

LAPORAN TEST CASE FSWD 2 ADS MBKM INTERNSHIP BATCH 6

Oleh: Mochammad Zaky Zamroni

Penjelasan Kode Program

Model:

Membuat mengelola interaksi dengan database, serta mendefinisikan dan memanfaatkan relasi antar tabel untuk mempermudah manipulasi data.

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Category extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = ['name'];
13
14     public function products(){
15         return $this->hasMany(Product::class);
16     }
17 }
```

Pada model Category tersebut memiliki relasi one to many dengan model Product.

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Product extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = ['category_id', 'name', 'slug', 'price'];
13
14     public function category()
15     {
16         return $this->belongsTo(Category::class);
17     }
18
19     public function assets(){
20         return $this->hasMany(ProductAsset::class);
21     }
22 }
```

Pada model Product juga memiliki relasi one to many dengan model ProductAsset

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class ProductAsset extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = ['product_id', 'image'];
13
14     public function product()
15     {
16         return $this->belongsTo(Product::class);
17     }
18 }
19
```

Resource:

Membuat Resource untuk mengelola dan mengubah format data yang dihasilkan oleh API menjadi struktur yang lebih baik, disini saya mengubahnya menjadi format JSON.

```
<?php
1 namespace App\Http\Resources;
2
3 use Illuminate\Http\Request;
4 use Illuminate\Http\Resources\Json\JsonResource;
5
6 class ProductResource extends JsonResource
7 {
8     /**
9      * Transform the resource collection into an array.
10      *
11      * @return array<int|string, mixed>
12      */
13     public function toArray(Request $request): array
14     {
15         return [
16             'id' => $this->id,
17             'category_id' => $this->category_id,
18             'name' => $this->name,
19             'slug' => $this->slug,
20             'price' => $this->price,
21             'assets' => ProductAssetResource::collection($this->assets),
22         ];
23     }
24 }
```

```
<?php
1 namespace App\Http\Resources;
2
3 use Illuminate\Http\Request;
4 use Illuminate\Http\Resources\Json\JsonResource;
5
6 class CategoryResource extends JsonResource
7 {
8     /**
9      * Transform the resource collection into an array.
10      *
11      * @return array<int|string, mixed>
12      */
13     public function toArray(Request $request): array
14     {
15         return [
16             'id' => $this->id,
17             'name' => $this->name
18         ];
19     }
20 }
```

```
<?php
1 namespace App\Http\Resources;
2
3 use Illuminate\Http\Request;
4 use Illuminate\Http\Resources\Json\JsonResource;
5
6 class ProductAssetResource extends JsonResource
7 {
8     /**
9      * Transform the resource collection into an array.
10      *
11      * @return array<int|string, mixed>
12      */
13     public function toArray(Request $request): array
14     {
15         return [
16             'id' => $this->id,
17             'product_id' => $this->product_id,
18             'images' => $this->images
19         ];
20     }
21 }
```

Controllers:

Membuat Controllers untuk mengatur bagaimana web berlaku, seperti interaksi menerima input dari user dan memproses data.

CategoryController

```
class CategoryController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $categories = Category::all();
        return CategoryResource::collection($categories);
    }

    public function sorted(Category $categories)
    {
        $categories = Category::withCount('products')->orderByDesc('products_count')->get();
        return CategoryResource::collection($categories);
    }
}
```

Disini hanya menggunakan function index untuk menampilkan semua data category, dan function sorted untuk menampilkan data category yang telah disorting berdasarkan jumlah product dalam category tersebut.

ProductController

```
class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
        $products = Product::with('assets')->get();
        return ProductResource::collection($products);
    }

    public function indexByPriceDescending()
    {
        $products = Product::with('assets')->orderByDesc('price')->get();
        return ProductResource::collection($products);
    }
}
```

Pada ProductController juga menggunakan function index untuk menampilkan semua data product, lalu function indexByPriceDescending digunakan untuk menampilkan data product yang telah disorting berdasarkan harga dari harga termahal.

```
public function store(StoreProductRequest $request)
{
    //
    $data = $request->validated();

    $product = Product::create([
        'category_id' => $data->input('category_id'),
        'name' => $data->input('name'),
        'slug' => $data->input('slug'),
        'price' => $data->input('price'),
    ]);

    foreach ($request->file('images') as $image) {
        $pathToUploadedFile = $image->store('product_assets');

        $asset = new ProductAsset(['image' => $pathToUploadedFile]);
        $product->assets()->save($asset);
    }

    return new ProductResource($product);
}
```

Function store digunakan untuk menyimpan request/data product yang telah diinputkan user ke database.

