



# **"IMPLEMENTASI DAN PENGUJIAN RESTFUL API: SISTEM MANAJEMEN PRODUK DAN KERANJANG BELANJA"**

Penyusun

Nama : Ilyas Nazma Esa Iskandar (20240040297)  
Deandry Zidan Agustia (20230040264)  
Alief Hazel Nurhakim (2024004027)

Kelas : TI24J

# POST /api/register

- Fungsi:
- Endpoint ini digunakan untuk mendaftarkan pengguna baru ke dalam sistem.
- Penjelasan:
- User mengirim data seperti nama, email, dan password. Sistem akan menyimpan data user ke database dengan password yang sudah dihash demi keamanan.

The screenshot shows a POST request to `http://localhost:3000/api/register`. The request body is a JSON object:

```
1 {  
2   "name": "Deandry4",  
3   "email": "zidandry@gmail.com",  
4   "password": "123456789"  
5 }  
6  
7
```

The response is a `200 OK` status with a message: "Register berhasil".

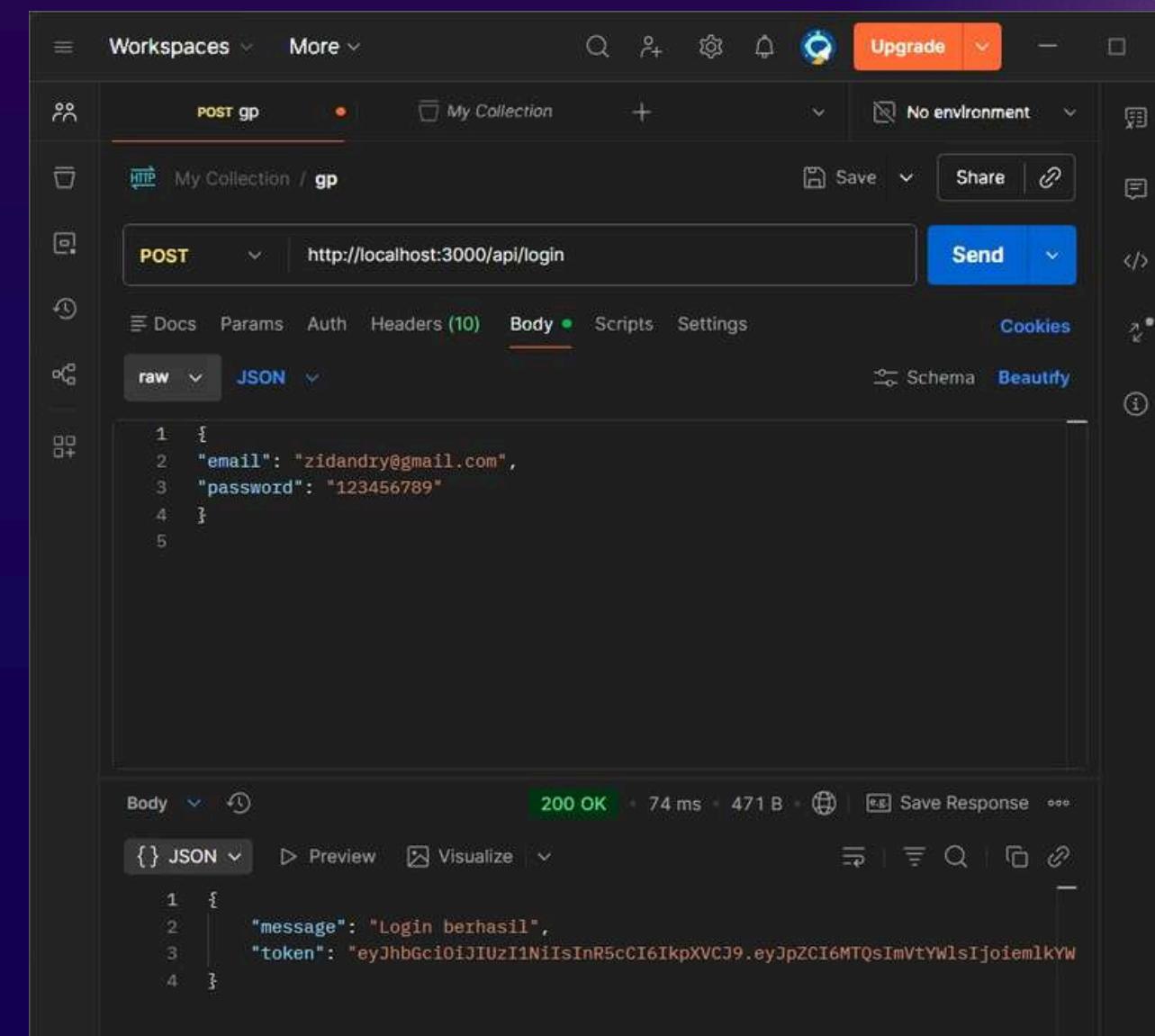
# POST /api/login

Fungsi:

Endpoint ini digunakan untuk autentikasi pengguna.

Penjelasan:

Jika email dan password benar, sistem akan mengembalikan JWT token yang digunakan untuk mengakses endpoint yang dilindungi (protected routes).



```
POST http://localhost:3000/api/login
```

```
1: {
2:   "email": "zidandry@gmail.com",
3:   "password": "123456789"
4: }
```

```
200 OK
```

```
1: {
2:   "message": "Login berhasil",
3:   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTQsImVtYWlsIjjoiemlkYW..."}
```



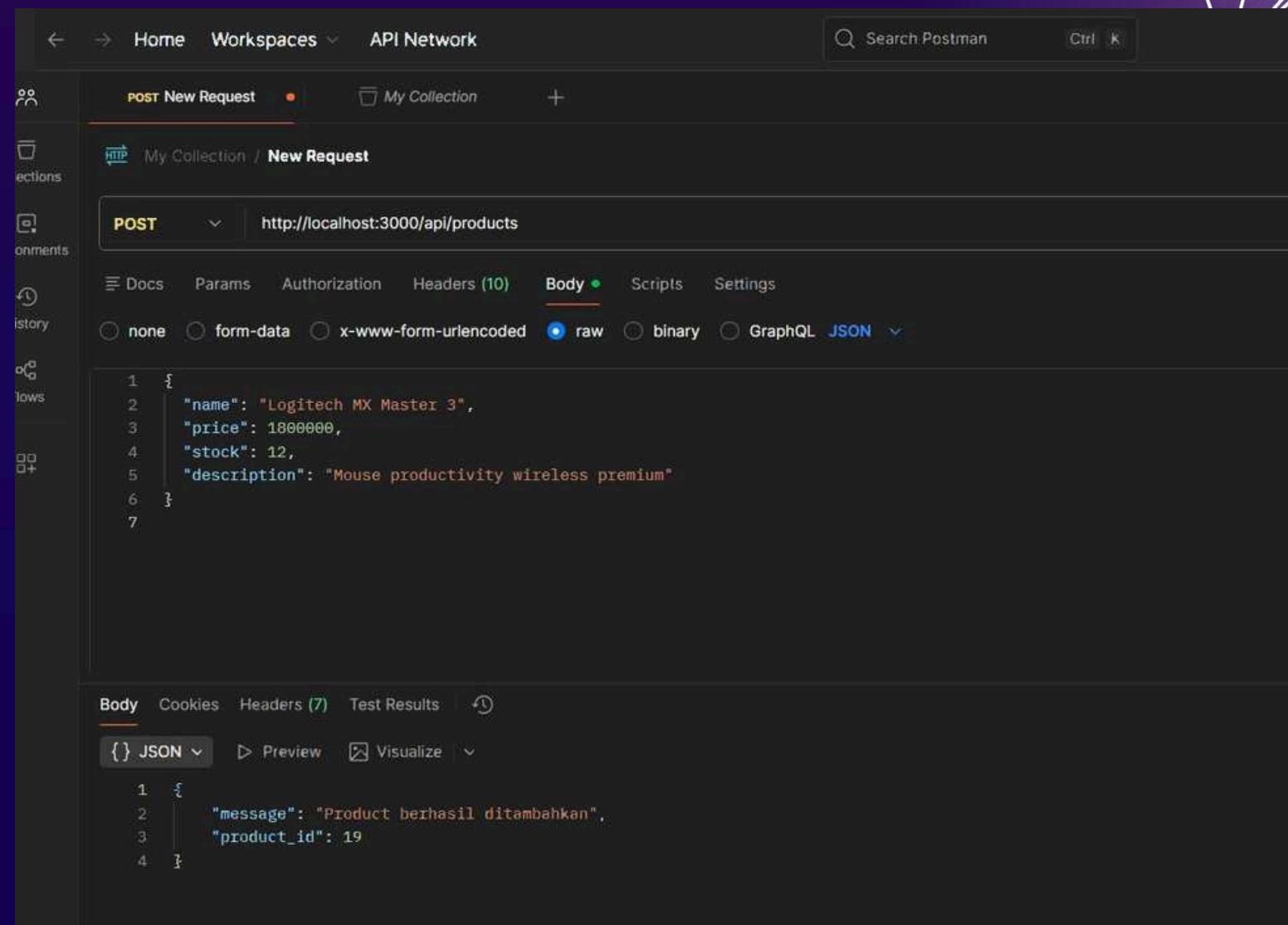
# GET /api/profile

Fungsi:

Mengambil data profil pengguna yang sedang login.

Penjelasan:

Endpoint ini bersifat protected. Data user diambil dari token JWT tanpa perlu query ulang berdasarkan ID manual.



