

TUTORIAL 07 – WEB SERVICE

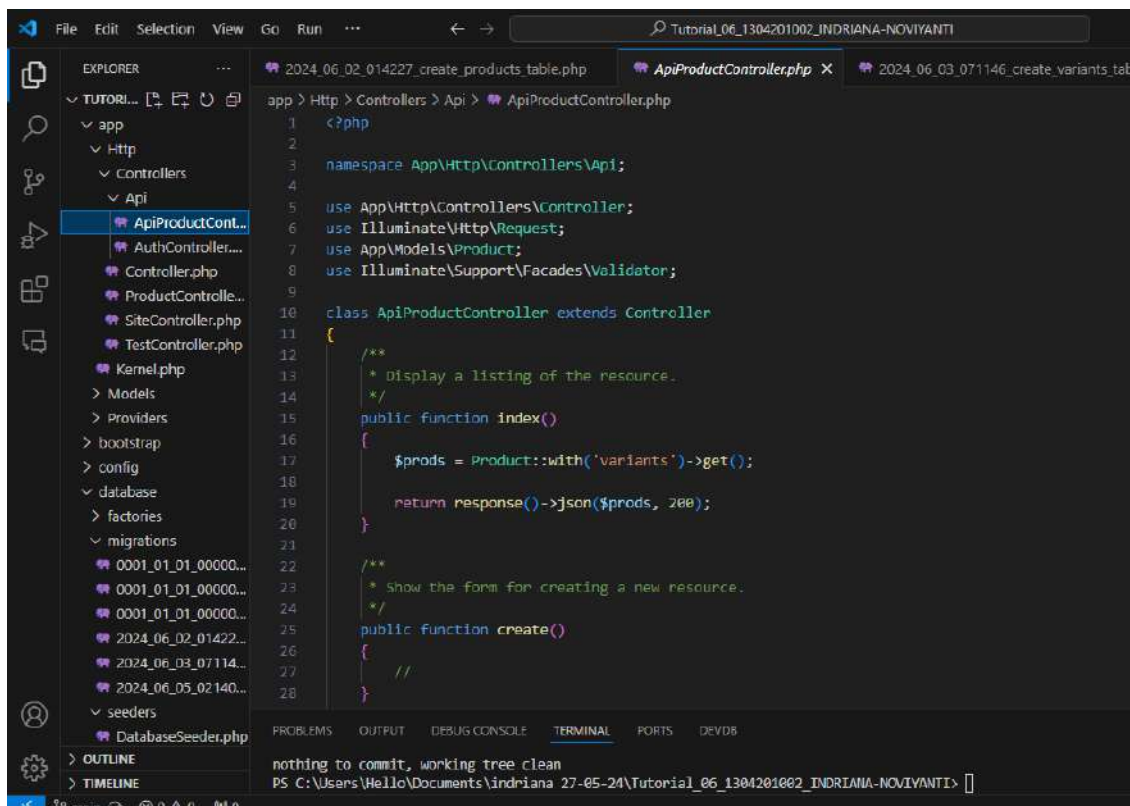
Name : Indriana Noviyanti

NIM. 1304201002

S1 IF Informatika PJJ

Screenshot Code Add Api

ApiProductController.php



```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\Product;
8  use Illuminate\Support\Facades\Validator;
9
10 class ApiProductController extends Controller
11 {
12     /**
13      * Display a listing of the resource.
14      */
15     public function index()
16     {
17         $prods = Product::with('variants')->get();
18
19         return response()->json($prods, 200);
20     }
21
22     /**
23      * Show the form for creating a new resource.
24      */
25     public function create()
26     {
27         //
28     }
```

nothing to commit, working tree clean
PS C:\Users\Hello\Documents\indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI>

The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows a project structure with folders like 'app', 'Http', 'Controllers', 'Api', 'Models', 'Providers', 'bootstrap', 'config', 'database', 'factories', 'migrations', and 'seeder'. The 'ApiProductController.php' file is selected in the 'Api' folder. The main editor window displays the code for the 'store' method. The code includes a validation step using 'Validator::make' and a response step using 'response()->json'. The terminal at the bottom shows 'nothing to commit, working tree clean'.

```
11 {
30 /**
32 */
33 public function store(Request $request)
34 {
35     $validateProduct = Validator::make($request->all(), [
36         'name' => 'required|min:4',
37         'price' => 'required|integer|min:1000000'
38     ]);
39
40     if($validateProduct->fails()) {
41         return response()->json([
42             'status' => false,
43             'message' => "Error validation",
44             'errors' => $validateProduct->errors()
45         ],200);
46     }
47
48     $prod = new Product;
49     $prod->name = $request->name;
50     $prod->price = $request->price;
51     $prod->save();
52
53     return response()->json([
54         'status' => true,
55         'message' => 'Saved successfully'
56     ],201);
57 }
```

nothing to commit, working tree clean
PS C:\Users\Hello\Documents\Indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI>

The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows a project structure with folders like 'app', 'Http', 'Controllers', 'Api', 'Models', 'Providers', 'bootstrap', 'config', 'database', 'factories', 'migrations', and 'seeder'. The 'ApiProductController.php' file is selected in the 'Api' folder. The main editor window displays the code for the 'show', 'edit', and 'update' methods. The code includes a validation step using 'Validator::make' and a response step using 'response()->json'. The terminal at the bottom shows 'nothing to commit, working tree clean'.

```
11 {
57 }
58 /**
59 */
60 * Display the specified resource.
61 */
62 public function show(string $id)
63 {
64     $prod = Product::find($id);
65
66     return response()->json($prod, 200);
67 }
68
69 /**
70 */
71 * Show the form for editing the specified resource.
72 */
73 public function edit(string $id)
74 {
75     //
76 }
77
78 /**
79 */
80 * Update the specified resource in storage.
81 */
82 public function update(Request $request, string $id)
83 {
84     $prod = Product::find($id);
85     $prod->name = $request->name;
86     $prod->price = $request->price;
87     $prod->save();
88
89     return response()->json($prod, 200);
90 }
```

nothing to commit, working tree clean
PS C:\Users\Hello\Documents\Indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI>

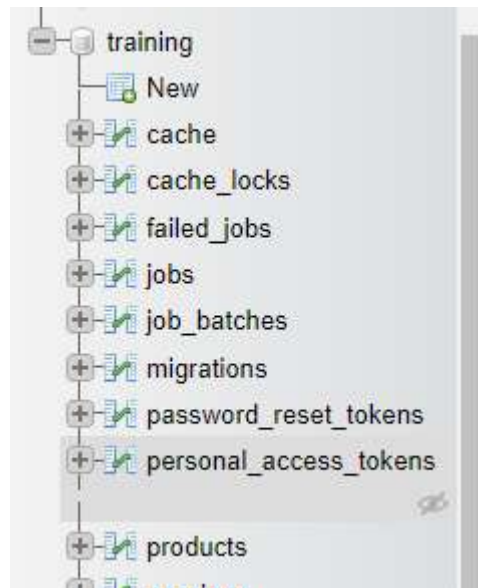
```
11 {
76
77
78     /**
79      * Update the specified resource in storage.
80      */
81     public function update(Request $request, string $id)
82     {
83         $prod = Product::find($id);
84         $prod->name = $request->name;
85         $prod->price = $request->price;
86         $prod->save();
87
88         return response()->json([
89             'status' => true,
90             'message' => 'Edit success'
91         ],201);
92     }
93
94     /**
95      * Remove the specified resource from storage.
96      */
97     public function destroy(string $id)
98     {
99         $isdelete = Product::destroy($id);
100
101         if($isdelete) {
102             return response()->json([
103                 'status' => true,
```

nothing to commit, working tree clean
PS C:\Users\Hello\Documents\indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI>

