

[illegible]

Kemudia pada VS Code kita masuk ke bagian models. Untuk menambahkan model pada tutorials kita ketik perintah `php artisan make:model tutorials` sehingga akan muncul file baru pada folder models. Kemudian ketikkan codingan seperti gambar berikut

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Tutorials extends Model
9  {
10     use HasFactory;
11     public $table = "tbl_tutorials";
12     protected $fillable = [
13         'id',
14         'title',
15         'description',
16         'published',
17     ];
18 }
19
```

Selanjutnya untuk dapat menampilkan data, tambahkan controller pada folder **api** sebuah file **TutorialsController.php** dengan perintah `php artisan make:controller Api/TutorialsController` sehingga akan muncul file baru pada folder **api**. Kemudian ketikkan codingan seperti gambar berikut :

```
30     });
31     //check if validation fails
32     if ($validator->fails()) {
33         return response()->json($validator->errors(), 422);
34     }
35     //upload image
36     $image = $request->file('image');
37     $image->storeAs('public/tutorials', $image->hashName());
38     //Simpan Data
39     $store = Tutorials::create([
40         'image' => $image->hashName(),
41         'title' => $request->title,
42         'published' => $request->published,
43         'description' => $request->description,
44         'id_user' => Auth::user()->id
45     ]);
46     //return response
47     return new ApiResponse(true, 'Data Tutorials Berhasil Ditambahkan!', $store);
48 }
49
50 //menampilkan Data berdasarkan ID
51 public function show($id)
52 {
53     $data = Tutorials::find($id);
54     if (is_null($data)) {
55         //jika ID tidak ditemukan dalam tabel
```

```

56         return new ApiResource(false, 'Data tidak ditemukan', $data);
57     }
58     return new ApiResource(true, 'Details Data!' , $data);
59 }
60
61 public function update(Request $request, Tutorials $tutorials)
62 {
63     //define validation rules
64     $validator = Validator::make($request->all(), [
65         'title'      => 'required',
66         'description' => 'required',
67     ]);
68
69     //check if validation fails
70     if ($validator->fails()) {
71         return response()->json($validator->errors(), 422);
72     }
73     if ($request->hasFile('image')) {
74         //upload image
75         $image = $request->file('image');
76         $image->storeAs('public/tutorials', $image->hashName());
77         //delete old image
78         Storage::delete('public/tutorials' . $image->hashName());
79
80         // update dengan image baru
81         $tutorials->update([
82             'image'      => $image->hashName(),
83             'title'      => $request->title,
84             'published' => $request->published,
85             'description' => $request->description
86         ]);
87     } else {
88         //update tanpa image
89         $tutorials->update([
90             'title'      => $request->title,
91             'published' => $request->published,
92             'description' => $request->description
93         ]);
94     }
95     return new ApiResource(true, 'Perubahan Data Tutorials Sudah Berasil!', $tutorials);
96 }
97
98 public function destroy(Tutorials $tutorials)
99 {
100     //delete image
101     Storage::delete('public/tutorials/' . $tutorials->image);
102     //Delete data dalam tabel
103     $tutorials->delete();
104     return new ApiResource(true, 'Data telah berhasil di HAPUS !', $tutorials);
105 }
106 }

```

Selanjutnya kita tambahkan routes untuk Tutorials pada folder **routes** didalam file **api.php** dengan codingan seperti pada gambar berikut.

```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\API\AuthController;
5 use Illuminate\Http\Request;
6
7 Route::prefix('v1')->group(function () {
8     Route::prefix('auth')->group(function () {
9         //API route for register new user
10         Route::post('/register', [App\Http\Controllers\API\AuthController::class, 'register']);
11         //API route for login user
12         Route::post('/login', [App\Http\Controllers\API\AuthController::class, 'login']);
13     });
14
15     //Protecting Routes
16     Route::group(['middleware' => ['auth:sanctum']], function () {
17         Route::prefix('user')->group(function () {
18             Route::get('/profile', function(Request $request) {
19                 return auth()->user();
20             });
21             Route::post('/logout', [App\Http\Controllers\API\AuthController::class, 'logout']);
22         });
23         Route::apiResource('/tutorials', App\Http\Controllers\Api\TutorialsController::class);
24         Route::apiResource('/mahasiswa', App\Http\Controllers\Api\MahasiswaController::class);
25     });
26 });
27
```

\

Selanjutnya lakukan juga untuk hal yang sama pada **mahasiswa**. Tambahkan pada folder models sebuah file **Mahasiswa.php** dengan perintah php artisan **make:model Mahasiswa -m** sehingga akan muncul file baru pada folder models. Kemudian ketikkan codingan seperti gambar berikut :

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Mahasiswa extends Model
9  {
10     use HasFactory;
11     public $table = "tbl_mahasiswa";
12     protected $fillable = [
13         'nama',
14         'nim',
15         'prodi',
16         'alamat'
17     ];
18 }
19
```

