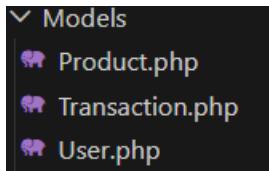
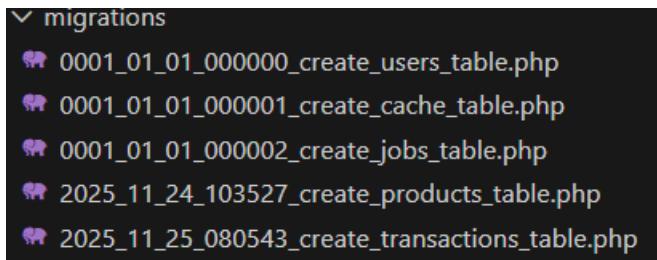


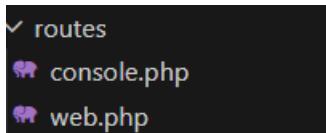
Pada controller di sini ini kita hanya menggunakan 2 yaitu controller untuk produk dan transaction



Pada model di sini ini kita hanya menggunakan 2 yaitu model untuk produk dan transaction



Pada migration juga kita menggunakan 2 yaitu migration untuk table product dan transaction



Untuk routing saya gunakan dan masukkan codenya pada web.php karena code yang dibuat pada route tidak banyak sehingga menurut saya akan lebih baik jika hanya pada 1 file.

Sekarang kita akan masuk ke dalam test back endnya

The screenshot shows the phpMyAdmin interface for the 'technical_test' database. The 'products' table is selected. The table structure includes columns: id, name, price, stock, created_at, updated_at, and deleted_at. There are two rows in the table:

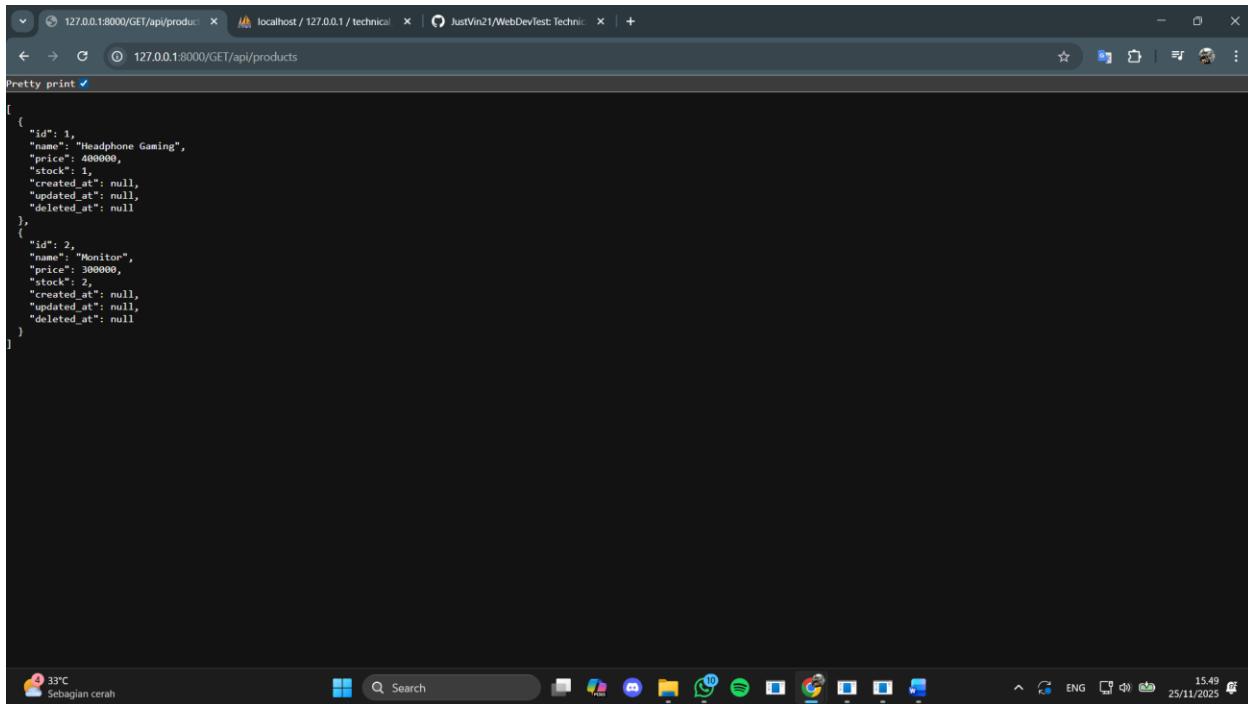
	id	name	price	stock	created_at	updated_at	deleted_at
	1	Headphone Gaming	400000	1	NULL	NULL	NULL
	2	Monitor	300000	2	NULL	NULL	NULL

Disini saya menggunakan mysql pada XAMPP, disini saya akan memasukkan data dummy pada table product.