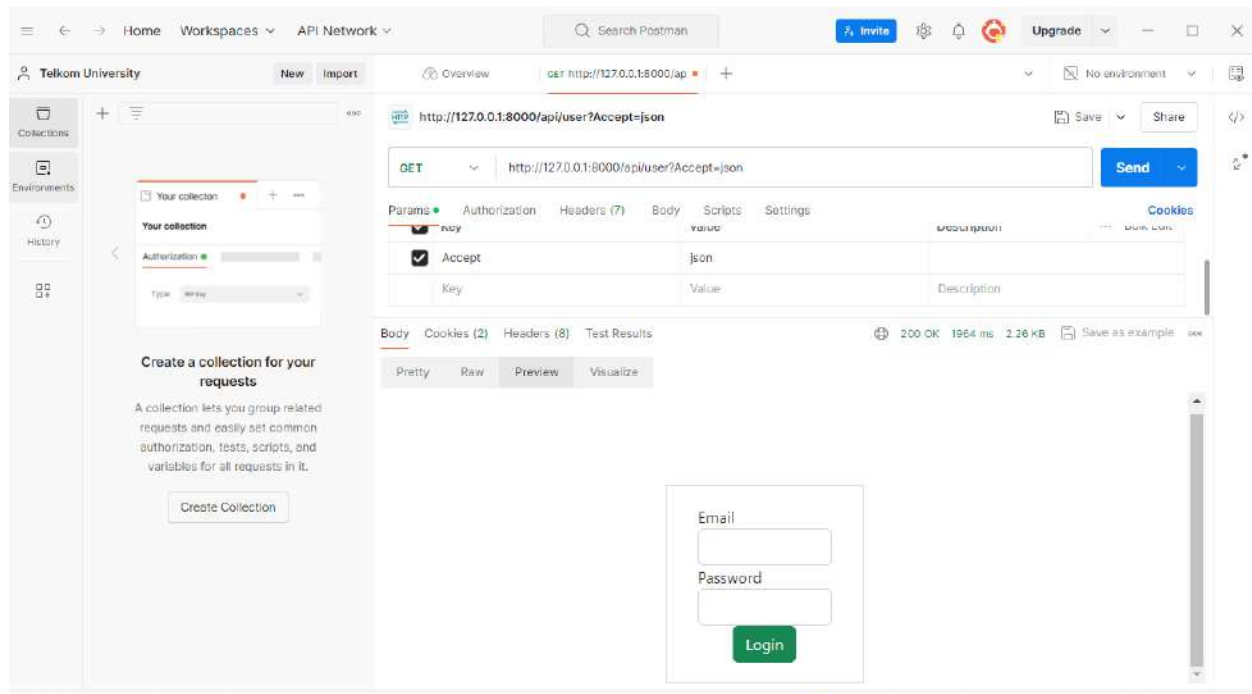
```
11 {
81
82     },201);
83 }
84
85 /**
86  * Remove the specified resource from storage.
87  */
88 public function destroy(string $id)
89 {
90     $isdelete = Product::destroy($id);
91
92     if($isdelete) {
93         return response()->json([
94             'status' => true,
95             'message' => 'Delete success'
96         ],200);
97     } else {
98         return response()->json([
99             'status' => false,
100             'message' => 'Delete Error'
101         ],200);
102     }
103 }
104 }
```

nothing to commit, working tree clean
PS C:\Users\Hello\Documents\indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI>

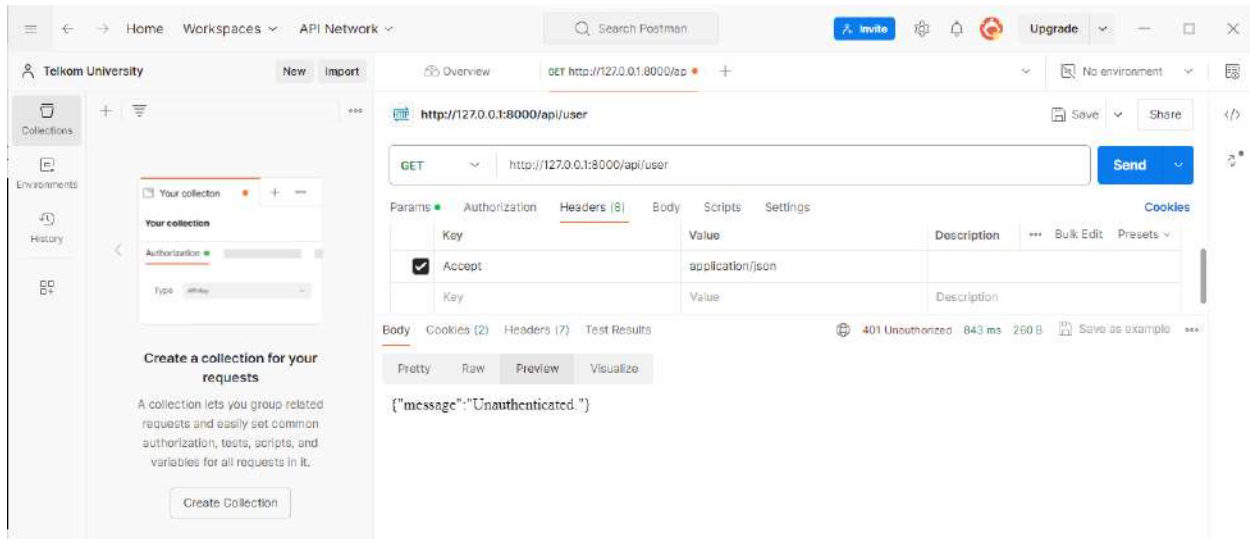
Setelah install laravel sanctum maka otomatis akan muncul “personal_access_tokens” di localhost\phpMyAdmin, seperti tampilan berikut.