The screenshot shows the Postman application interface. A POST request is being prepared to the URL `http://localhost:3000/api/products`. The request method is set to POST, and the body is defined in raw JSON format. The JSON payload is as follows:

```
1 {  
2   "name": "Logitech MX Master 3",  
3   "price": 1800000,  
4   "stock": 12,  
5   "description": "Mouse productivity wireless premium"  
6 }  
7
```

Below the request, the response body is displayed in JSON format, showing a successful addition of a product:

```
1 {  
2   "message": "Product berhasil ditambahkan",  
3   "product_id": 19  
4 }
```

# GET /api/users

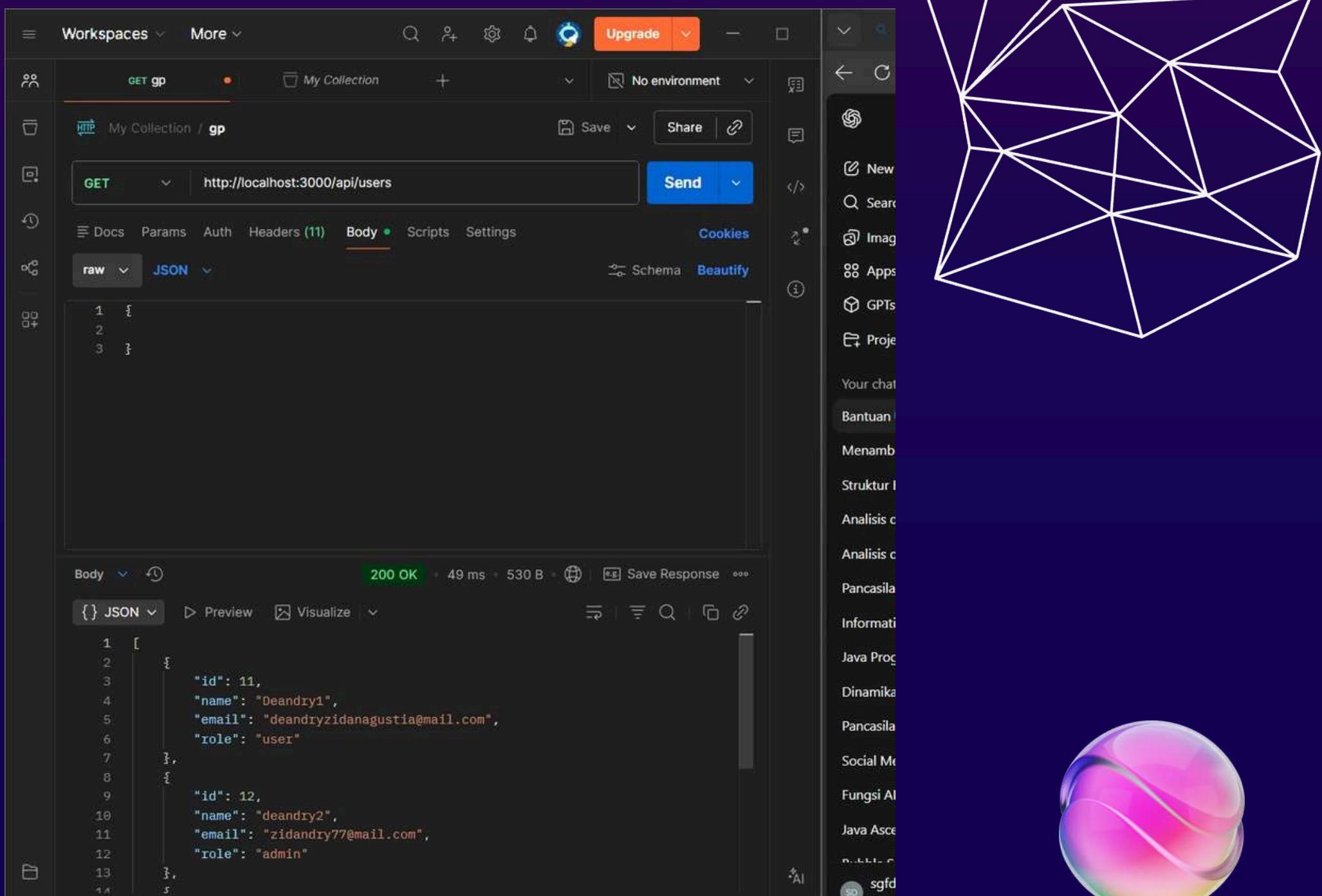
Fungsi:

Mengambil daftar seluruh user dalam sistem.

Penjelasan:

Endpoint ini hanya bisa diakses oleh admin.

Digunakan untuk keperluan manajemen user.



```
1 [  
2   {  
3     "id": 11,  
4     "name": "Deandry1",  
5     "email": "deandryzidanagustia@mail.com",  
6     "role": "user"  
7   },  
8   {  
9     "id": 12,  
10    "name": "deandry2",  
11    "email": "zidandry77@mail.com",  
12    "role": "admin"  
13  }  
14 ]
```

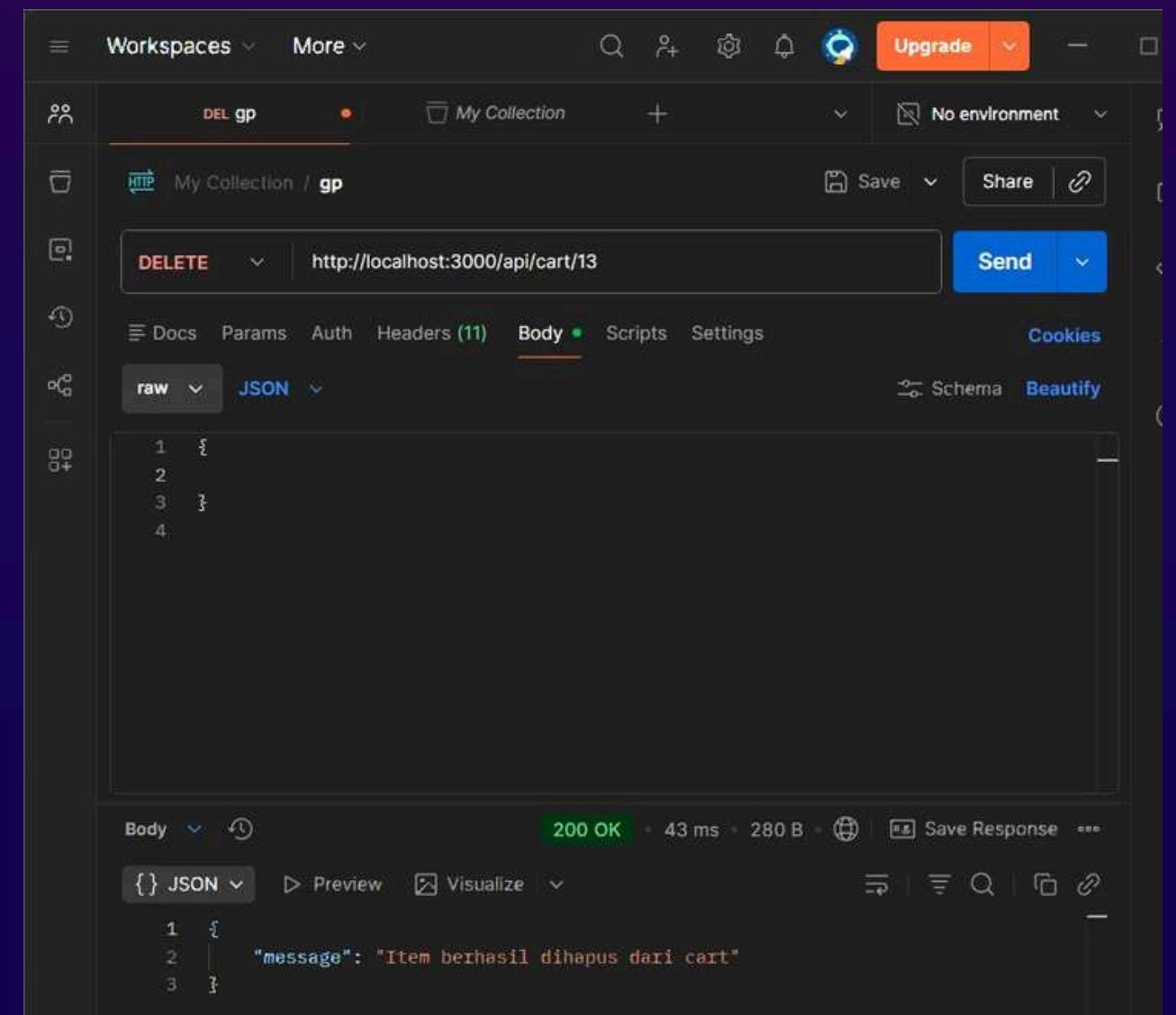
# DELETE /api/users/:id

Fungsi:

Menghapus akun user berdasarkan ID.

Penjelasan:

Digunakan oleh admin untuk menghapus user yang tidak valid atau melanggar kebijakan sistem.



The screenshot shows a Postman interface with a collection named "My Collection". A specific request is selected, labeled "gp". The method is set to "DELETE" and the URL is "http://localhost:3000/api/cart/13". The "Body" tab is active, showing a raw JSON payload with four lines of code: 1, 2, 3, and 4. Below the request, the response is displayed. The status is "200 OK" with a response time of "43 ms" and a size of "280 B". The response body is a JSON object with one key, "message", which has the value "Item berhasil dihapus dari cart".

```
1
2
3
4
```

```
1 {<br>}<br>2   "message": "Item berhasil dihapus dari cart"<br>3 }<br>4
```

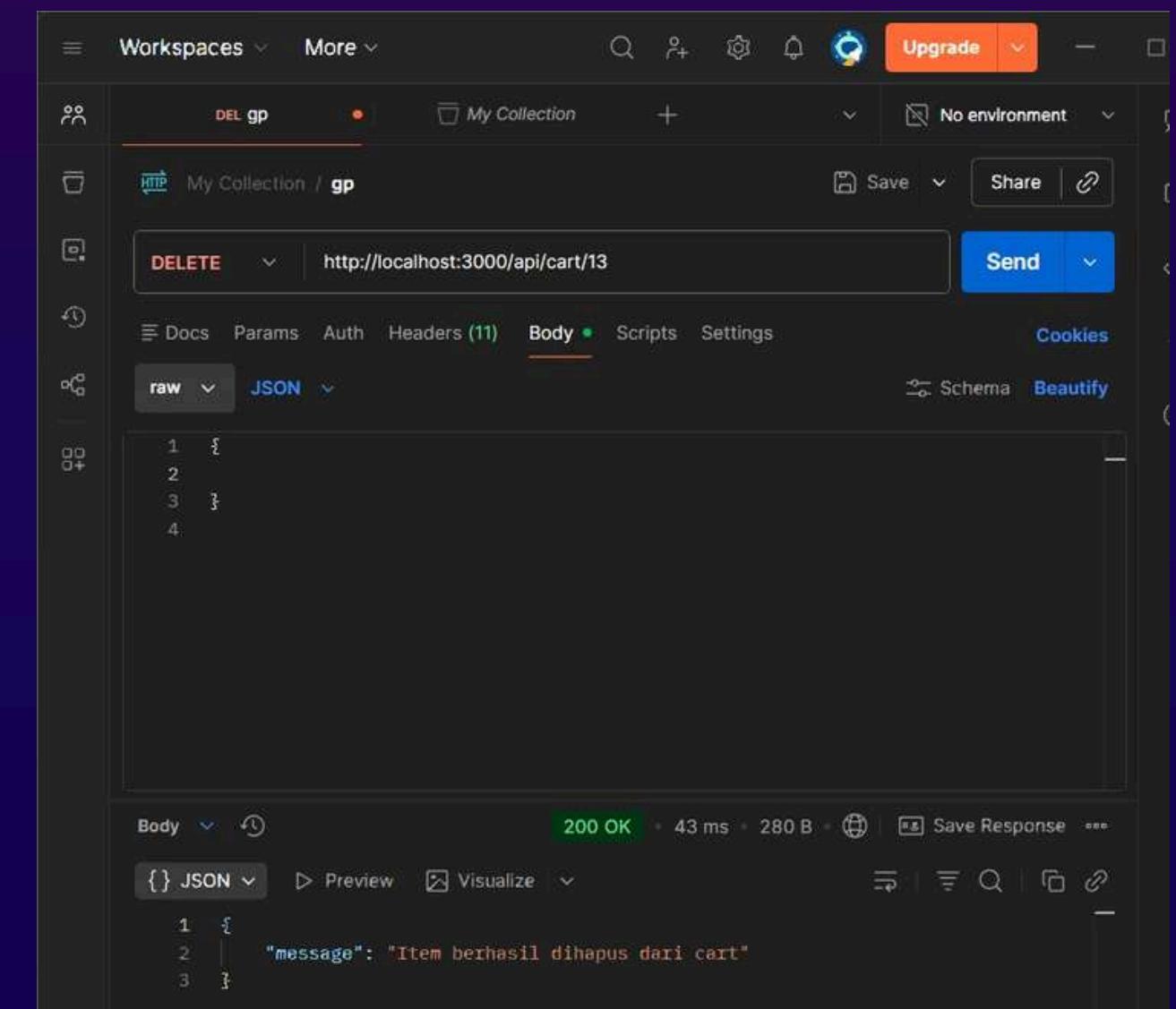
# GET /api/products

Fungsi:

Mengambil seluruh daftar produk.