The screenshot shows the phpMyAdmin interface for the 'technical_test' database. The 'products' table is selected. The table structure includes columns: id, name, price, stock, created_at, updated_at, and deleted_at. There are three rows in the table:

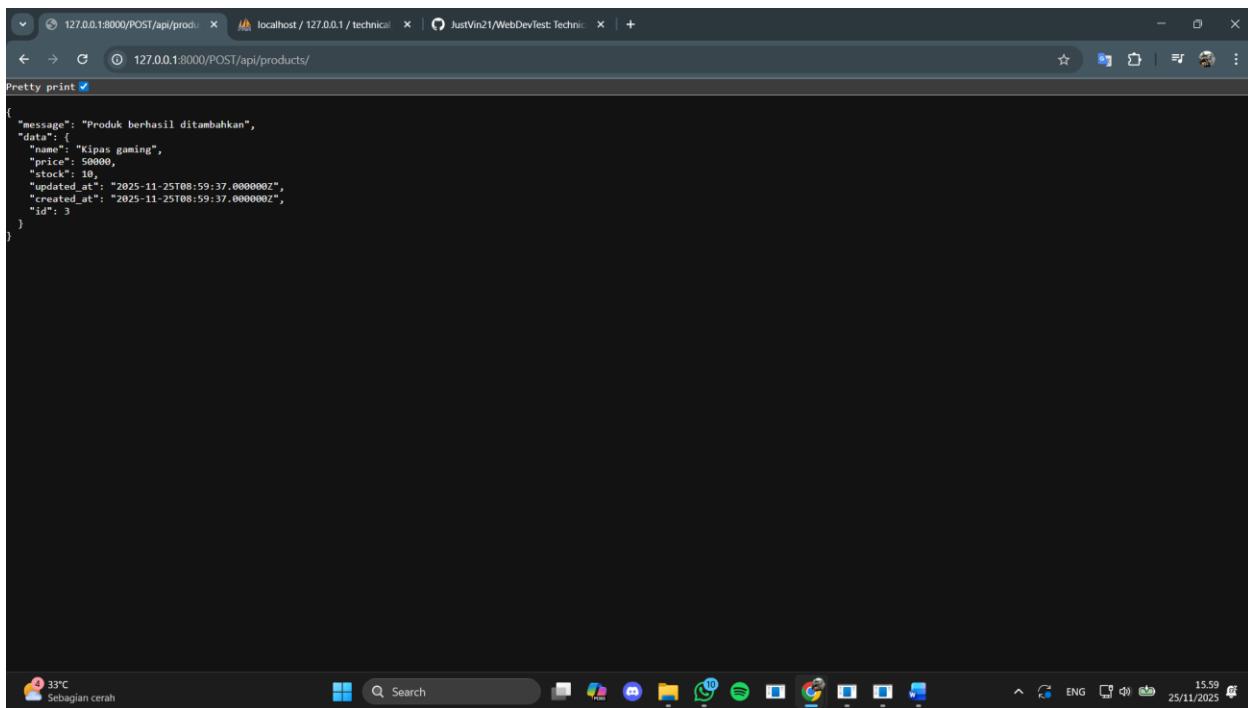
	id	name	price	stock	created_at	updated_at	deleted_at
	1	Headphone Gaming	400000	1	NULL	NULL	NULL
	2	Monitor	300000	2	NULL	NULL	NULL
	3	Laptop	500000	3	NULL	NULL	NULL

Disni saya telah membuat 2 data dummy dengan id 1 "Headphone Gaming" dan id 2 "Monitor"



