     ...    ...$datas = [];
143
144     ...    ...// pagination
145     ...    ...$pagination = [];
146
147     ...    ...return new ProductResource(true, 'Delete Data Successfull', $datas, $pagination);
148     ...}
```

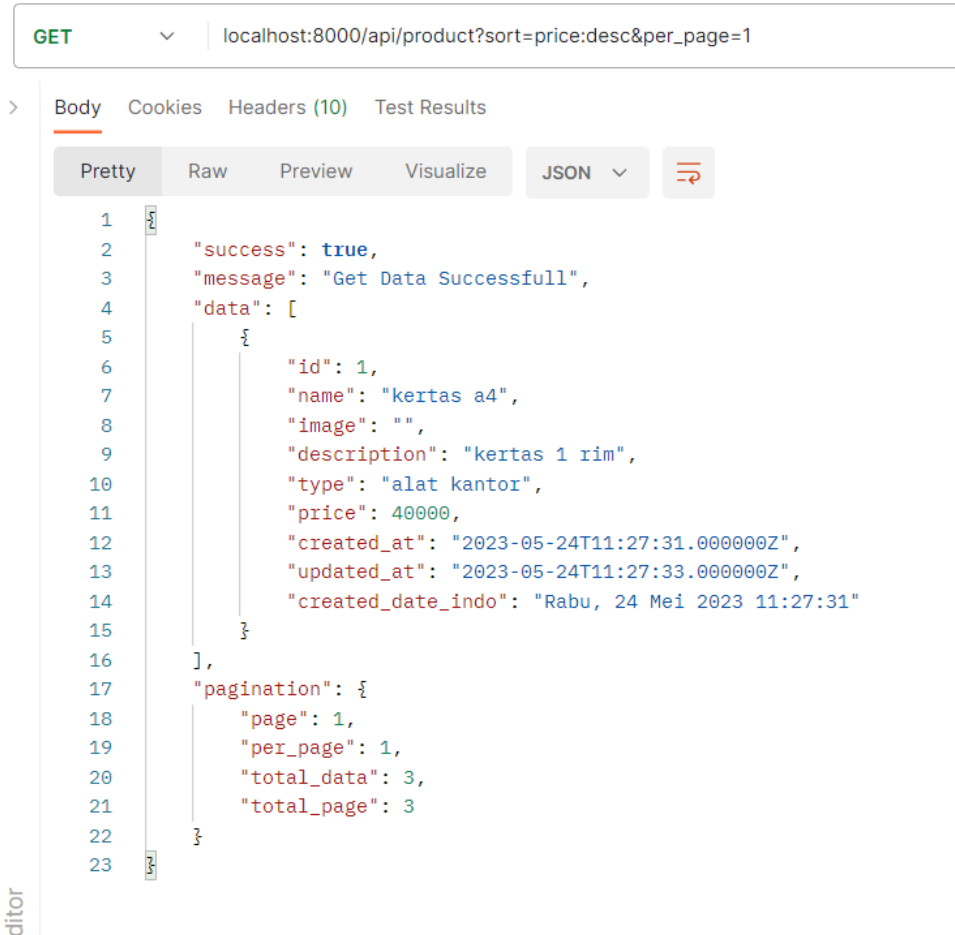
Postman Response

Gunakan method GET

Ketikan url "localhost:8000/api/product"

Dapat diberi parameter

- ?count=true
- ?sort=price:desc
- ?per_page=2&page=1
- ?where={'name':'pensil 2b'}
- ?search=a4



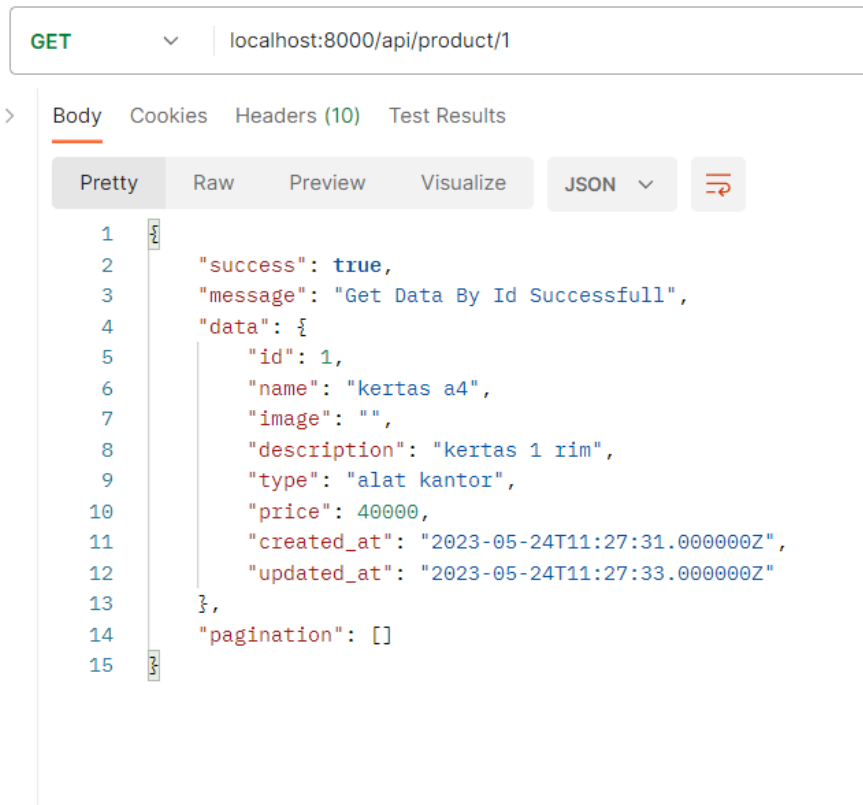
The screenshot shows the Postman interface with a GET request to `localhost:8000/api/product?sort=price:desc&per_page=1`. The response is displayed in the 'Body' tab, showing a JSON object with the following structure:

```
1 {
2   "success": true,
3   "message": "Get Data Successfull",
4   "data": [
5     {
6       "id": 1,
7       "name": "kertas a4",
8       "image": "",
9       "description": "kertas 1 rim",
10      "type": "alat kantor",
11      "price": 40000,
12      "created_at": "2023-05-24T11:27:31.000000Z",
13      "updated_at": "2023-05-24T11:27:33.000000Z",
14      "created_date_indo": "Rabu, 24 Mei 2023 11:27:31"
15    }
16  ],
17  "pagination": {
18    "page": 1,
19    "per_page": 1,
20    "total_data": 3,
21    "total_page": 3
22  }
23 }
```

Postman Response

Gunakan method GET

Ketikan url "localhost:8000/api/product/1"



Postman Response



Gunakan method POST

Ketikan url "localhost:8000/api/product"

Isi Body -> Raw -> Json

The image shows the Postman application interface. At the top, the request method is set to "POST" and the URL is "localhost:8000/api/product". Below the URL bar, there are tabs for "Params", "Auth", "Headers (9)", "Body", "Pre-req.", "Tests", and "Settings". The "Body" tab is selected. On the left side of the "Body" tab, there is a dropdown menu showing "raw" and "JSON", with "JSON" selected. A "Beautify" button is also present. The main area on the left displays the raw JSON body of the request:

```
1 {
2   "name": "spidol",
3   "image": "http://google.com",
4   "description": "12 warna",
5   "type": "alat kantor",
6   "price": 120000
7 }
```

On the right side, there are tabs for "Body", "Cookies", "Headers (10)", and "Test Results". The "Body" tab is selected. Below these tabs are buttons for "Pretty", "Raw", "Preview", and "Visualize", followed by a "JSON" dropdown and a "Beautify" button. The main area on the right displays the formatted JSON response:

```
1 {
2   "success": true,
3   "message": "Insert Data Successfull",
4   "data": {
5     "name": "spidol",
6     "image": "http://google.com",
7     "description": "12 warna",
8     "type": "alat kantor",
9     "price": 120000,
10    "updated_at": "2023-05-30T18:32:21.000000Z",
11    "created_at": "2023-05-30T18:32:21.000000Z",
12    "id": 6
13  },
14   "pagination": []
15 }
```

Postman Response



Gunakan method PUT

Ketikan url "localhost:8000/api/product/6"

Isi Body -> Raw -> Json

The screenshot displays the Postman interface for a PUT request. The top bar shows the method 'PUT' and the URL 'localhost:8000/api/product/6'. Below the bar, the 'Body' tab is selected, and the 'JSON' format is chosen. The request body is a JSON object with the following fields: 'name' (spidol 2), 'image' (http://google2.com), 'description' (12 warna 2), 'type' (alat kantor 2), and 'price' (120002). The right-hand pane shows the response in the 'Pretty' format, which is a JSON object with 'success' (true), 'message' (Update Data Successfull), 'data' (an object with the same fields as the request), and 'pagination' (an empty array).

```
PUT localhost:8000/api/product/6
```

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "name": "spidol 2",
3   "image": "http://google2.com",
4   "description": "12 warna 2",
5   "type": "alat kantor 2",
6   "price": 120002
7 }
```

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Update Data Successfull",
4   "data": {
5     "id": 6,
6     "name": "spidol 2",
7     "image": "http://google2.com",
8     "description": "12 warna 2",
9     "type": "alat kantor 2",
10    "price": 120002,
11    "created_at": "2023-05-30T18:32:21.000000Z",
12    "updated_at": "2023-05-30T18:33:43.000000Z"
13  },
14   "pagination": []
15 }
```

Postman Response

Gunakan method DELETE

Ketikan url "localhost:8000/api/product/6"