Penjelasan:

Endpoint ini bersifat publik dan digunakan untuk menampilkan produk pada halaman katalog.

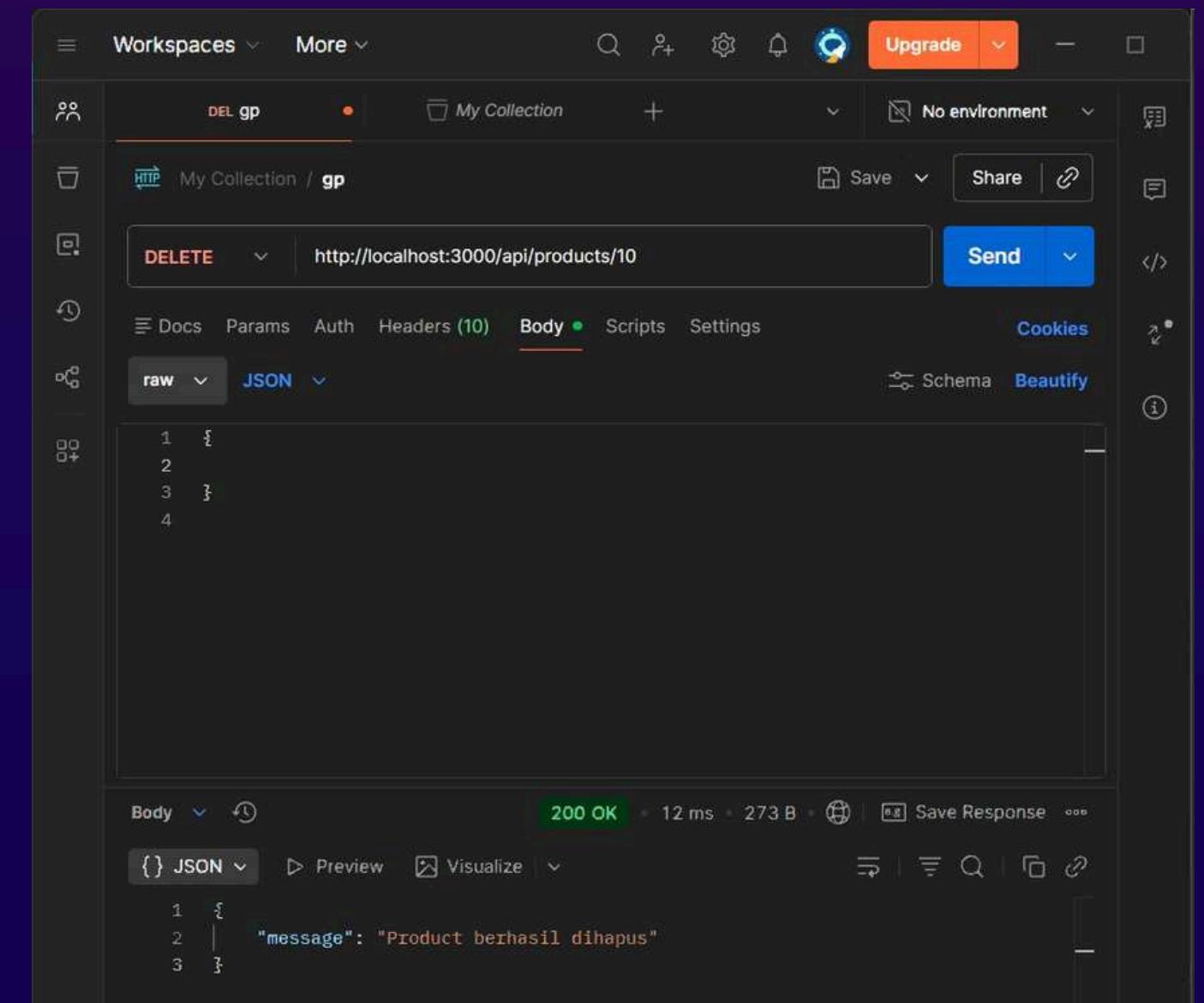


The screenshot shows a REST client interface with the following details:

- Request Method:** DELETE
- URL:** http://localhost:3000/api/cart/13
- Body:** An empty JSON object ({}).
- Response Status:** 200 OK
- Response Time:** 43 ms
- Response Size:** 280 B
- Response Content:** {"message": "Item berhasil dihapus dari cart"}

# Delete Products

menghapus produk sistem, digunakan oleh admin untuk menghapus produk yang sudah tidak di jual



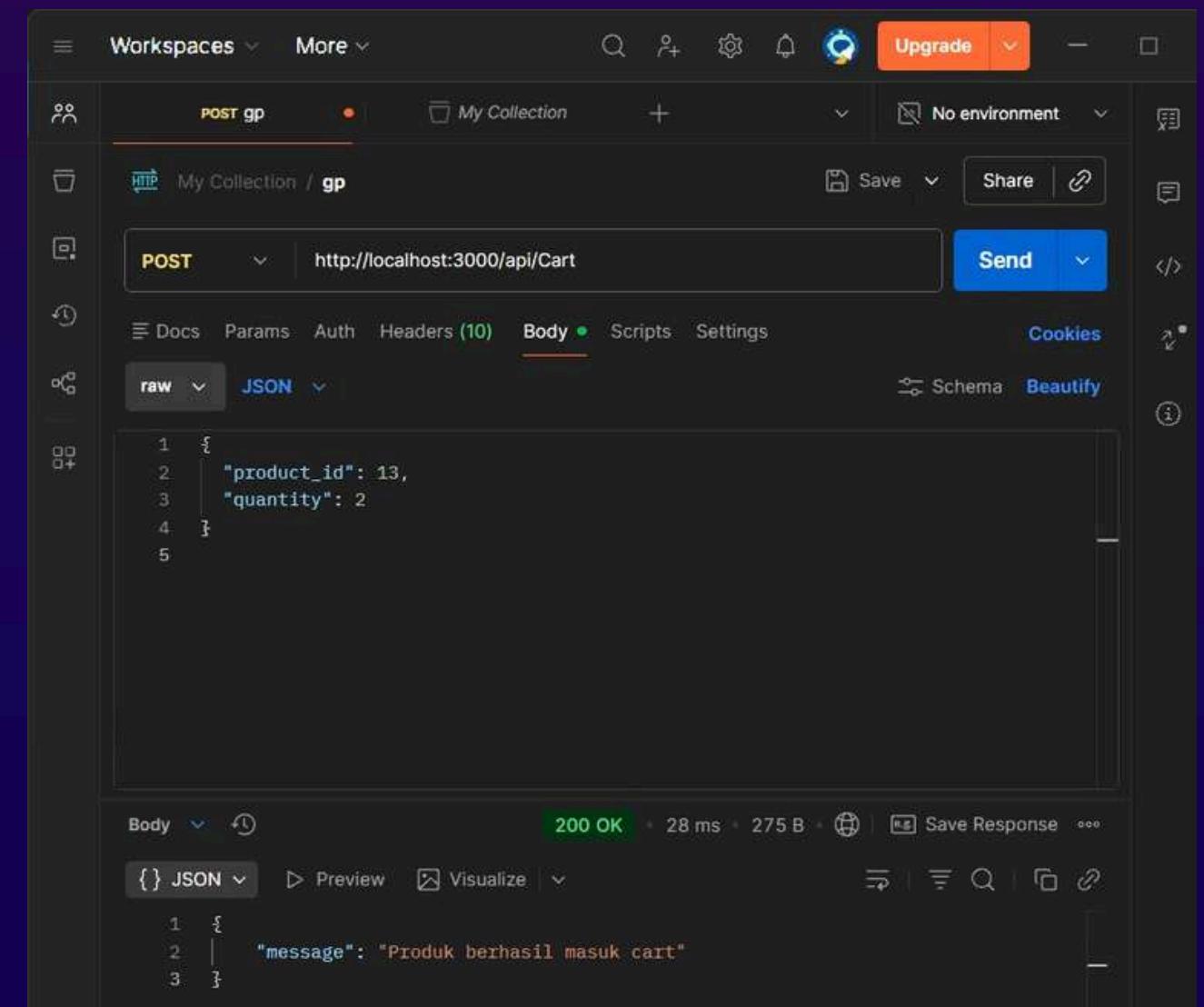
A screenshot of a REST client interface. The top bar shows 'Workspaces' and 'More'. A search icon and a profile icon are on the right. Below the bar, a collection named 'My Collection' is selected. A 'DELETE' operation is selected in the dropdown. The URL is set to 'http://localhost:3000/api/products/10'. The 'Body' tab is active, showing an empty JSON object: { }.

The response section shows a green '200 OK' status with a timestamp of '12 ms' and a size of '273 B'. The response body is a JSON object with a single key 'message': "Product berhasil dihapus".



# Post Cart

Definisi : menambahkan produk kedalam keranjang belanja user, Setiap user memiliki cart masing masing produk akan disimpan ke tabel cart atau cart\_items.