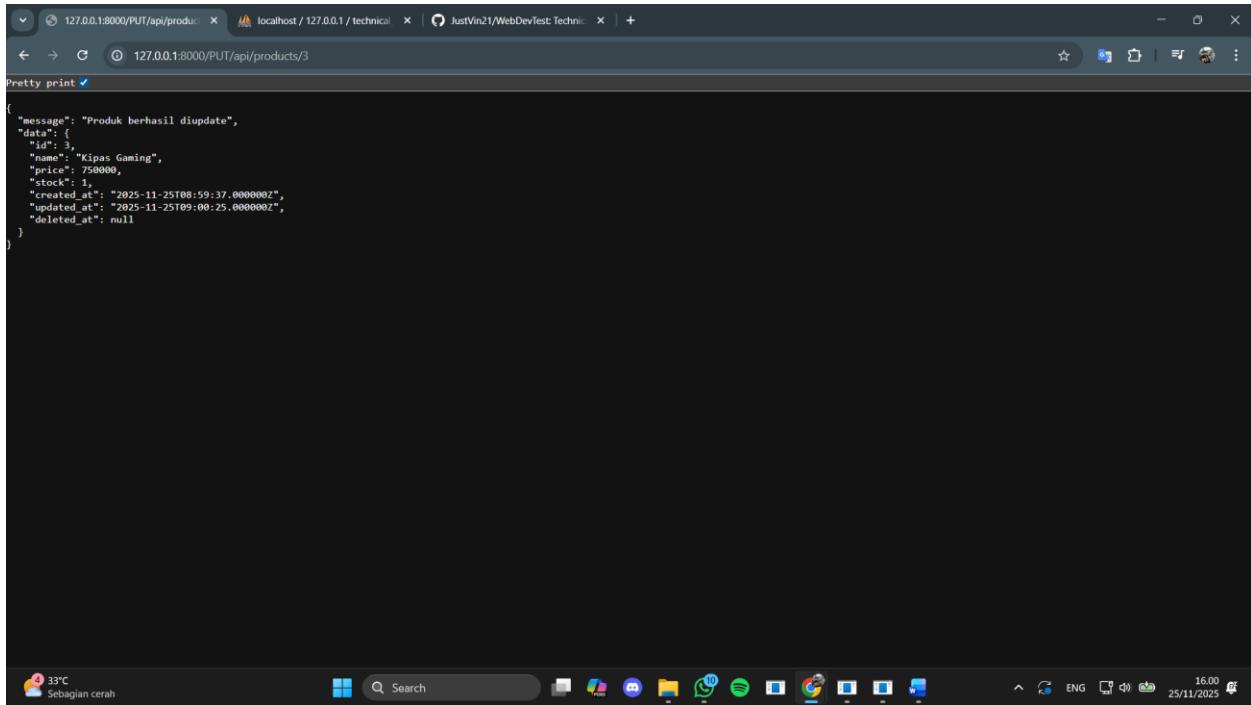
```
[{"id": 1, "name": "Headphone Gaming", "price": 400000, "stock": 1, "created_at": null, "updated_at": null, "deleted_at": null}, {"id": 2, "name": "Monitor", "price": 300000, "stock": 2, "created_at": null, "updated_at": null, "deleted_at": null}]
```

Sekarang kita akan lihat data yang telah di masukkan tadi. Dengan memasukkan link <http://127.0.0.1:8000/GET/api/products> yang bertujuan untuk GET (melihat semua data pada table) dapat kita lihat bahwa 2 data dummny yang dimasukkan pada mysql tadi telah muncul dan dapat dilihat.



```
{"message": "Produk berhasil ditambahkan", "data": {"id": 3, "name": "Kipas gaming", "price": 50000, "stock": 10, "updated_at": "2025-11-25T08:59:37.000000Z", "created_at": "2025-11-25T08:59:37.000000Z", "deleted_at": null}}
```