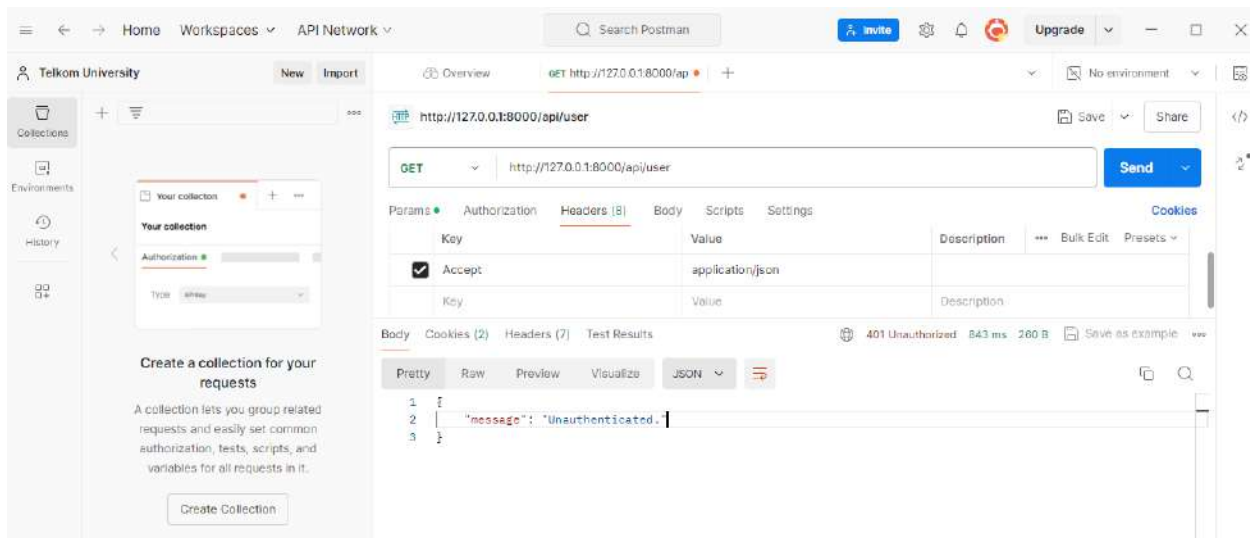
Tampilan awal login user sebelum ditambahkan header json atau masih tampilan web.



Tampilan postman setelah diberi header json.



Setelah ditambahkan autentikasi dan tidak boleh masuk sebelum login

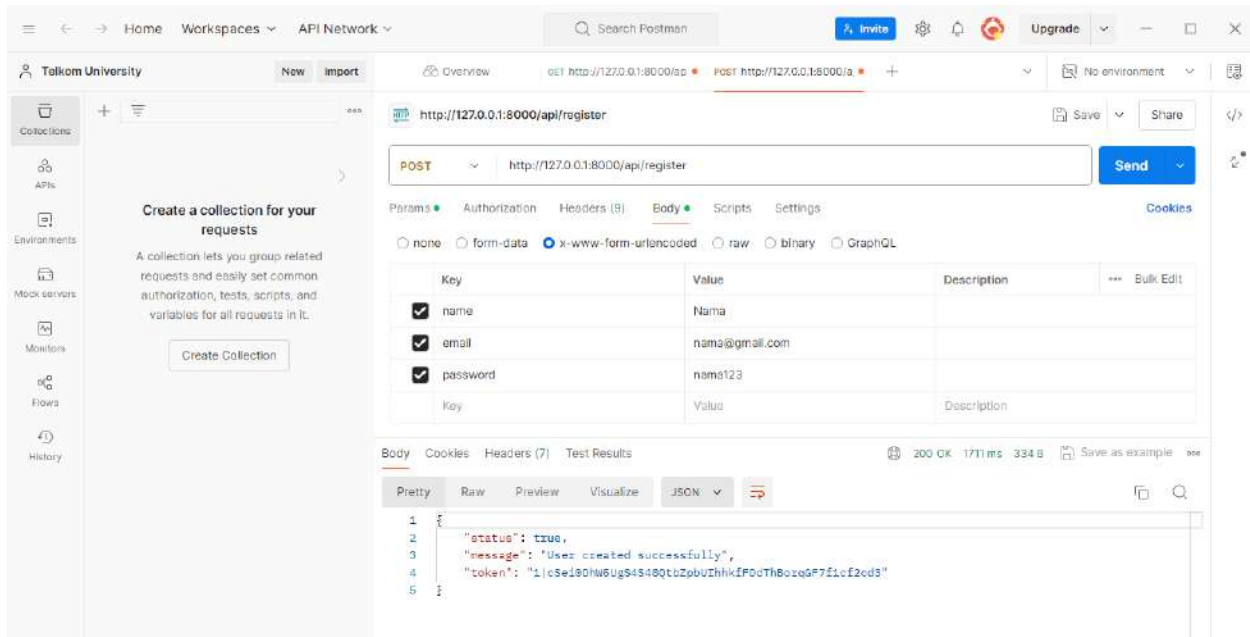


Buat folder Api dalam folder Controller

```
nothing to commit, working tree clean
PS C:\Users\Hello\Documents\indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI> php artisan make:controller Api\AuthController

[INFO] Controller [C:\Users\Hello\Documents\indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI\app\Http\Controllers\Api\AuthController.php] created successfully.
PS C:\Users\Hello\Documents\indriana 27-05-24\Tutorial_06_1304201002_INDRIANA-NOVIYANTI> 
```


Create Token Success



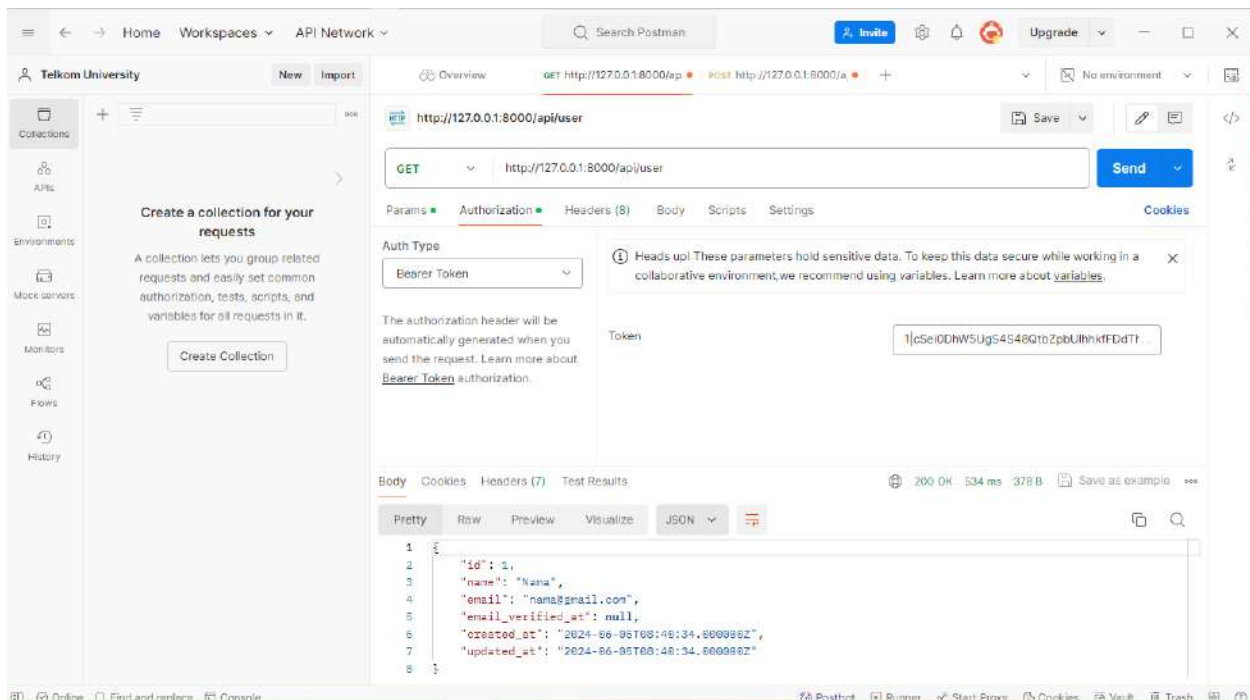
The screenshot shows the Postman interface with a POST request to `http://127.0.0.1:8000/api/register`. The request body is in the 'Body' tab, showing a JSON object with the following fields:

Key	Value	Description
<input checked="" type="checkbox"/> name	Nama	
<input checked="" type="checkbox"/> email	nama@gmail.com	
<input checked="" type="checkbox"/> password	nama123	

The response is shown in the 'Body' tab, displaying a JSON object with the following fields:

```
1 {
2   "status": true,
3   "message": "User created successfully",
4   "token": "1|cSei80W6UgS4S48QtbZpbUihkIFDcThBorq5F7f1cf2cd3"
5 }
```

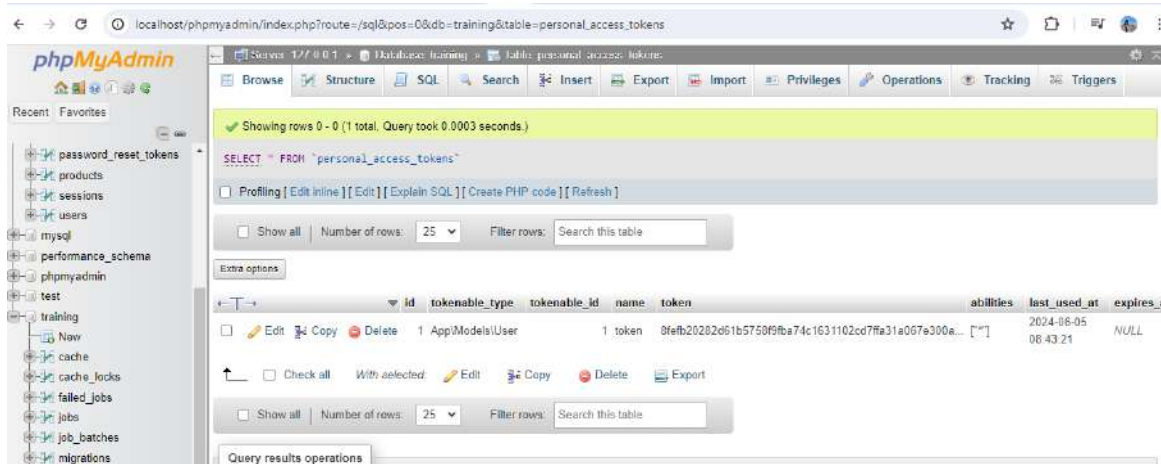
Login dengan Copy token success,



The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:8000/api/user`. The request is in the 'Authorization' tab, showing the 'Auth Type' as 'Bearer Token' and the 'Token' as `1|cSei80W6UgS4S48QtbZpbUihkIFDcThBorq5F7f1cf2cd3`. The response is shown in the 'Body' tab, displaying a JSON object with the following fields:

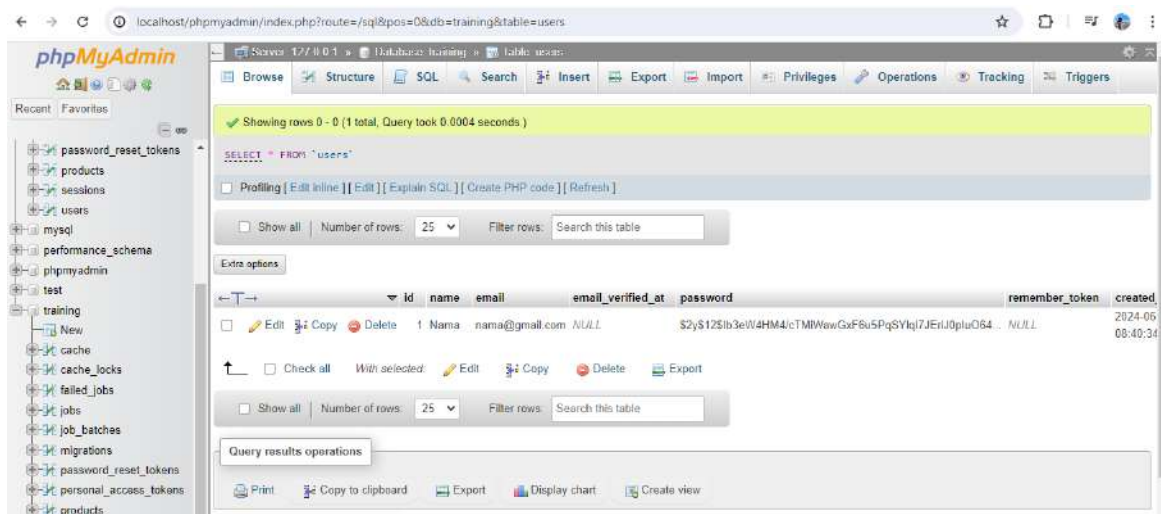
```
1 {
2   "id": 1,
3   "name": "Nama",
4   "email": "nama@gmail.com",
5   "email_verified_at": null,
6   "created_at": "2024-06-05T08:48:34.600998Z",
7   "updated_at": "2024-06-05T08:48:34.600998Z"
8 }
```

User yang berhasil login, masuk dalam database personal akses token, sebagai berikut.



The screenshot shows the phpMyAdmin interface with the 'personal_access_tokens' table selected. The table contains one row of data for a user named 'nama'.