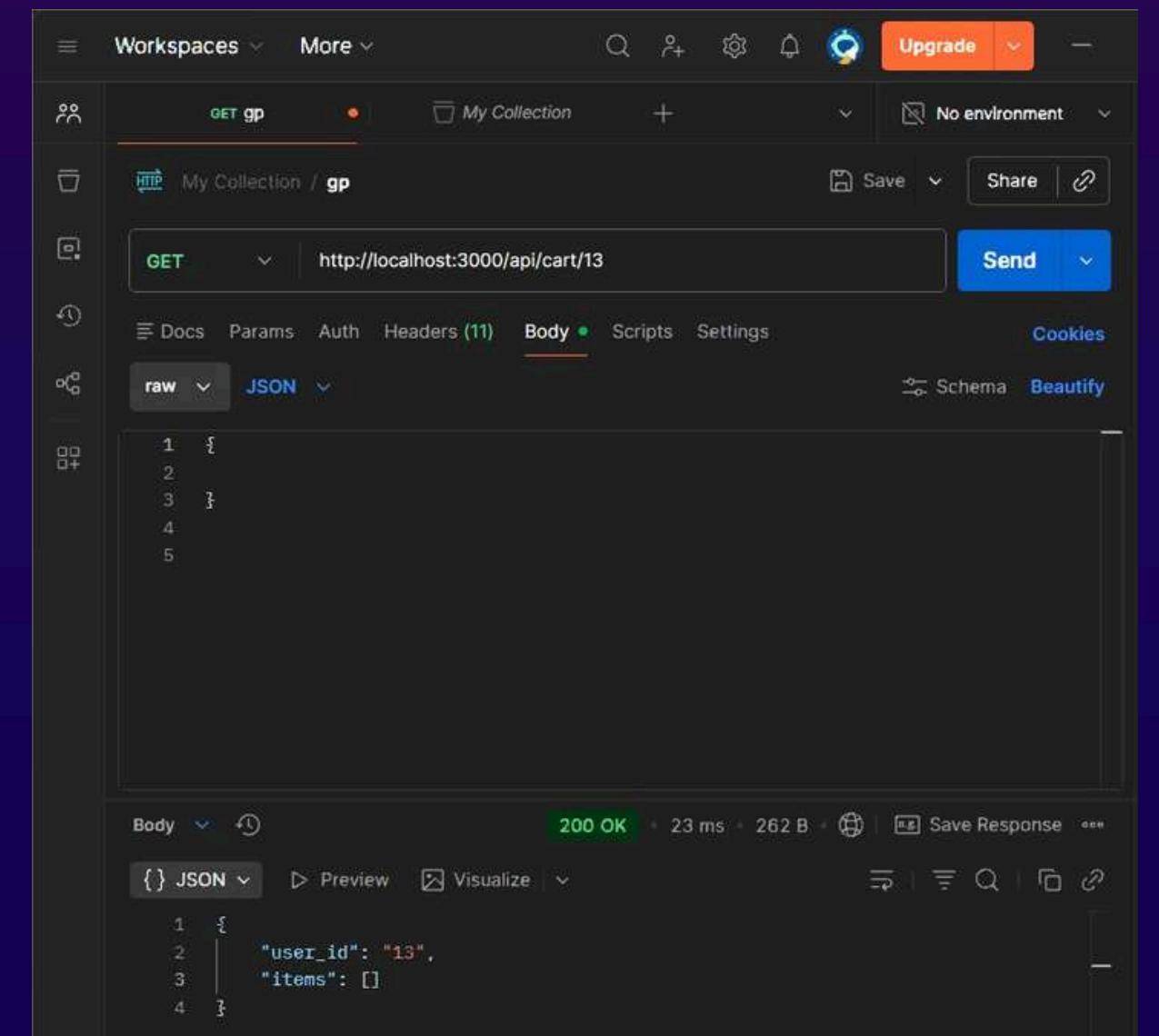
The screenshot shows a Postman collection named "POST gp". A POST request is selected with the URL "http://localhost:3000/api/Cart". The "Body" tab is active, showing a JSON payload:

```
1 {  
2   "product_id": 13,  
3   "quantity": 2  
4 }  
5
```

The response status is 200 OK, with a response body message: "Produk berhasil masuk cart".

# Get Card

Menampilkan isi keranjang belanja cart pada user dengan id tertentu



The screenshot shows a Postman workspace titled "My Collection". A collection named "gp" is selected. A GET request is defined with the URL `http://localhost:3000/api/cart/13`. The "Body" tab is selected, showing an empty JSON object: 

```
{} JSON
```

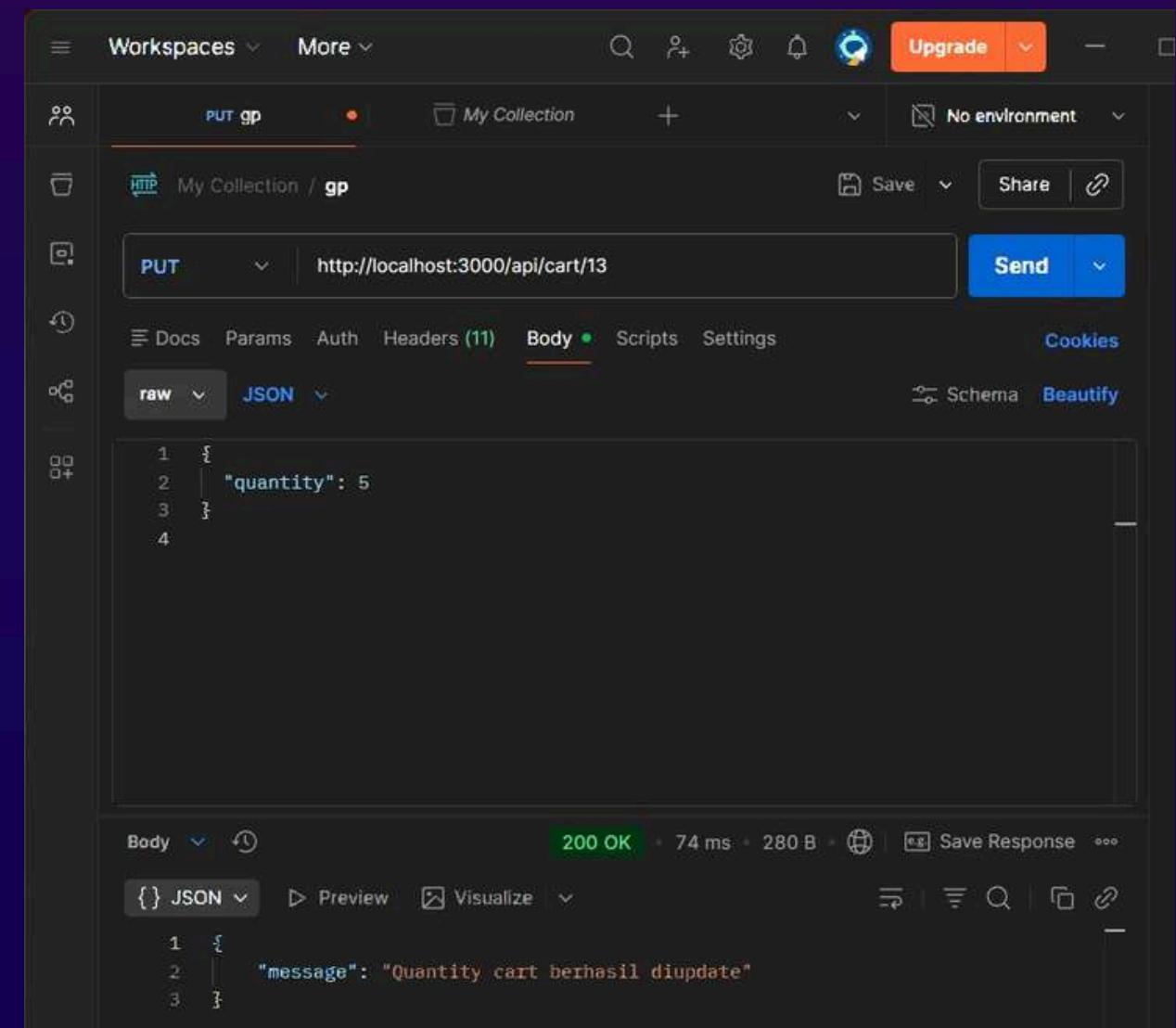
. The response section shows a 200 OK status with a response body containing: 

```
{ "user_id": "13", "items": []}
```

.

# Put Card

Digunakan untuk mengubah jumlah  
(quantity) ,menambah atau mengurangi  
produk di dalam cart

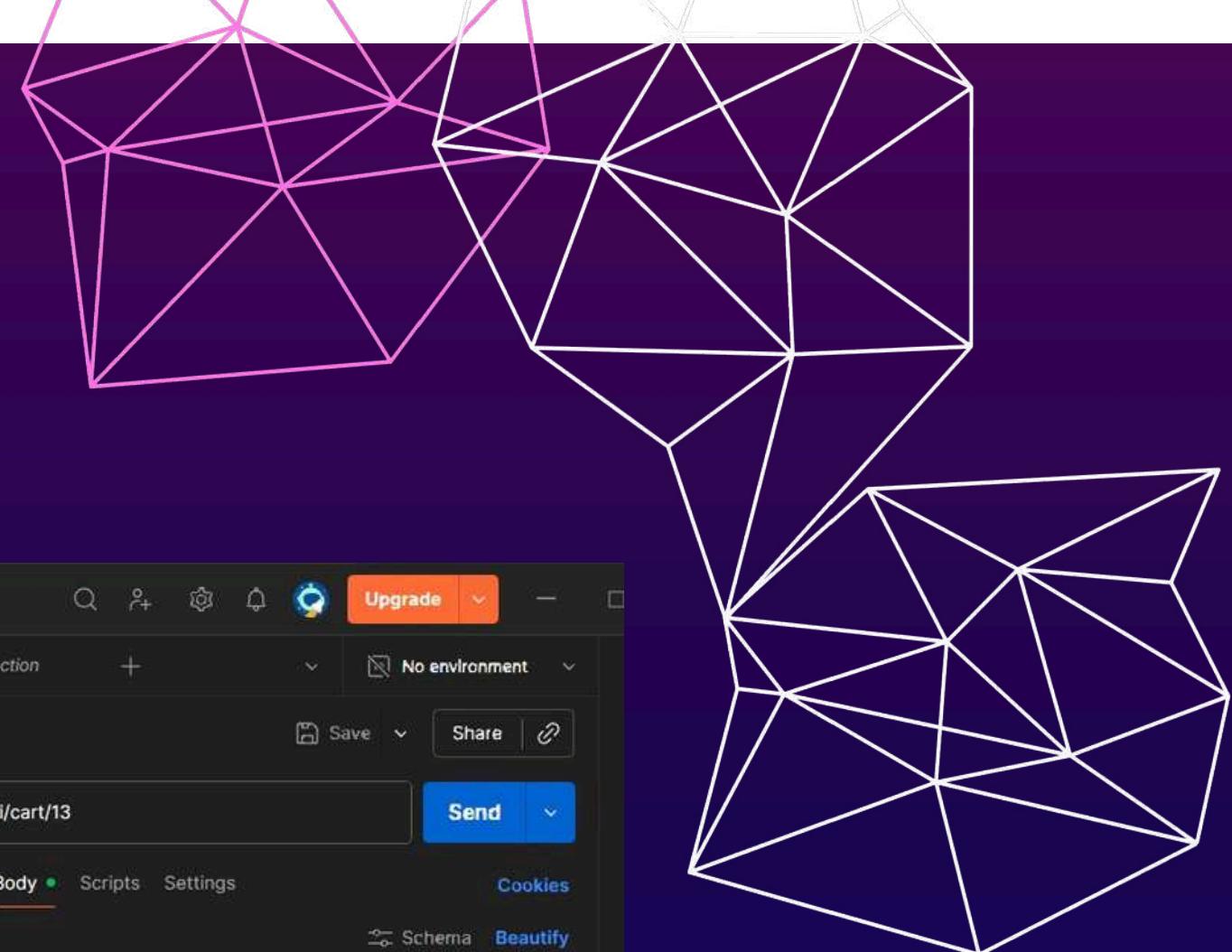


The screenshot shows a Postman interface with a dark theme. A collection named "gp" is selected. A specific request is shown for "My Collection / gp". The method is set to "PUT" and the URL is "http://localhost:3000/api/cart/13". The "Body" tab is active, showing a JSON payload:

```
1 {  
2   "quantity": 5  
3 }
```