Selanjutnya untuk dapat menampilkan data, tambahkan controller pada folder api sebuah file **MahasiswaController.php** dengan perintah **php artisan make:controller Api/MahasiswaController** sehingga akan muncul file baru pada folder api. Kemudian ketikkan codingan seperti gambar berikut :

```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4  use App\Models\Mahasiswa;
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Http\Resources\ApiResource;
8  use Auth;
9  use Validator;
10 class MahasiswaController extends Controller
11 {
12     public function index()
13     {
14         //get Mahasiswa
15         $mahasiswa = Mahasiswa::latest()->paginate(5);
16         //return collection of posts as a resource
17         return new ApiResource(true, 'List Data Mahasiswa', $mahasiswa);
18     }
19
20     public function store(Request $request )
21     {
22         //definisi validation rules
23         $validator = Validator::make($request->all(), [
24             'nama' => 'required|string|max:255',
25             'nim' => 'required|string|max:255',
26             'prodi' => 'required|string|max:255',
27             'alamat' => 'required|string|max:255'
28         ]);
29         //check if validation fails
30
31         if ($validator->fails()) {
32             return response()->json($validator->errors(), 422);
33         }
34         //Simpan Data
35         $store = Mahasiswa::create([
36             'nama' => $request->nama,
37             'nim' => $request->nim,
38             'prodi' => $request->prodi,
39             'alamat' => $request->alamat,
40             'id_mahasiswa' => Auth::user()->id
41         ]);
42         //return response
43         return new ApiResource(true, 'Data Mahasiwa Berhasil Ditambahkan!', $store);
44
45         //menampilkan Data berdasarkan ID
46         public function show($id)
47         {
48             $data = Mahasiswa::find($id);
49             if (is_null($data)) {
50                 //Jika ID tidak ditemukan dalam tabel
51                 return new ApiResource(false, 'Data Mahasiswa tidak ditemukan',$data);
52             }
53             return new ApiResource(true, 'Details Data Mahasiswa!' , $data);
54         }
55
56         public function update(Request $request, Mahasiswa $mahasiswa)
```

```

57     {
58         //define validation rules
59         $validator = Validator::make($request->all(), [
60             'nama' => 'required|string|max:255',
61             'nim' => 'required|string|max:255',
62             'prodi' => 'required|string|max:255',
63             'alamat' => 'required|string|max:255'
64         ]);
65
66         //check if validation fails
67         if ($validator->fails()) {
68             return response()->json($validator->errors(), 422);
69         }
70
71         //update tanpa image
72         $mahasiswa->update([
73             'nama' => $request->nama,
74             'nim' => $request->nim,
75             'prodi' => $request->prodi,
76             'alamat' => $request->alamat,
77         ]);
78         return new ApiResource(true, 'Perubahan Data Mahasiswa Sudah Berasil!', []);
79     }
80
81     //Hapus Data
82     public function destroy(Mahasiswa $mahasiswa)
83     {
84         //Delete data dalam tabel
85         $mahasiswa->delete();
86         return new ApiResource(true, 'Data telah berhasil di HAPUS !', []);
87     }
88 }