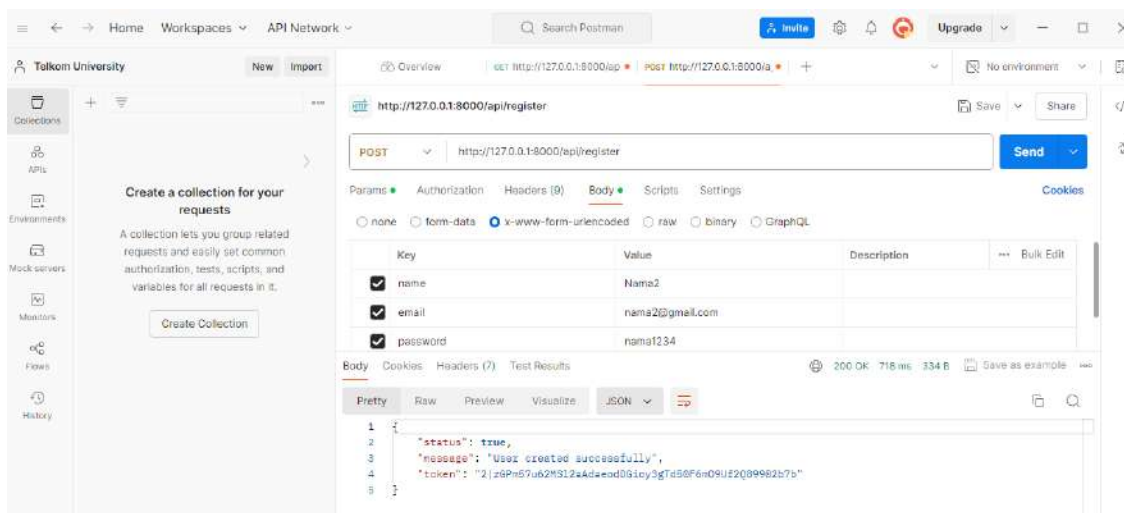
	id	tokenable_type	tokenable_id	name	token	abilities	last_used_at	expires_at
<input type="checkbox"/>	1	App\Models\User	1	token	0febf20282d51b57509fba74c1631102cd77fa31a007e300a...	["*"]	2024-06-05 08:43:21	NULL



The screenshot shows the phpMyAdmin interface with the 'users' table selected. The table contains one row of data for a user named 'nama'.

	id	name	email	email_verified_at	password	remember_token	created_at
<input type="checkbox"/>	1	Nama	nama@gmail.com	NULL	\$2y\$12\$ib3eW4HM4cTMMWawGxFeu5PqSYkq7JErlU0pluO64...	NULL	2024-06-05 08:40:34

Hasil registrasi setelah validasi diterapkan

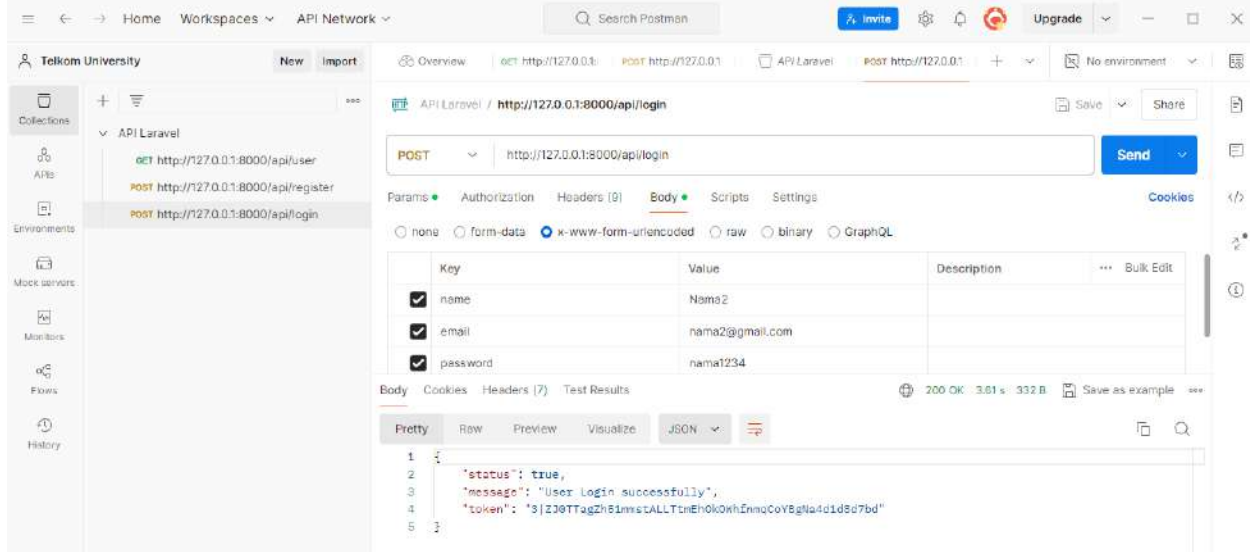


The screenshot shows a Postman interface with a successful POST request to the 'http://127.0.0.1:8000/api/register' endpoint. The response is a JSON object indicating successful registration.

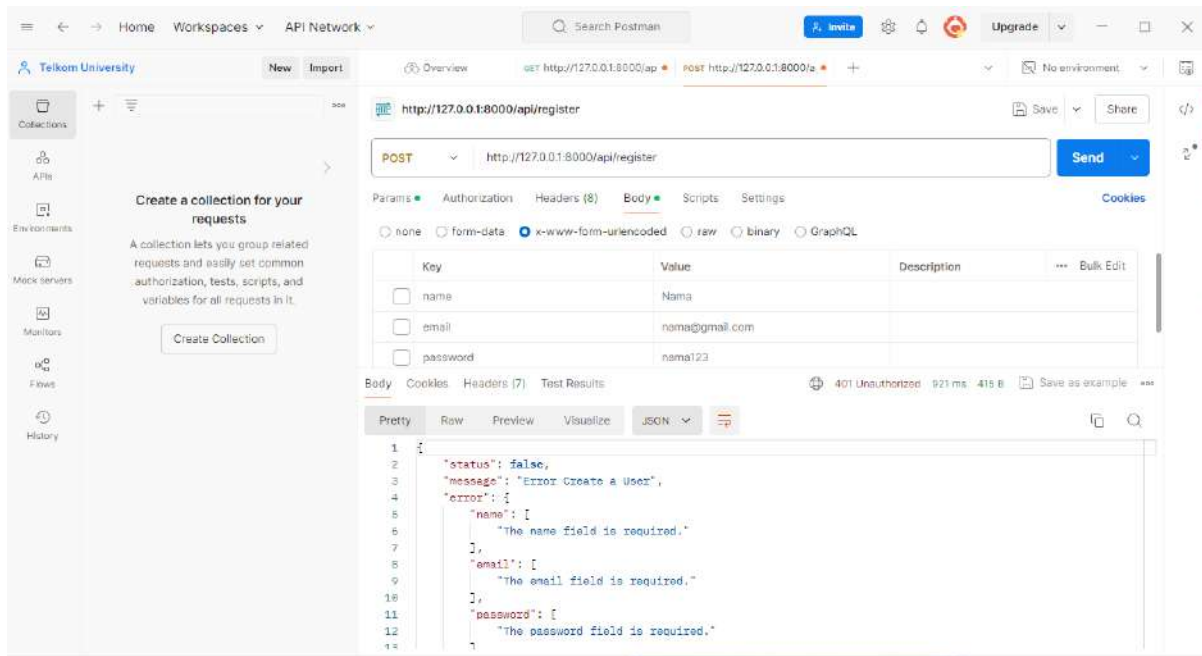
Key	Value	Description
name	Nama2	
email	nama2@gmail.com	
password	nama1234	