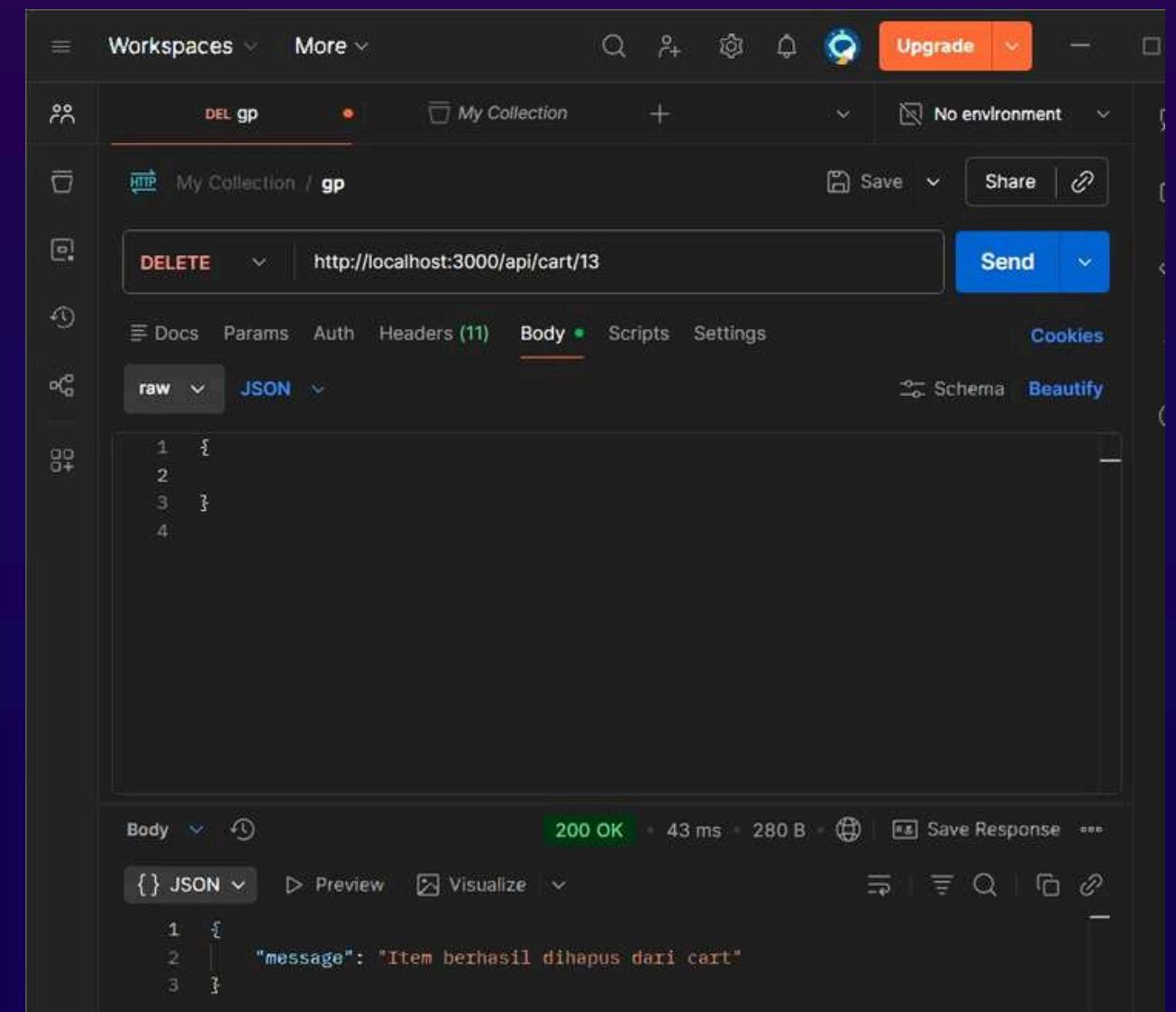
Below the request, the response is displayed as "200 OK" with a timestamp of "74 ms" and a size of "280 B". The response body is:

```
1 {  
2   "message": "Quantity cart berhasil diupdate"  
3 }
```



# Delete Card

User dapat Menghapus produk dari keranjang sebelum checkout

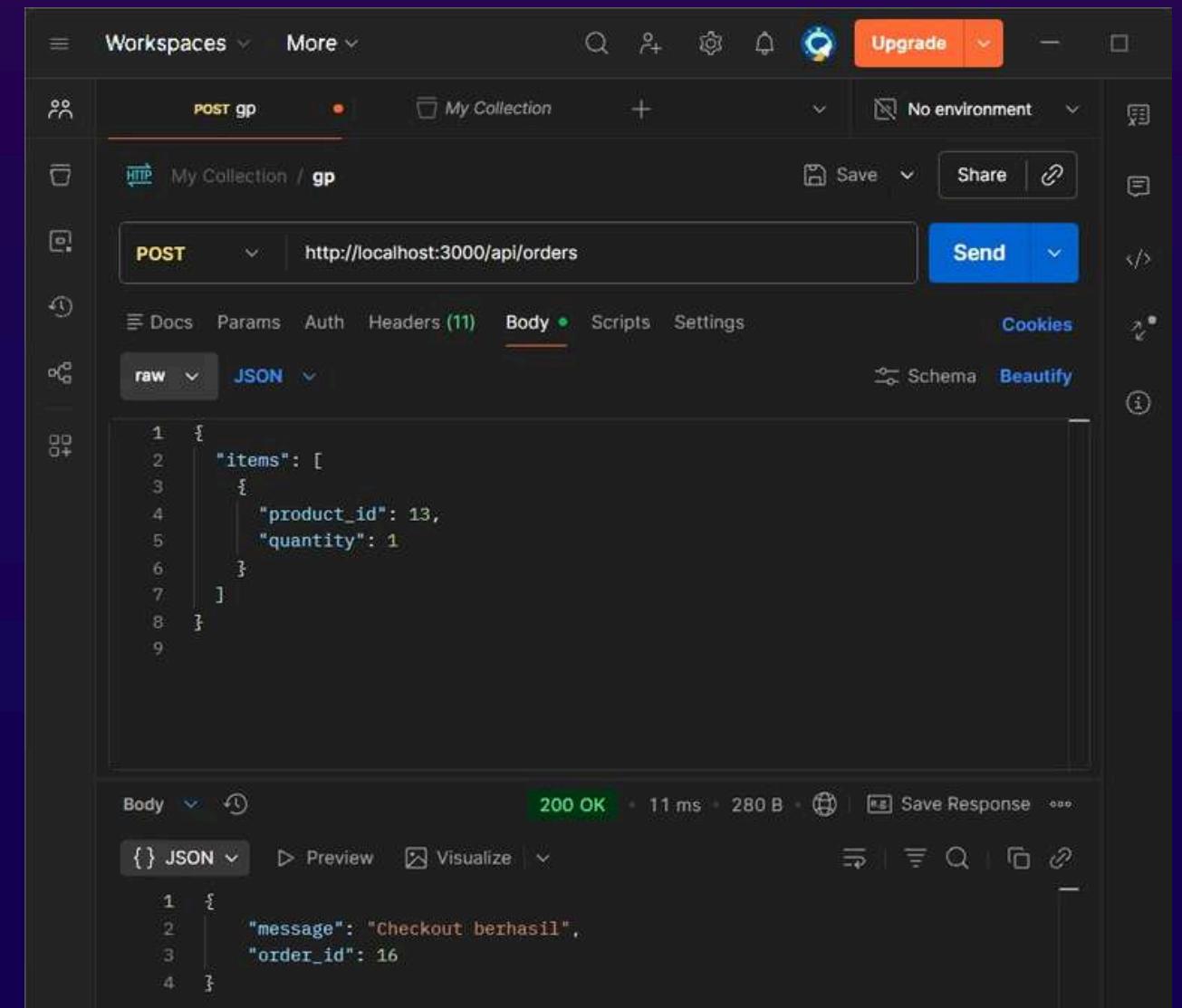


The screenshot shows a Postman collection interface. The collection is named "gp" and contains a single endpoint named "gp". The endpoint is a DELETE request to the URL `http://localhost:3000/api/cart/13`. The "Body" tab is selected, showing an empty JSON object: `{}`. The "Send" button is highlighted in blue. Below the request, the response is displayed as a 200 OK status with a response time of 43 ms and a body size of 280 B. The response body is a JSON object with a single key "message": "Item berhasil dihapus dari cart".

```
1 {  
2 }  
3 }  
4  
  
Body 200 OK 43 ms 280 B Save Response  
{ } JSON Preview Visualize  
1 {  
2 "message": "Item berhasil dihapus dari cart"  
3 }
```

# Post Orders

Digunakan untuk melakukan proses checkout, Datacart dipindahkan ke order dan order\_items, setelah checkout cart akan dikosongkan



The screenshot shows a Postman interface with a collection named "gp". A POST request is selected, pointing to the URL `http://localhost:3000/api/orders`. The request body is set to "JSON" and contains the following data:

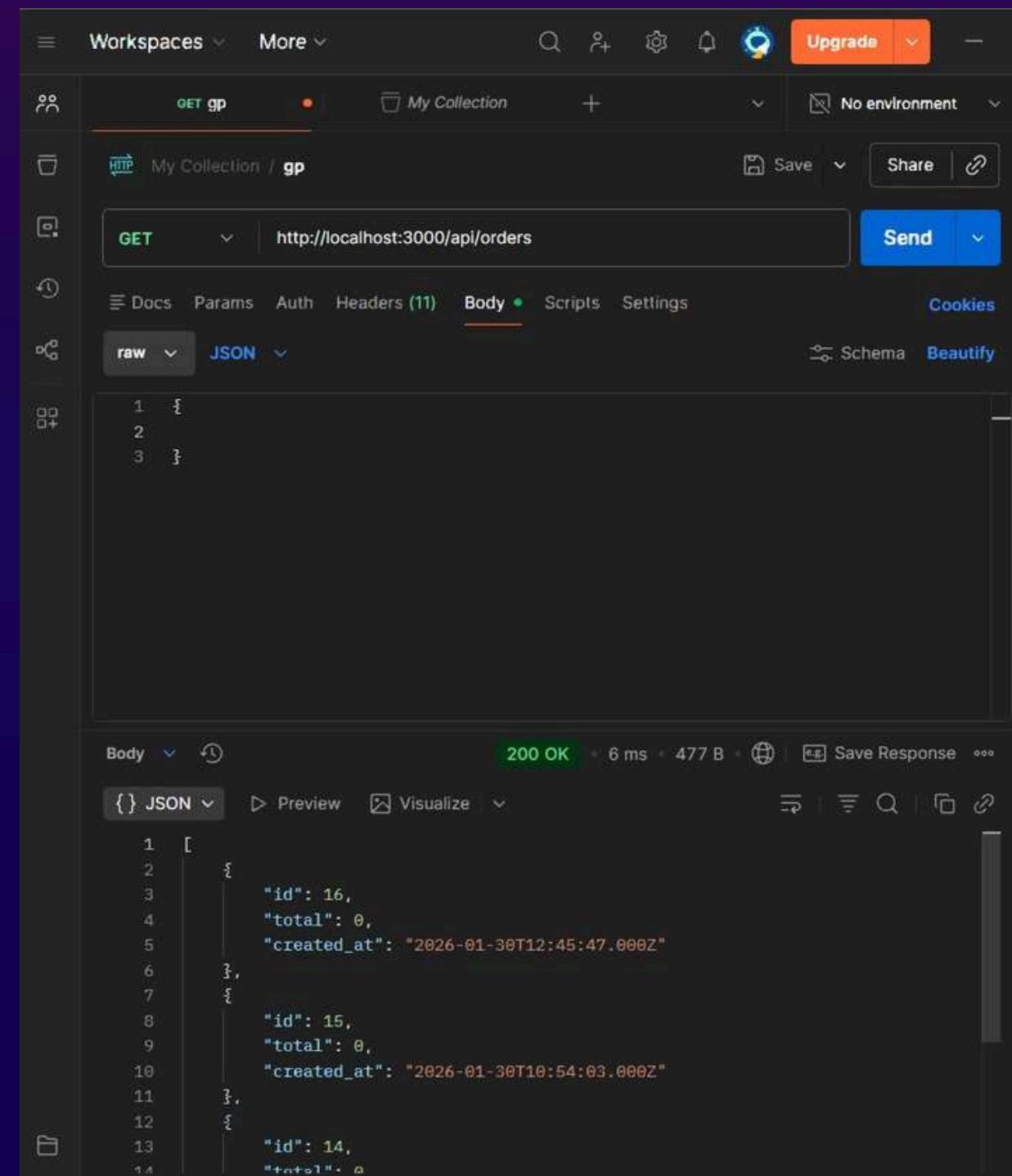
```
1 {  
2   "items": [  
3     {  
4       "product_id": 13,  
5       "quantity": 1  
6     }  
7   ]  
8 }
```