```

public function update(UpdateProductRequest $request, Product $product)
{
    //
    $data = $request->validated();

    $slugIsUnique = Product::where('slug', $data['slug'])
        ->where('id', '!=', $product->id)
        ->doesNotExist();

    if (!$slugIsUnique) {
        return response()->json([
            'message' => 'The slug has already been taken.',
            'errors' => [
                'slug' => ['The slug has already been taken.']
            ]
        ], 422);
    }

    $product->update([
        'name' => $data['name'],
        'slug' => $data['slug'],
        'price' => $data['price'],
    ]);

    return new ProductResource($product);
}

```

Function update digunakan untuk menyimpan data yang diupdate oleh user, dengan mengharuskan slug unique karena umumnya slug digunakan untuk url masing-masing product.

```

public function destroy(Product $product)
{
    $product->assets()->delete();

    $product->delete();

    return response()->json(['message' => 'Product deleted successfully']);
}

```

Function destroy digunakan untuk menghapus produk.

```

public function addAssets(Request $request, Product $product)
{
    $image = $request->file('image');
    $pathToUploadedFile = $image->store('product_assets');
    $asset = new ProductAsset([
        'image' => $pathToUploadedFile,
    ]);

    $product->assets()->save($asset);

    return response()->json(['message' => 'Asset added successfully']);
}

public function removeAsset(ProductAsset $asset)
{
    Storage::delete($asset->image);

    $asset->delete();

    return response()->json(['message' => 'Asset deleted successfully']);
}

```

Function addAsset digunakan untuk menyimpan request/data ProductAsset yang diinputkan user, sedangkan removeAsset adalah kebalikannya, yaitu digunakan untuk menghapus data ProductAsset.

Define Routes:

Mendefine routes untuk menentukan cara URL atau permintaan HTTP tertentu akan diarahkan ke fungsi atau metode khusus dalam controller.

```
Route::apiResource('categories', CategoryController::class);
Route::get('categories2', [CategoryController::class, 'sorted'])->name('categories_sorted');

Route::apiResource('products', ProductController::class);
Route::post('products/{product}/add-asset', [ProductController::class, 'addAssets']);
Route::get('products2', [ProductController::class, 'indexByPriceDescending'])->name('products_sorted');
Route::put('products/{product}/edit', [ProductController::class, 'edit']);
Route::delete('products/{product}', [ProductController::class, 'destroy']);
Route::delete('products/assets/{asset}', [ProductController::class, 'removeAsset']);
```

Migration & Seeder:

Migration digunakan untuk membuat table pada database, sedangkan seeder digunakan untuk mengisi table pada database.

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('name');
        });

        /**
         * Reverse the migrations.
         */
        public function down(): void
        {
            Schema::dropIfExists('categories');
        }
    };
};
```

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->unsignedBigInteger('category_id');
            $table->foreign('category_id')
                ->references('id')
                ->on('categories')
                ->onDelete('cascade');
            $table->string('slug');
            $table->integer('price');
            $table->timestamps();
        });

        /**
         * Reverse the migrations.
         */
        public function down(): void
        {
            Schema::dropIfExists('products');
        }
    };
};
```

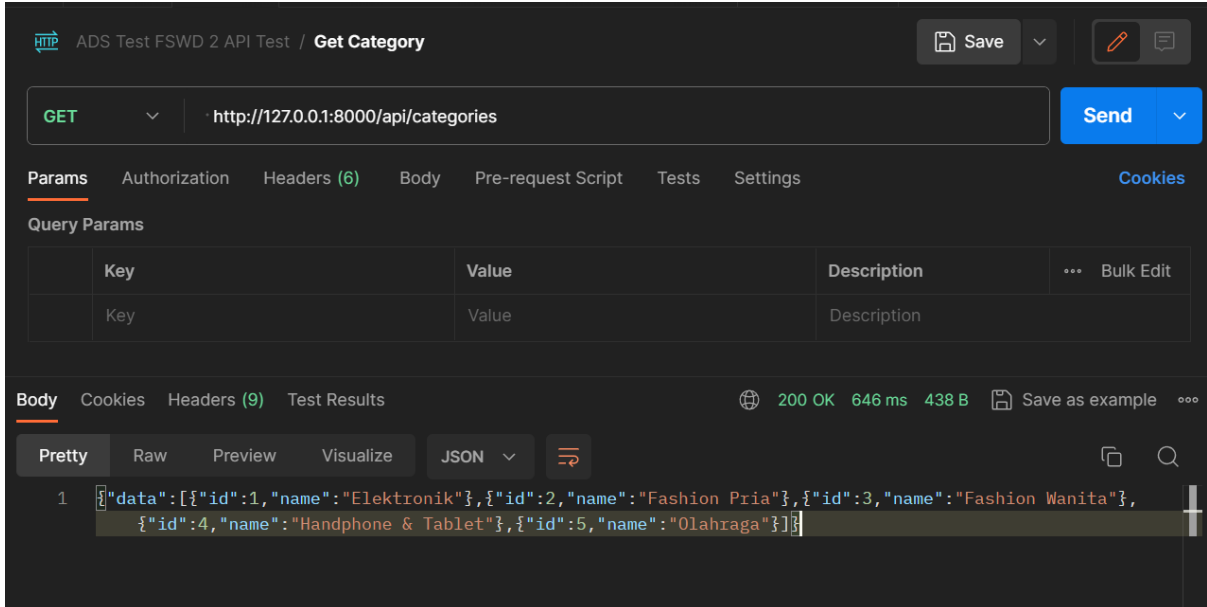
```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('product_assets', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('product_id');
            $table->foreign('product_id')
                ->references('id')
                ->on('products')
                ->onDelete('cascade');
            $table->string('image');
            $table->timestamps();
        });