Body
<pre>{ "status": true, "message": "User created successfully", "token": "2 z6Pn57ub2M312x4deod091cy3gTd50F6n09U#2Q09982b7b" }</pre>

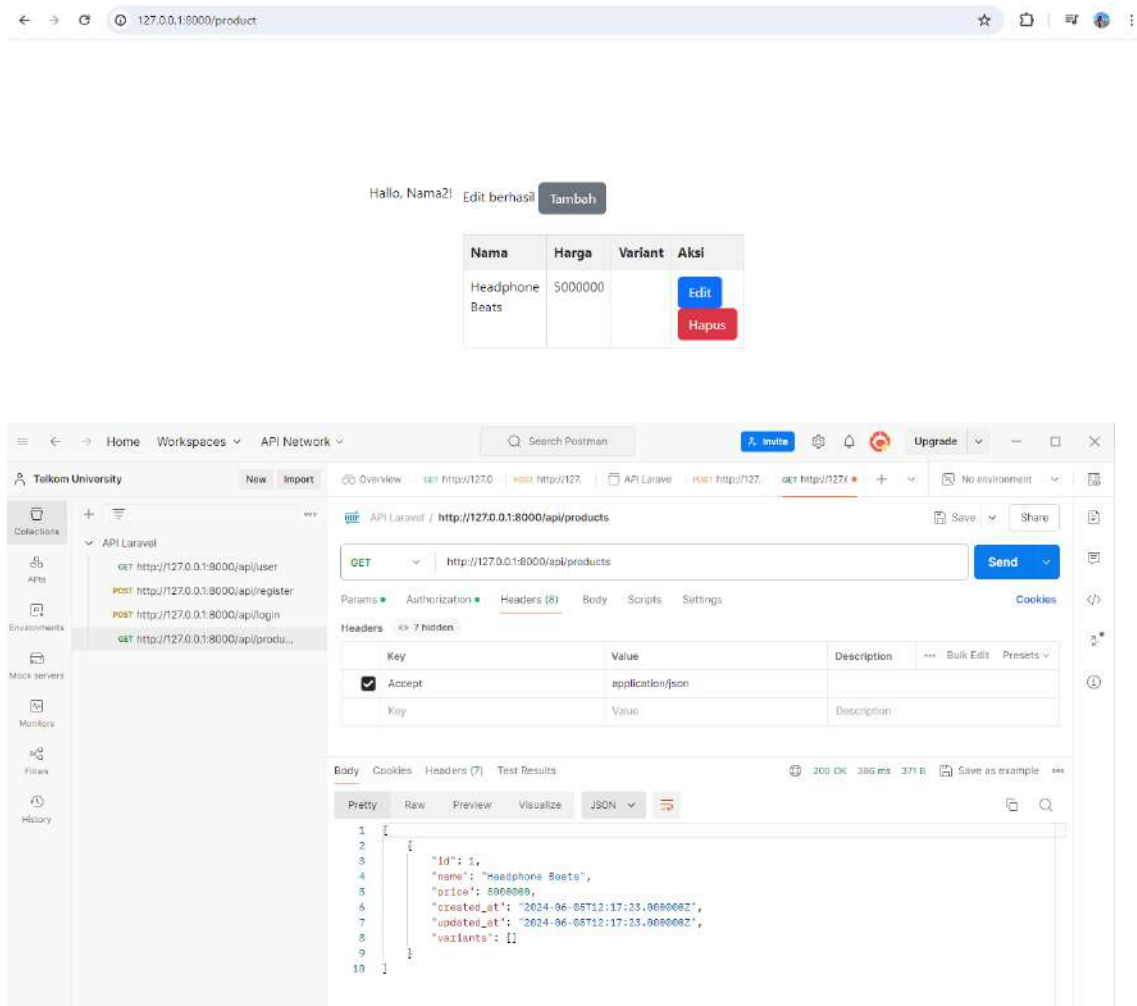
Dan ketika login dengan email yang sama menghasilkan token yang berbeda



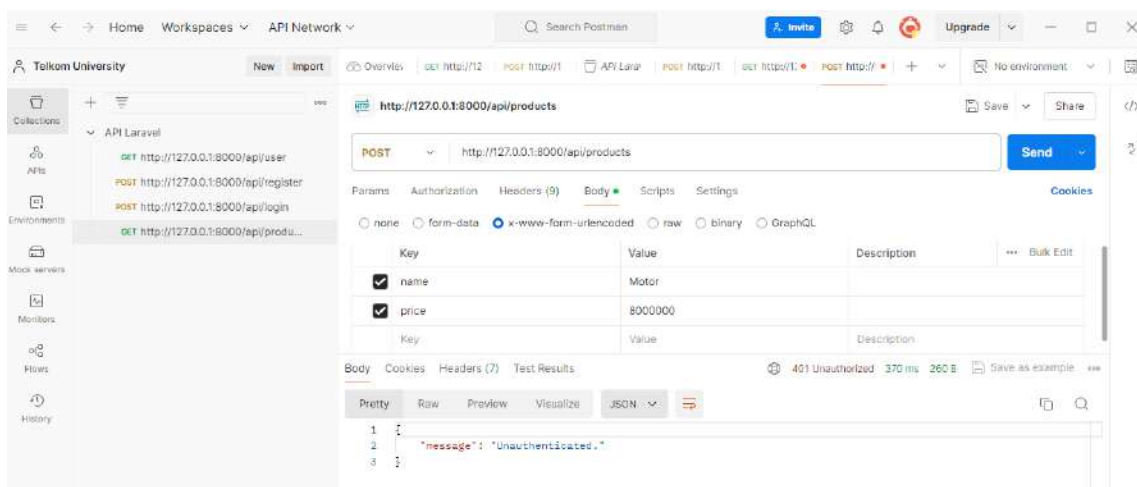
Validasi berhasil di aplikasikan, sehingga apabila field body tidak terisi maka akan ada pesan error.