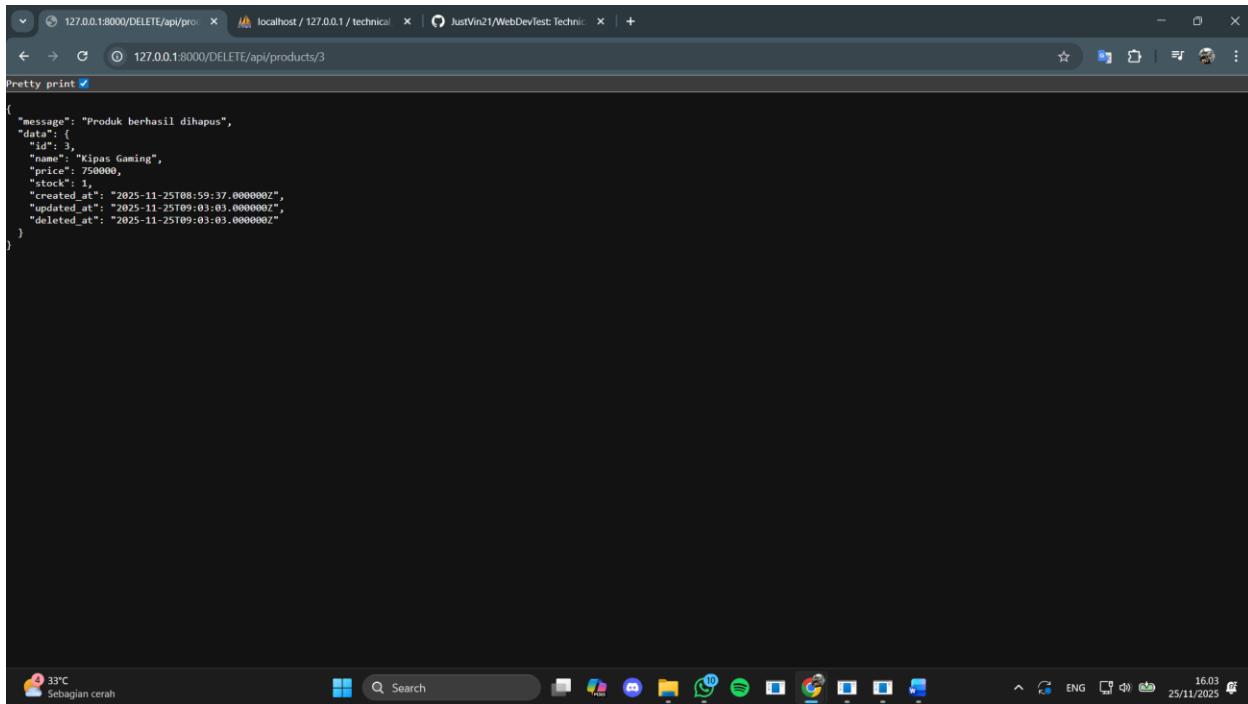
Disini kita akan melakukan POST yang bertujuan untuk menambahkan data dummy namun tidak melalui mysql tetapi bisa secara langsung pada Laravel. Dengan memasukkan link <http://127.0.0.1:8000/POST/api/products> maka kita dapat melakukan POST. Setelah melakukan POST maka data akan masuk ke table dan disimpan.



A screenshot of a Microsoft Edge browser window. The address bar shows the URL 127.0.0.1:8000/PUT/api/products/3. The page content is a JSON response with "Pretty print" checked. The JSON object contains a "message" key with the value "Produk berhasil diupdate" and a "data" key which is another JSON object containing "id": 3, "name": "Kipas Gaming", "price": 750000, "stock": 1, "created_at": "2025-11-25T08:59:37.000000Z", "updated_at": "2025-11-25T09:00:25.000000Z", and "deleted_at": null. The browser's taskbar at the bottom shows various pinned icons and the date/time as 25/11/2025.

```
{ "message": "Produk berhasil diupdate", "data": { "id": 3, "name": "Kipas Gaming", "price": 750000, "stock": 1, "created_at": "2025-11-25T08:59:37.000000Z", "updated_at": "2025-11-25T09:00:25.000000Z", "deleted_at": null } }
```