The response status is 200 OK, with a response time of 11 ms and a response size of 280 B. The response body is:

```
1 {  
2   "message": "Checkout berhasil",  
3   "order_id": 16  
4 }
```

# Get Orders

digunakan untuk mengambil riwayat pesanan user dan user hanya bisa melihat order miliknya sendiri



The screenshot shows a REST client interface with the following details:

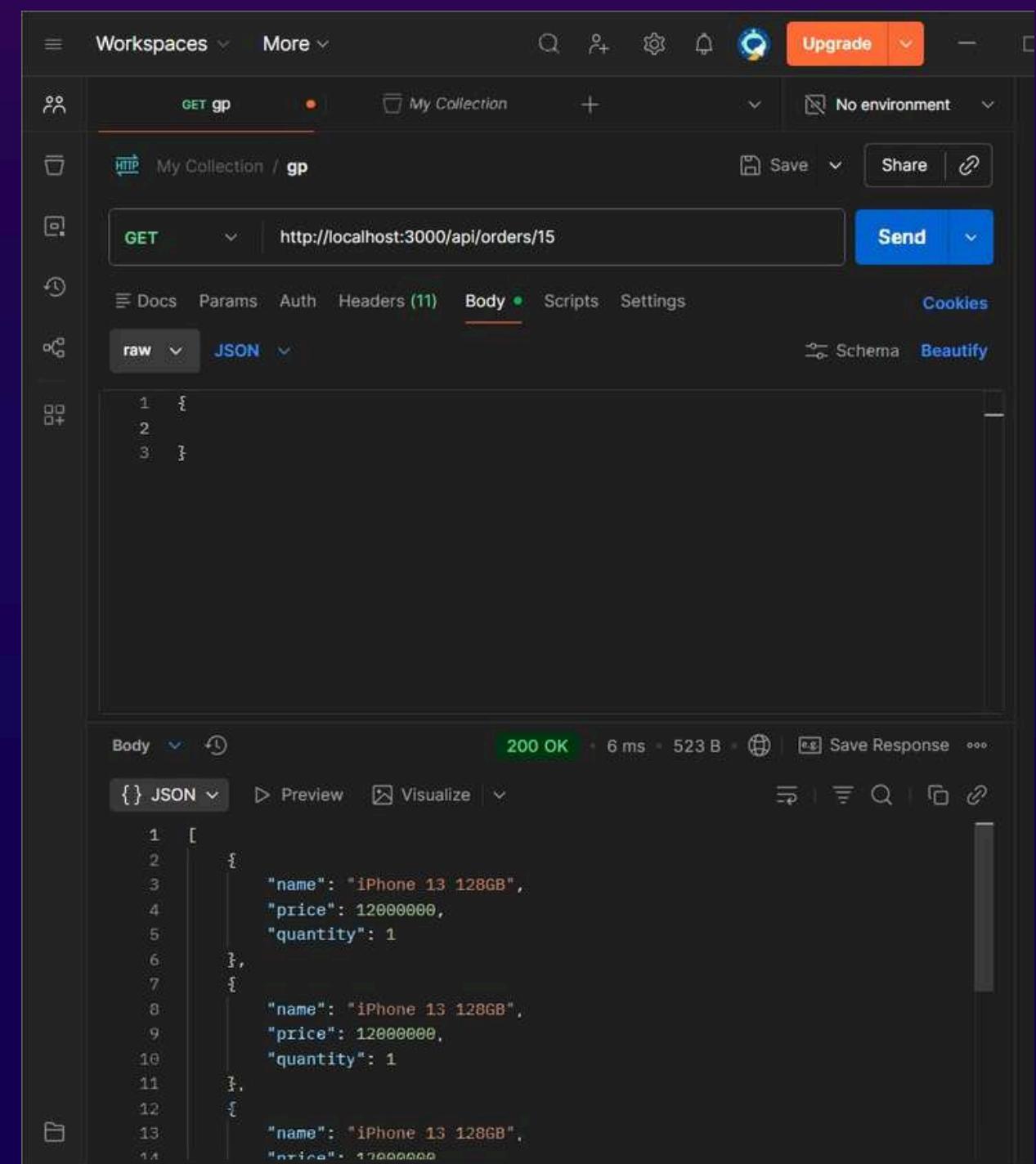
- Request URL:** GET /gp
- Method:** GET
- Endpoint:** http://localhost:3000/api/orders
- Body:** An empty JSON object: {}
- Response Status:** 200 OK
- Response Time:** 6 ms
- Response Size:** 477 B
- Response Content:** A JSON array of three order objects:

```
1: [
2:   {
3:     "id": 16,
4:     "total": 0,
5:     "created_at": "2026-01-30T12:45:47.000Z"
6:   },
7:   {
8:     "id": 15,
9:     "total": 0,
10:    "created_at": "2026-01-30T10:54:03.000Z"
11:   },
12:   {
13:     "id": 14,
14:     "total": 0
15:   }
]
```



# Get Orders/ID

Dapat mengambil detail tertentu serta menampilkan informasi order dan item yang di pesan.

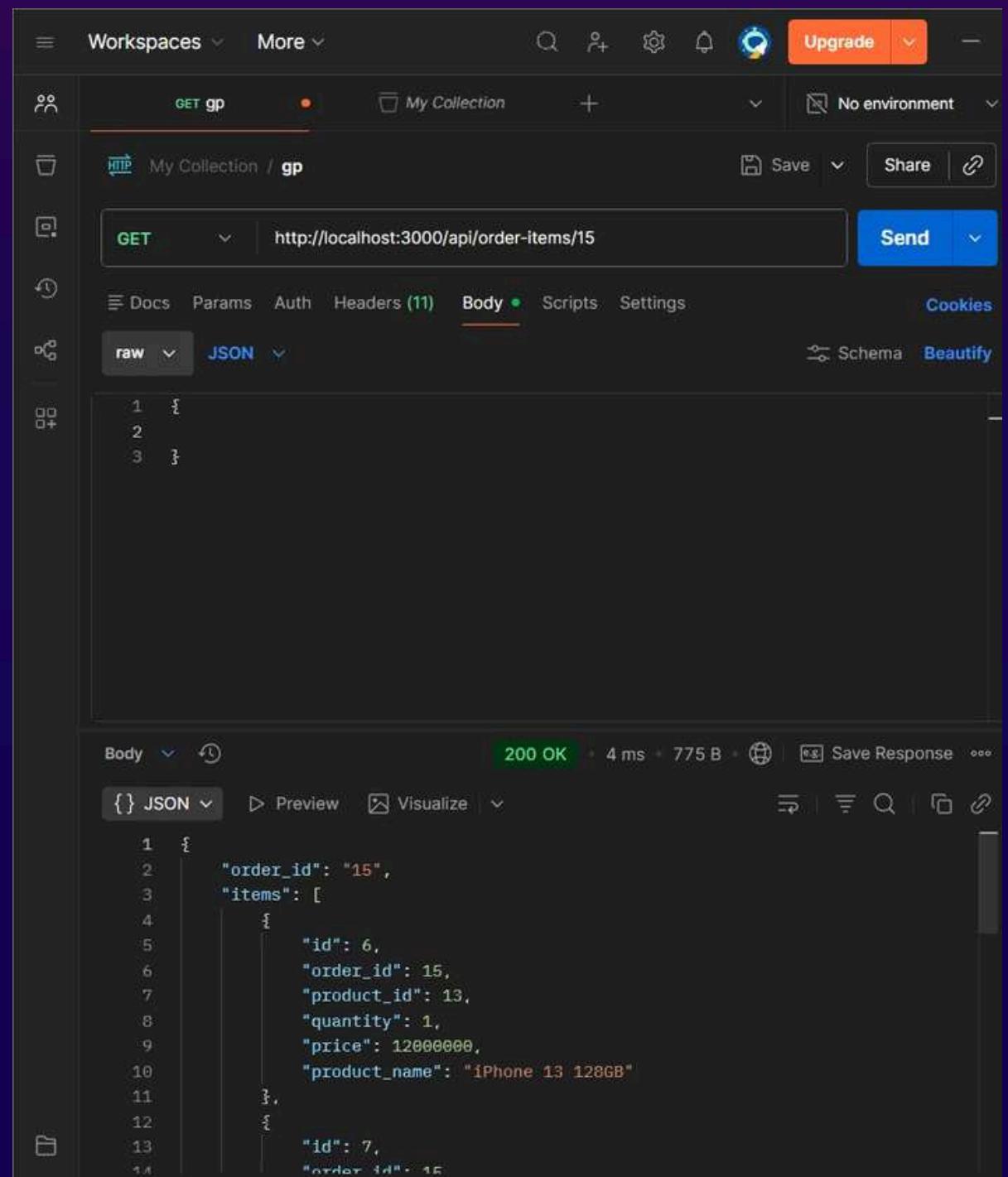


```
1 [  
2 {  
3   "name": "iPhone 13 128GB",  
4   "price": 12000000,  
5   "quantity": 1  
6 },  
7 {  
8   "name": "iPhone 13 128GB",  
9   "price": 12000000,  
10  "quantity": 1  
11 },  
12 {  
13   "name": "iPhone 13 128GB",  
14   "price": 12000000,
```



# GET /api/order-items/:orderId

berfungsi mengambil item item dari satu order tertentu dan bisa melihat produk apa saja yang ada di dalam satu pesanan