```

Selanjutnya kita tambahkan routes untuk mahasiswa pada folder **routes** didalam file api.php dengan codingan seperti pada gambar berikut :

```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\API\AuthController;
5 use Illuminate\Http\Request;
6
7 Route::prefix('v1')->group(function () {
8     Route::prefix('auth')->group(function () {
9         //API route for register new user
10         Route::post('/register', [App\Http\Controllers\API\AuthController::class, 'register']);
11         //API route for login user
12         Route::post('/login', [App\Http\Controllers\API\AuthController::class, 'login']);
13     });
14
15     //Protecting Routes
16     Route::group(['middleware' => ['auth:sanctum']], function () {
17         Route::prefix('user')->group(function () {
18             Route::get('/profile', function(Request $request) {
19                 return auth()->user();
20             });
21             Route::post('/logout', [App\Http\Controllers\API\AuthController::class, 'logout']);
22         });
23         Route::apiResource('/tutorials', App\Http\Controllers\Api\TutorialsController::class);
24         Route::apiResource('/mahasiswa', App\Http\Controllers\Api\MahasiswaController::class);
25     });
26 });
27
```

Kemudian dilakukan uji coba untuk CRUD pada tabel **Tutorials** di Postman sebagai berikut :

Register

The screenshot shows a Postman interface for a POST request to `http://localhost:8000/api/v1/auth/register`. The request body is in `form-data` format with the following fields:

Key	Value	Description
name	admin2	
email	admin89@gmail.com	
password	admin123	

The response is a JSON object with the following structure:

```
{
  "success": true,
  "message": "Register Berhasil",
  "data": {
    "User": {
      "name": "admin2",
      "email": "admin89@gmail.com",
      "updated_at": "2024-07-16T14:28:45.000000Z",
      "created_at": "2024-07-16T14:28:45.000000Z",
      "id": 6
    },
    "access_token": "14|RQ8Ut4ofJqfWNVxw0628MVwYIKlxCD8kKgR7xor",
  }
}
```


Login

POST ▼ http://localhost:8000/api/v1/auth/login Send ▼

Params Authorization Headers (8) **Body** • Scripts Settings Cookies

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

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> email	Text ▼ admin89@gmail.com			
<input checked="" type="checkbox"/> password	Text ▼ admin123			
Key	Text ▼ Value	Description		

Body Cookies Headers (10) Test Results 🌐 Status: 200 OK Time: 817 ms Size: 626 B 📄 Save as example ⋮