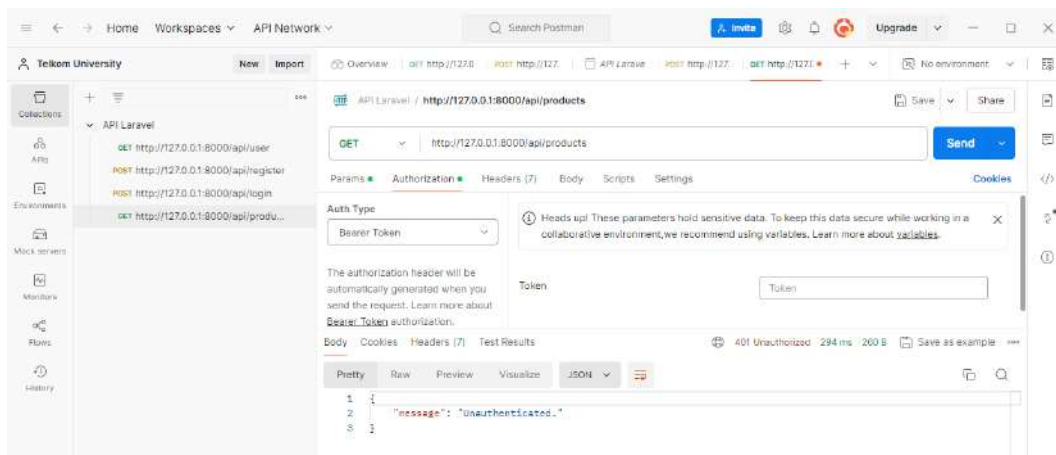


Ketika input data baru pada web, maka API postman mengikuti

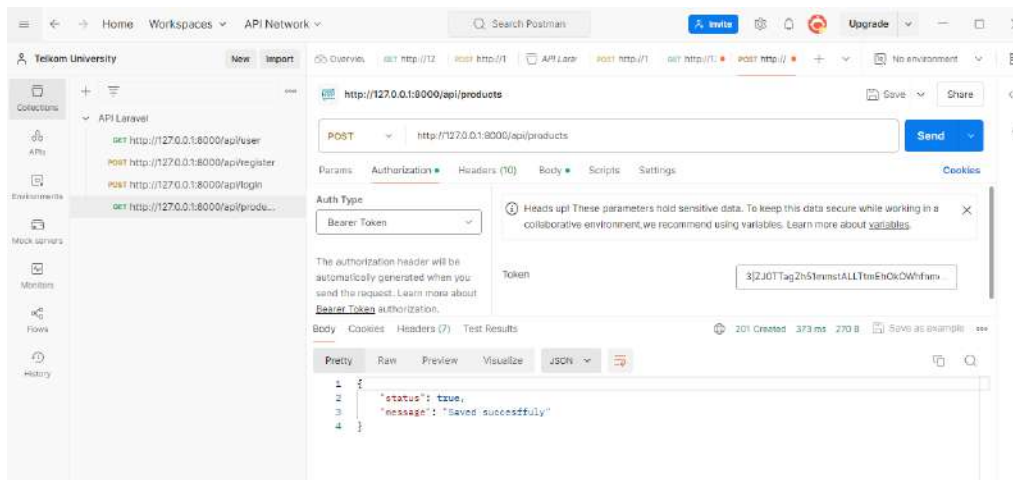


Ketika tidak login pakai token, maka tidak bisa akses product, termasuk menambahkan, edit, maupun delete.

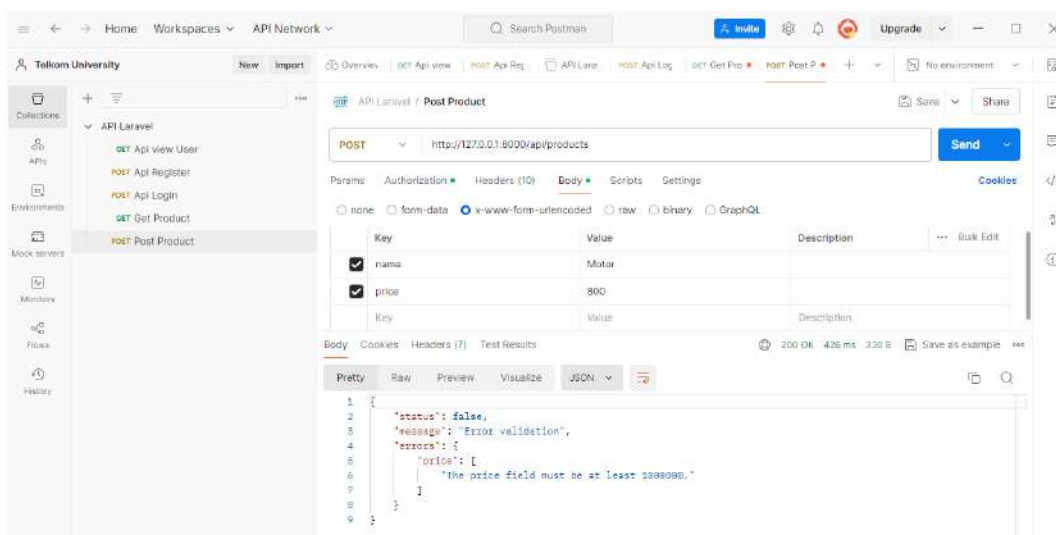




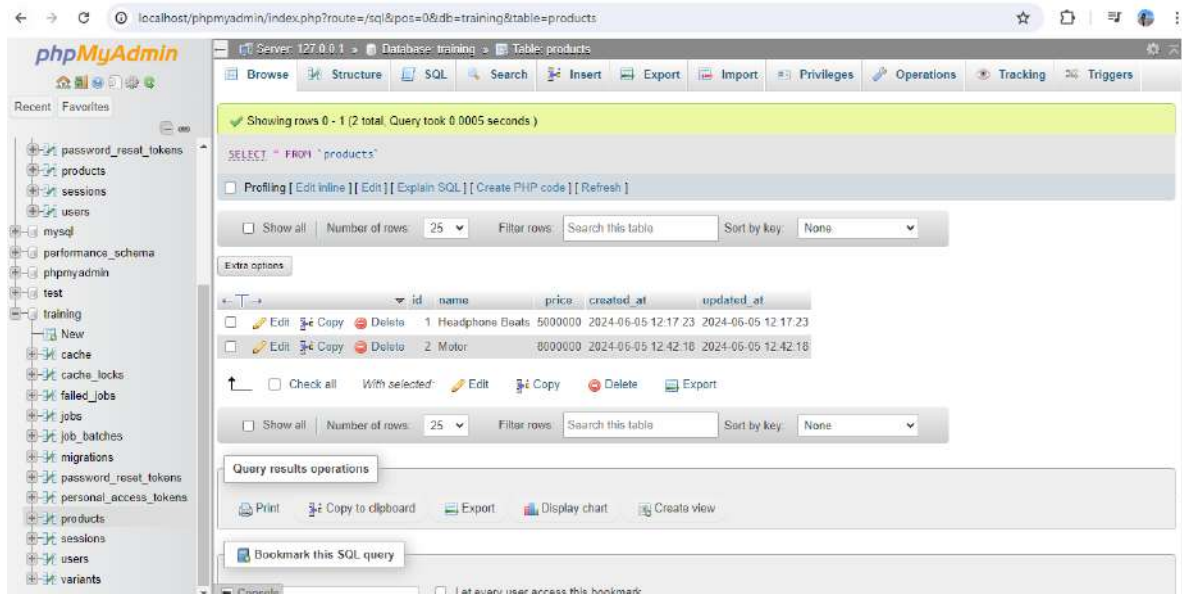
Dan ketika telah berhasil login menggunakan token, maka berhasil menambahkan product