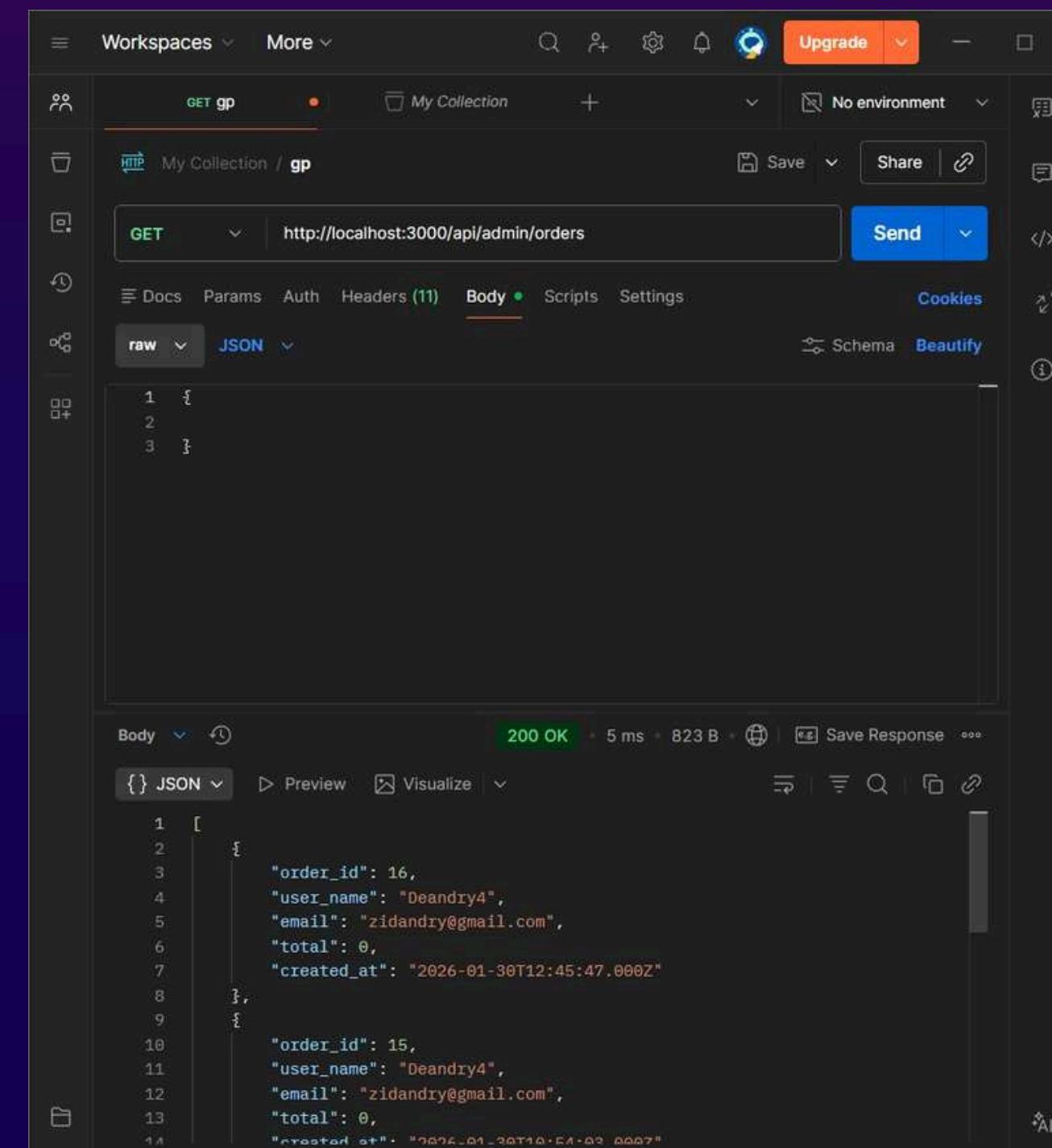


```
1 {  
2   "order_id": "15",  
3   "items": [  
4     {  
5       "id": 6,  
6       "order_id": 15,  
7       "product_id": 13,  
8       "quantity": 1,  
9       "price": 12000000,  
10      "product_name": "iPhone 13 128GB"  
11    },  
12    {  
13      "id": 7,  
14      "order_id": 15  
15    }  
16  ]  
17}
```



# GET /api/admin/orders

digunakan untuk mengambil dan menampilkan daftar semua pesanan (orders) yang ada di sistem. Endpoint ini khusus untuk admin yang memiliki akses penuh untuk melihat semua transaksi pesanan dari seluruh user.

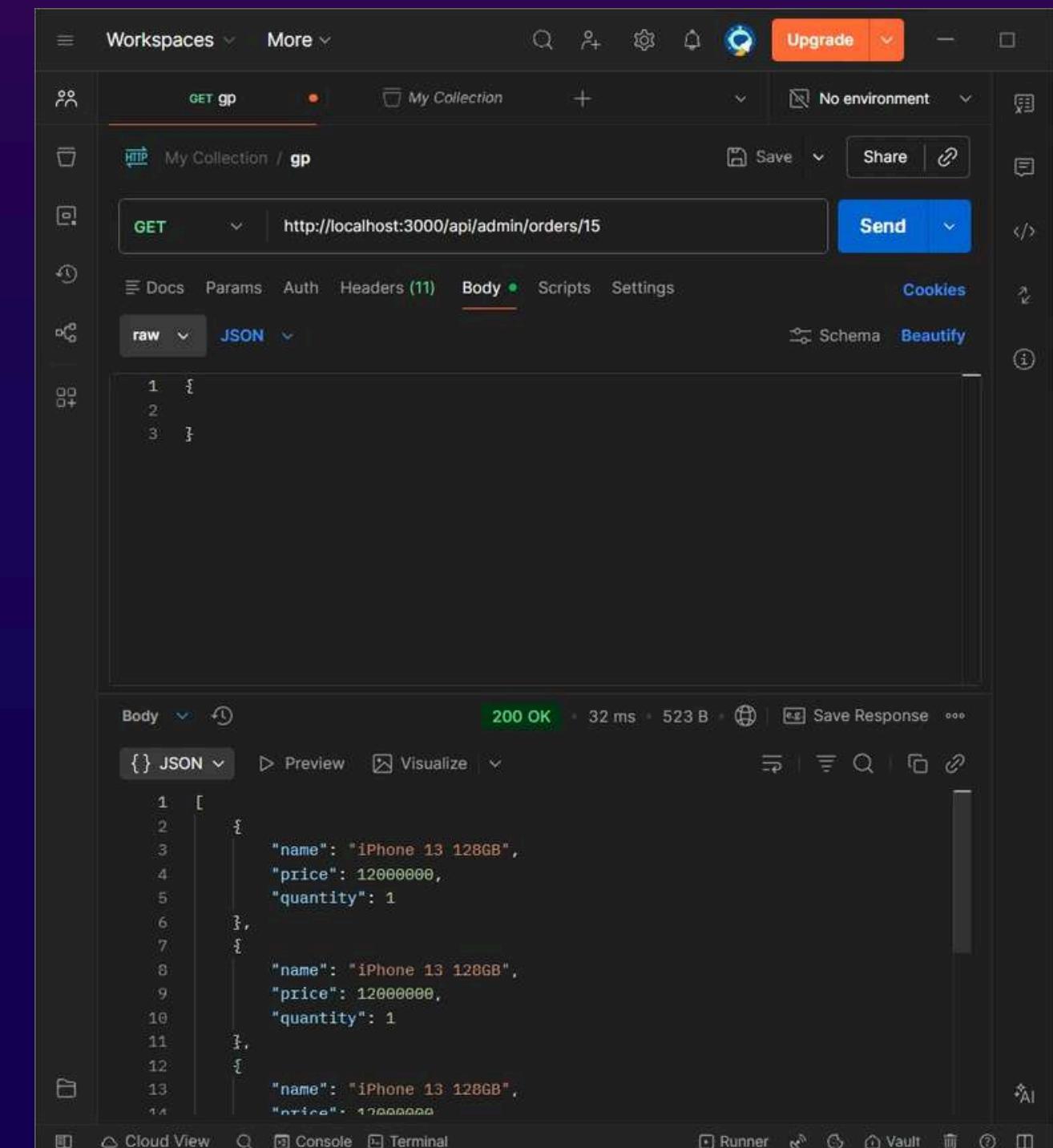


The screenshot shows a POSTMAN interface with a GET request to `http://localhost:3000/api/admin/orders`. The response is a 200 OK status with a total size of 823 B. The JSON response body contains the following data:

```
[{"order_id": 16, "user_name": "Deandry4", "email": "zidandry@gmail.com", "total": 0, "created_at": "2026-01-30T12:45:47.000Z"}, {"order_id": 15, "user_name": "Deandry4", "email": "zidandry@gmail.com", "total": 0, "created_at": "2026-01-30T12:45:43.000Z"}]
```