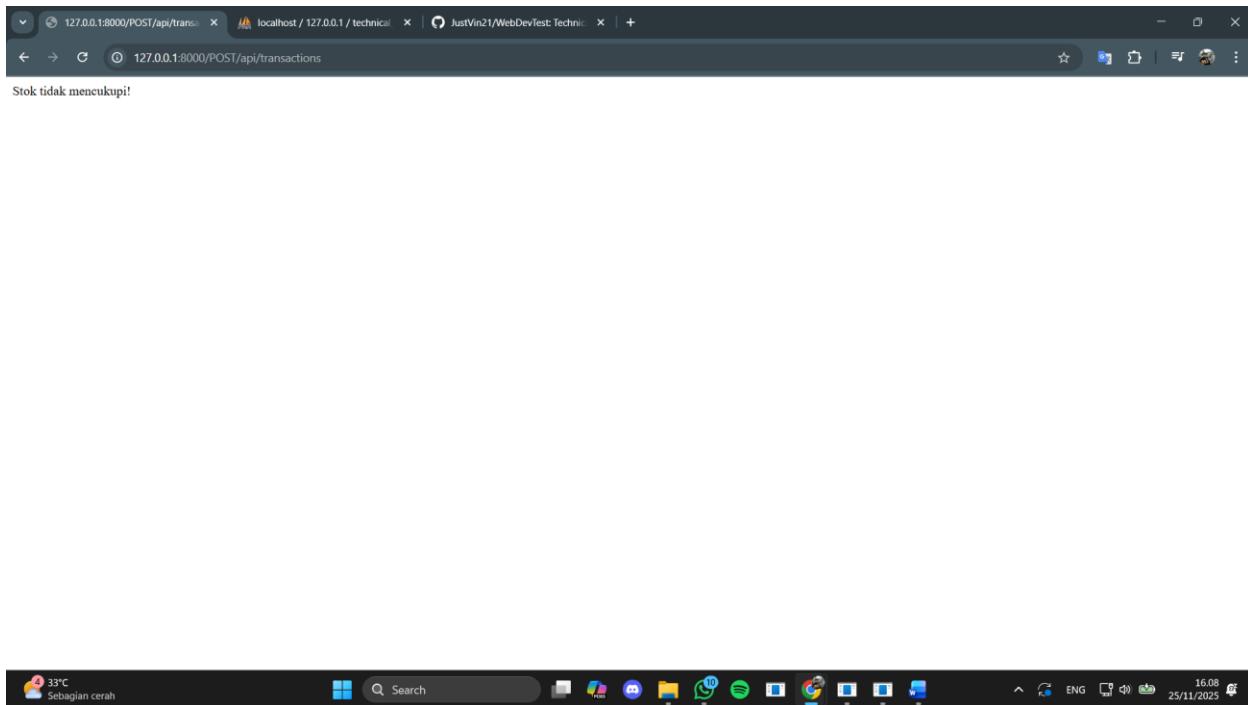
Disini kita melakukan PUT yang bertujuan untuk mengupdate data lama ke data yang terbaru. Dengan memasukkan link ini <http://127.0.0.1:8000/PUT/api/products/3> maka akan mengupdate data pada id 3 dengan data yang terbaru. Data akan disimpan di database.



A screenshot of a Microsoft Edge browser window. The address bar shows the URL 127.0.0.1:8000/DELETE/api/products/3. The main content area displays a JSON response with "Pretty print" checked:

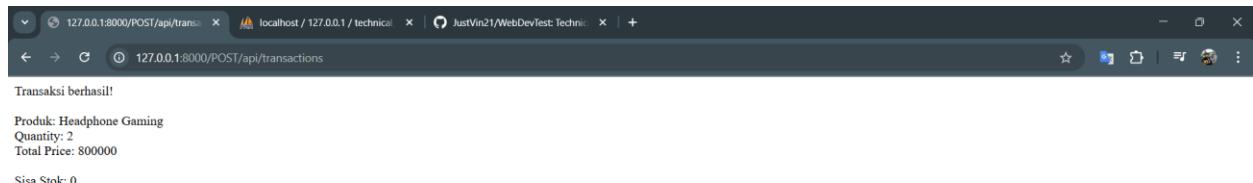
```
{ "message": "Produk berhasil dihapus", "data": { "id": 3, "name": "Kipas Gaming", "price": 750000, "stock": 1, "created_at": "2025-11-25T08:59:37.000000Z", "updated_at": "2025-11-25T09:01:03.000000Z", "deleted_at": "2025-11-25T09:03:03.000000Z" } }
```

Disini kita melakukan DELETE yang bertujuan unuk menghapus data yang ada. Dengan memasukkan link <http://127.0.0.1:8000/DELETE/api/products/3> maka akan secara otomatis menghapus data dengan id 3 dan akan hilang dari data base pada Laravel namun tidak pada database mysql jadinya seperti di hide dari Laravel namun masih ada pada mysql.



A screenshot of a Microsoft Edge browser window. The address bar shows the URL 127.0.0.1:8000/POST/api/transactions. The main content area displays the message "Stok tidak mencukupi!" (Stock is insufficient!).

Disni kita melakukan POST transactions dengan ide 1 “Headphone Gaming” karena kita akan melakukan pembelian id 1 dengan quantity 2 maka headphone gaming tidak dapat di beli karen jumlah stock kurang. Jika stock kurang maka akan menampilkan pesan stock tidak mencukupi.



Namun ketika kita menyiapkan stocknya sama dengan atau lebih dari quantity yang dibeli maka produk dapat dibeli dan akan menampilkan pesan ini. Dengan memasukkan link <http://127.0.0.1:8000/POST/api/transactions> maka kita dapat membeli produk dan akan mengupdate data terbaru.