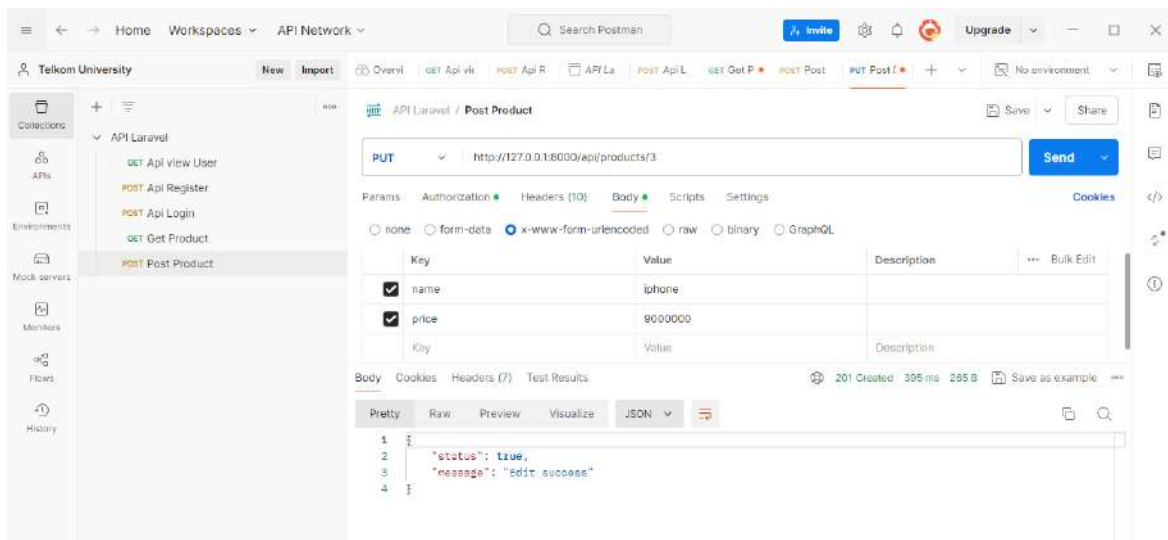
Muncul pesan error ketika input product tidak sesuai requirement



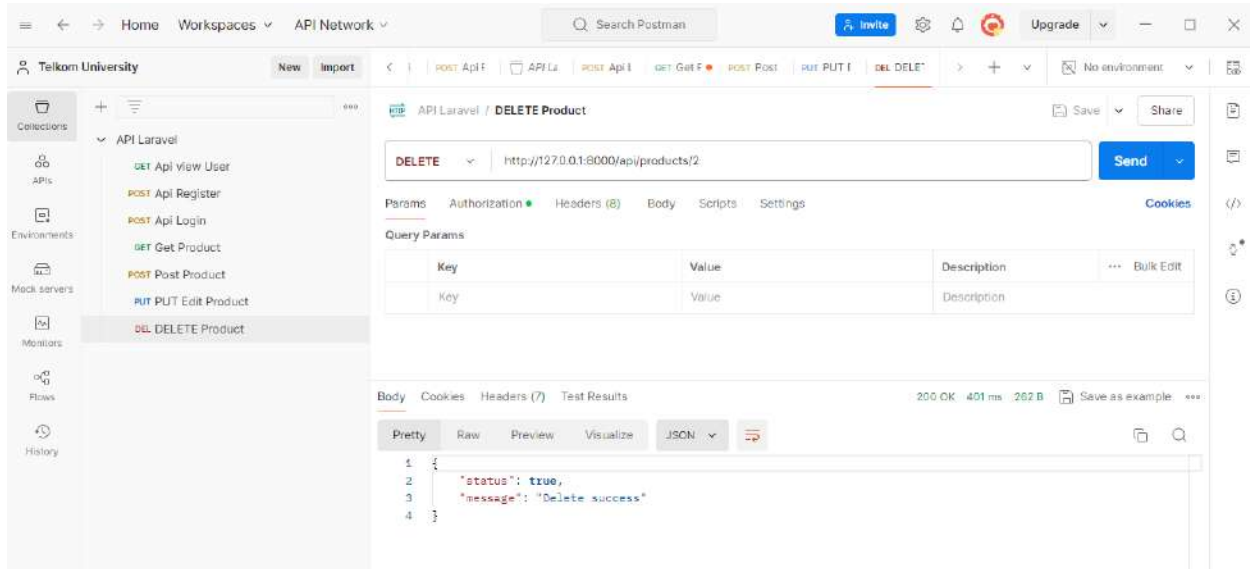
Product langsung terdeteksi di localhost\phpMyAdmin



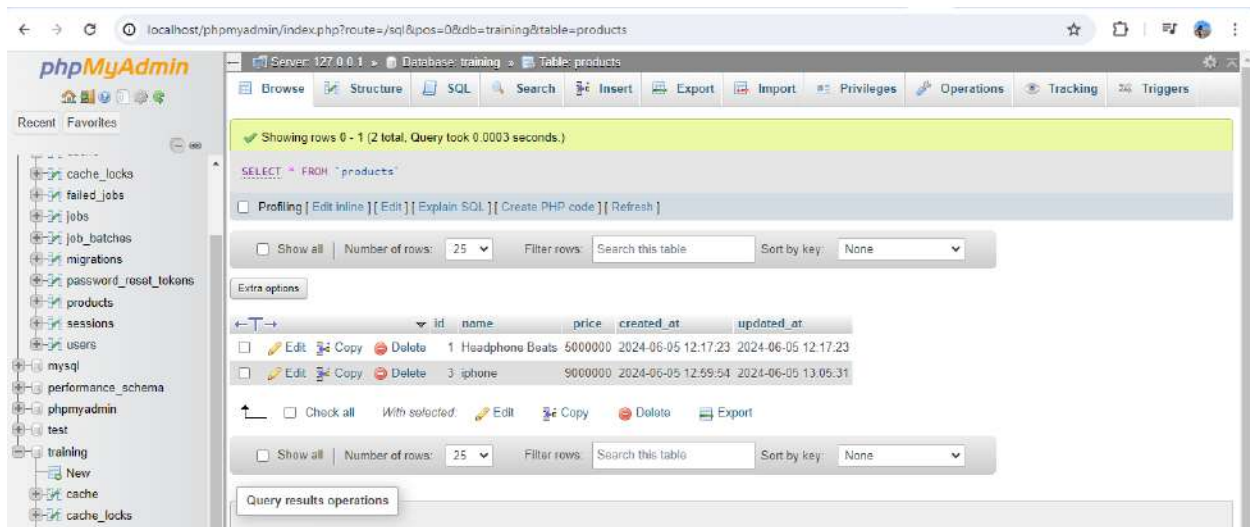
Edit product dengan “PUT” succeed



Delete Product dengan id 2 Succeed



Product dengan id 2 berhasil terhapus di localhost\\phpMyAdmin



Link Github:

https://github.com/Indriana-create/Tutorial_07_ABAP_API_LARAVEL-MySQL.git