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```
1 {
2   "success": true,
3   "message": "Hi admin2, welcome to home",
4   "data": {
5     "User": {
6       "id": 6,
7       "name": "admin2",
8       "email": "admin89@gmail.com",
9       "email_verified_at": null,
10      "created_at": "2024-07-16T14:28:45.000000Z",
11      "updated_at": "2024-07-16T14:28:45.000000Z"
12    },
13  },
14 }
```

Create

POST ▼ http://localhost:8000/api/v1/tutorials Send ▼

Params Authorization • Headers (9) **Body** • Scripts Settings Cookies

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

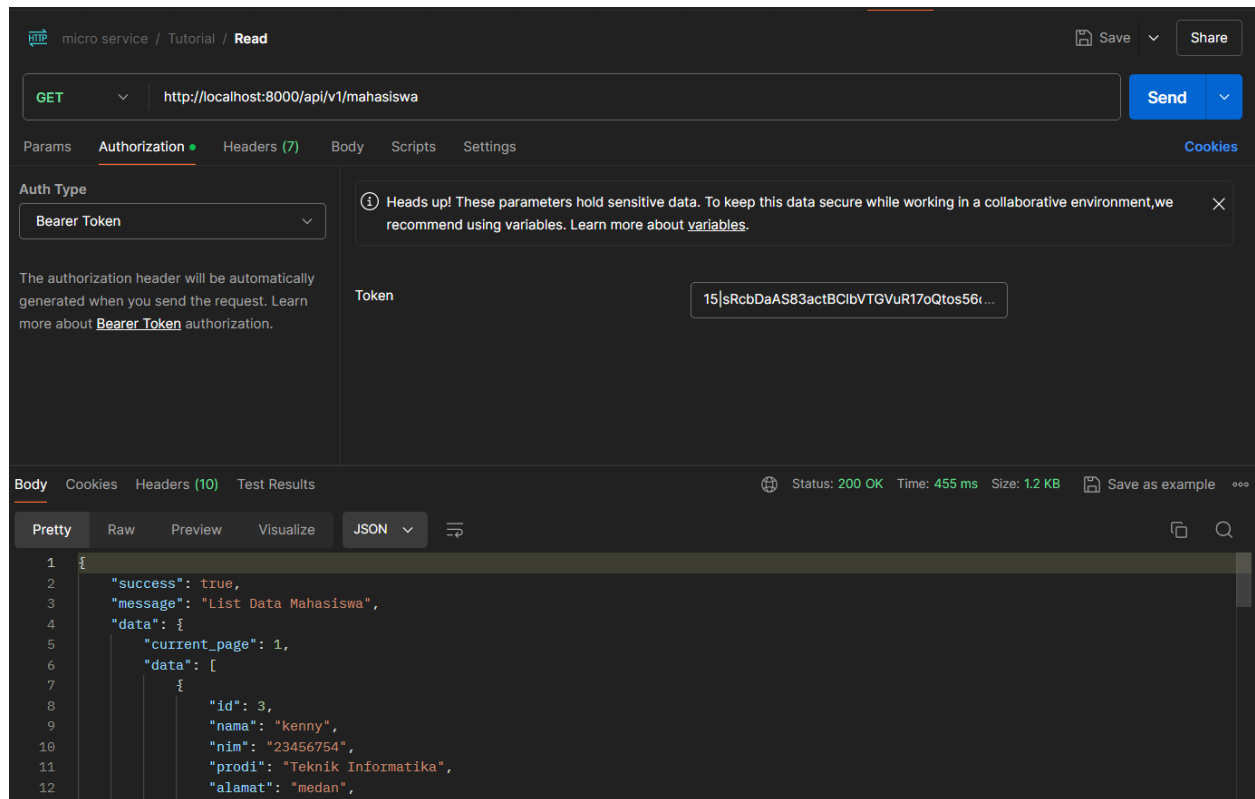
Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> image	File ▼ 📎 OIP.jpg			
<input checked="" type="checkbox"/> title	Text ▼ Coba Create			
<input checked="" type="checkbox"/> published	Text ▼ 1			
<input checked="" type="checkbox"/> description	Text ▼ Percobaan			
Key	Text ▼ Value	Description		

more actions 🌐 Status: 201 Created Time: 478 ms Size: 518 B 📄 Save as example ⋮

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```
1 {
2   "success": true,
3   "message": "Data Tutorials Berhasil Ditambahkan!",
4   "data": {
5     "title": "Coba Create",
6     "published": "1",
7     "updated_at": "2024-07-16T14:31:47.000000Z",
8     "created_at": "2024-07-16T14:31:47.000000Z",
9     "id": 3
10  },
11 }
```