# GET api/admin/prders/:id

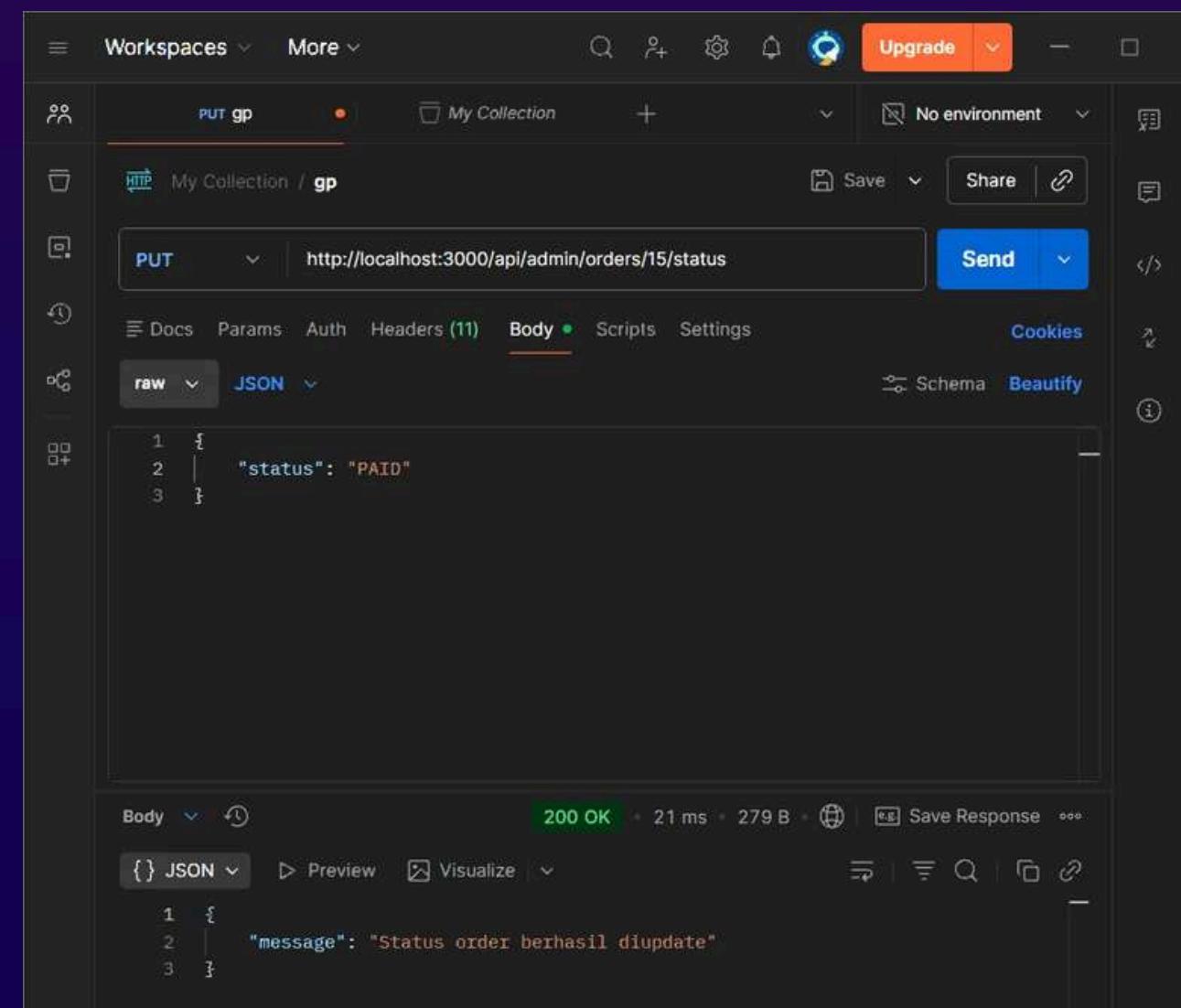
endpoint ini digunakan untuk mengambil dan menampilkan detail lengkap dari satu pesanan tertentu berdasarkan ID pesanan. Endpoint ini khusus untuk admin yang ingin melihat informasi detail dan menyeluruh tentang satu transaksi pesanan tertentu.



```
1 [  
2 {  
3   "name": "iPhone 13 128GB",  
4   "price": 12000000,  
5   "quantity": 1  
6 },  
7 {  
8   "name": "iPhone 13 128GB",  
9   "price": 12000000,  
10  "quantity": 1  
11 },  
12 {  
13   "name": "iPhone 13 128GB",  
14   "price": 12000000,
```

# PUT api/admin/orders/:id/status

admin dapat mengubah status pesanan seperti pending, paid, shipped atau completed.



The screenshot shows a POSTMAN interface with the following details:

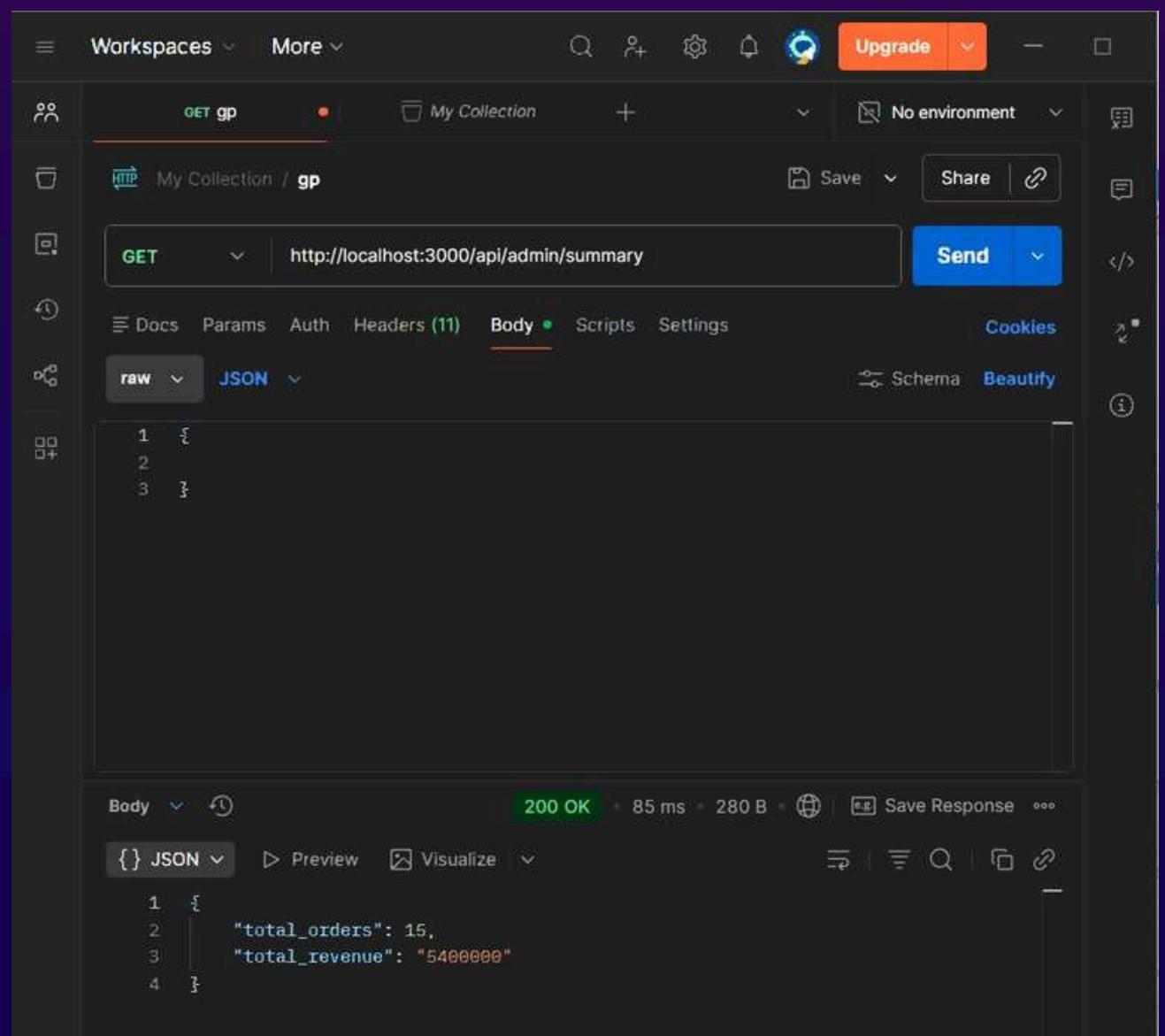
- Method:** PUT
- URL:** `http://localhost:3000/api/admin/orders/15/status`
- Body (JSON):**

```
1 {  
2   "status": "PAID"  
3 }
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {  
2   "message": "Status order berhasil diupdate"  
3 }
```

# GET api/admin/summary

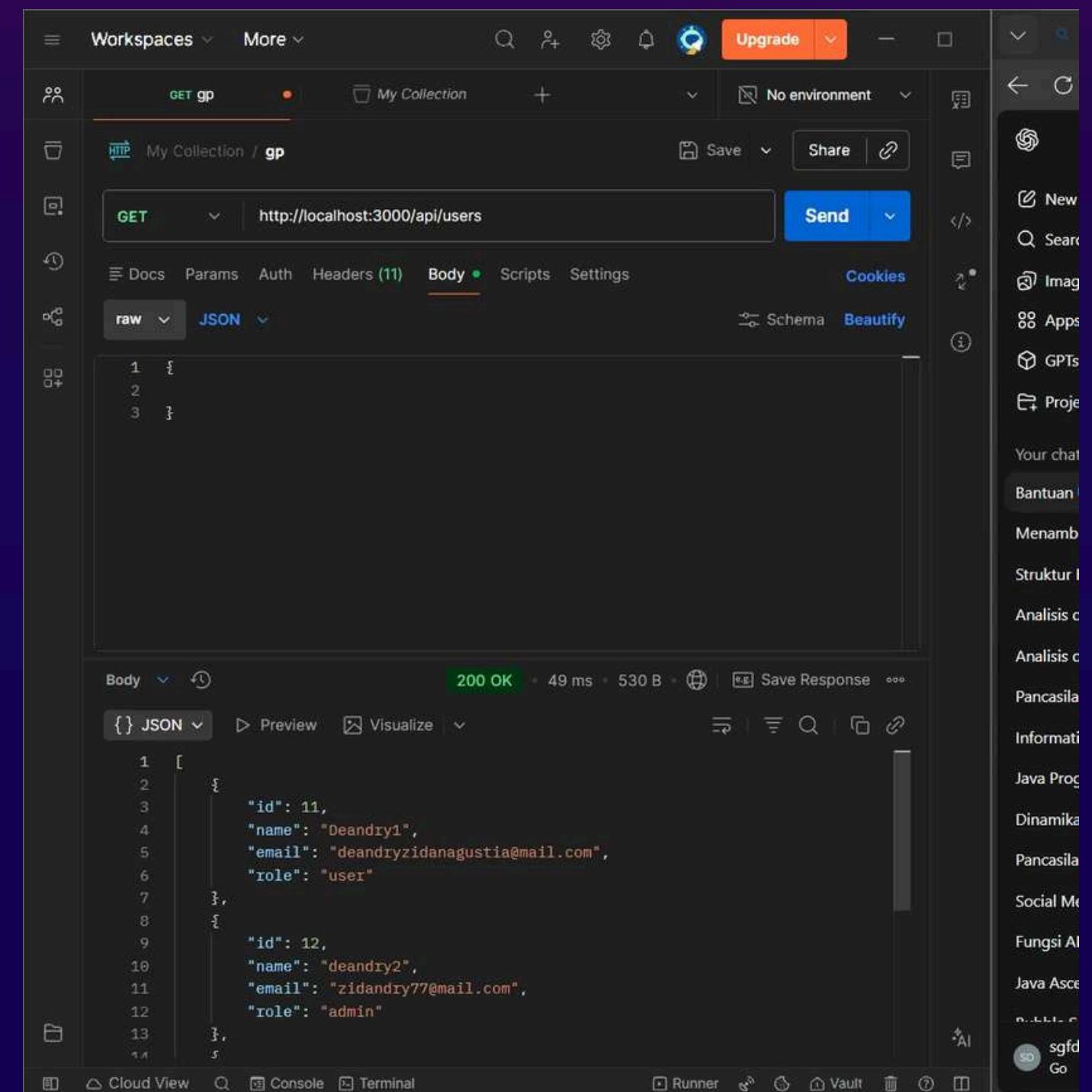
menampilkan ringkasan data sistem berisi total user, total order, total pendapatan, dan statistik penting lainnya untuk admin



```
{} JSON > Preview Visualize
1 {
2   "total_orders": 15,
3   "total_revenue": "5400000"
4 }
```

# GET api/users

Admin dapat mengambil daftar seluruh user dalam sistem dan ini hanya bisa di akses oleh admin digunakan untuk keperluan manajemen user



```
1 [  
2   {  
3     "id": 11,  
4     "name": "Deandry1",  
5     "email": "deandryzidanagustia@mail.com",  
6     "role": "user"  
7   },  
8   {  
9     "id": 12,  
10    "name": "deandry2",  
11    "email": "zidandry77@mail.com",  
12    "role": "admin"  
13  }  
14 ]
```

Terimakasih 