Read



micro service / Tutorial / **Read** Save Share

GET ▼ Send ▼

Params **Authorization** • Headers (7) Body Scripts Settings Cookies

Auth Type

Bearer Token ▼

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

Token

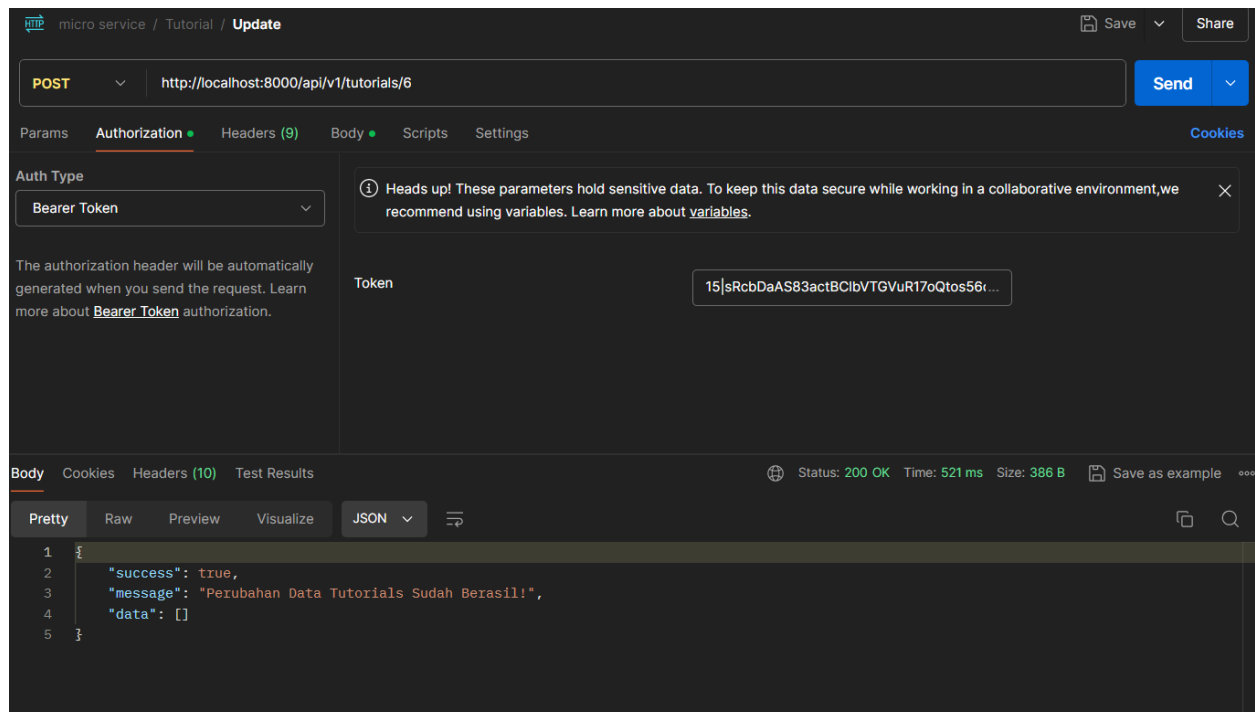
15[sRcbDaAS83actBCIbVTGVuR17oQtos56i...

Body Cookies Headers (10) Test Results Status: 200 OK Time: 455 ms Size: 1.2 KB Save as example ...

Pretty Raw Preview Visualize **JSON** ▼ ≡ 📄 🔍

```
1 {
2   "success": true,
3   "message": "List Data Mahasiswa",
4   "data": {
5     "current_page": 1,
6     "data": [
7       {
8         "id": 3,
9         "nama": "kenney",
10        "nim": "23456754",
11        "prodi": "Teknik Informatika",
12        "alamat": "medan",
```

Update



micro service / Tutorial / **Update** Save Share

POST ▼ Send ▼

Params **Authorization** • Headers (9) **Body** • Scripts Settings Cookies

Auth Type

Bearer Token ▼

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

Token

15[sRcbDaAS83actBCIbVTGVuR17oQtos56i...

Body Cookies Headers (10) Test Results Status: 200 OK Time: 521 ms Size: 386 B Save as example ...

Pretty Raw Preview Visualize **JSON** ▼ ≡ 📄 🔍

```
1 {
2   "success": true,
3   "message": "Perubahan Data Tutorials Sudah Berasil!",
4   "data": []
5 }
```

Delete

micro service / Tutorial / delete

POST http://localhost:8000/api/v1/tutorials/6

Params Authorization Headers (9) Body Scripts Settings Cookies

Auth Type: Bearer Token

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](#).

Token: 15sRcbDaAS83actBCIbVTGVuR17oQtos56r...

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

Body Cookies Headers (10) Test Results

Status: 200 OK Time: 514 ms Size: 377 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Data telah berhasil di HAPUS !",
4   "data": []
5 }
```

Selanjutnya dilakukan uji coba untuk CRUD pada tabel **Mahasiswa** di Postman sebagai berikut :

Register

micro service / Mahasiswa / Register

POST http://localhost:8000/api/v1/auth/register

Params Authorization Headers (8) Body Scripts Settings Cookies

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

Key	Value	Description
name	abqari	
email	fabqari12@gmail.com	
password	12345678	

Body Cookies Headers (10) Test Results

Status: 200 OK Time: 672 ms Size: 594 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Register Berhasil",
4   "data": {
5     "User": {
6       "name": "abqari",
7       "email": "fabqari12@gmail.com",
8       "updated_at": "2024-07-16T14:40:43.000000Z",
9       "created_at": "2024-07-16T14:40:43.000000Z",
10      "id": 7
11    },
12    "access_token": "16|3RhCBkbJUTLNj6szchH5uB8A2bDzKtJXzkWxnAVC"
```

Login

The screenshot shows a REST client interface for a "micro service / Mahasiswa / login" endpoint. The request is a POST to "http://localhost:8000/api/v1/auth/login". The "Authorization" tab is active, showing a "Bearer Token" type. A warning message states: "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 token value is "16j3RhCBkbJUTLNj6szchH5uB0A2bDzKtJ>...". The response is displayed in the "Body" tab, showing a JSON object with a success status, a welcome message, and user details.

micro service / Mahasiswa / login

POST http://localhost:8000/api/v1/auth/login

Params Authorization Headers (9) Body Scripts Settings Cookies

Auth Type: Bearer Token

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

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](#).

Token: 16j3RhCBkbJUTLNj6szchH5uB0A2bDzKtJ>...

Status: 200 OK Time: 697 ms Size: 629 B Save as example

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Hi abqari, welcome to home",
4   "data": {
5     "User": {
6       "id": 5,
7       "name": "abqari",
8       "email": "fabqari006@gmail.com",
9       "email_verified_at": null,
10      "created_at": "2024-06-28T00:59:20.000000Z",
11      "updated_at": "2024-06-28T00:59:20.000000Z"
12    }
13  },
14 }
```

Create

The screenshot shows a REST client interface for a "micro service / Mahasiswa / create" endpoint. The request is a POST to "http://localhost:8000/api/v1/mahasiswa". The "Authorization" tab is active, showing a "Bearer Token" type. A warning message states: "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 token value is "16j3RhCBkbJUTLNj6szchH5uB0A2bDzKtJ>...". The response is displayed in the "Body" tab, showing a JSON object with a success status, a confirmation message, and student details.

micro service / Mahasiswa / create

POST http://localhost:8000/api/v1/mahasiswa

Params Authorization Headers (9) Body Scripts Settings Cookies

Auth Type: Bearer Token

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

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](#).

Token: 16j3RhCBkbJUTLNj6szchH5uB0A2bDzKtJ>...

Status: 201 Created Time: 506 ms Size: 553 B Save as example

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Data Mahasiwa Berhasil Ditambahkan!",
4   "data": {
5     "name": "k",
6     "nim": "23456754",
7     "prodi": "Teknik Informatika",
8     "alamat": "medan",
9     "updated_at": "2024-07-16T14:43:12.000000Z",
10    "created_at": "2024-07-16T14:43:12.000000Z",
11    "id": 4
12  }
13 }
```

Update

POST http://localhost:8000/api/v1/mahasiswa/4 Send

Params Authorization Headers (9) **Body** Scripts Settings Cookies

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

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	nama	Text yoans			
<input checked="" type="checkbox"/>	nim	Text 123453421			
<input checked="" type="checkbox"/>	prodi	Text Teknik Informatika			
<input checked="" type="checkbox"/>	alamat	Text Tarutung			
<input checked="" type="checkbox"/>	_method	Text PUT			
	Key	Text Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 493 ms Size: 386 B Save as example

Pretty Raw Preview Visualize **JSON**

```
1 {
2   "success": true,
3   "message": "Perubahan Data Mahasiswa Sudah Berasil!",
4   "data": []
5 }
```

Delete

micro service / Mahasiswa / delete Save Share

DELETE http://localhost:8000/api/v1/mahasiswa/4 Send

Params Authorization Headers (9) **Body** Scripts Settings Cookies

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


	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	_method	Text DELETE			
	Key	Text Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 338 ms Size: 377 B Save as example

Pretty Raw Preview **Visualize** JSON

```
1 {
2   "success": true,
3   "message": "Data telah berhasil di HAPUS !",
4   "data": []
5 }
```

Logout

 micro service / Mahasiswa / **logout**SaveShare

POST ▼ http://localhost:8000/api/v1/user/logout Send ▼

Params **Authorization** ▼ Headers (8) Body Scripts Settings Cookies

Auth Type

Bearer Token ▼

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

ⓘ 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.](#) ×

Token

16j3RhCBkbJUTLNj6szchH5uB0A2bDzKtJ...

Body Cookies Headers (10) Test ResultsStatus: 200 OK Time: 450 ms Size: 408 B Save as example ⋮

Pretty Raw Preview Visualize JSON ▼ ≡

```
1  {}
2  {
3    "success": true,
4    "message": "You have successfully logged out and the token was successfully deleted"
5  }
```