        /**
         * Reverse the migrations.
         */
        public function down(): void
        {
            Schema::dropIfExists('product_assets');
        }
    };
};
```

Hasil Test Case

1. Berdasarkan sample data di atas, buatlah web dan endpoint (API) untuk menampilkan data-data:

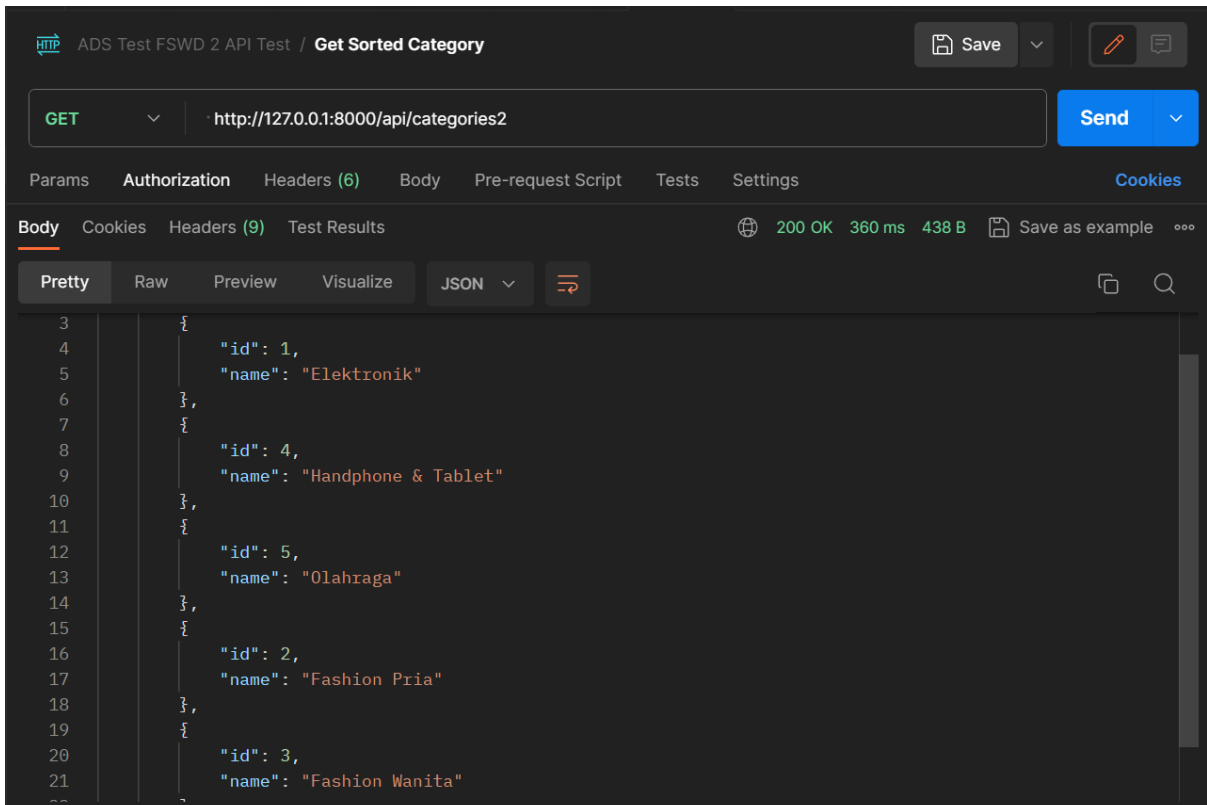
- Categories



The screenshot shows a Postman interface for a GET request to `http://127.0.0.1:8000/api/categories`. The response status is 200 OK, with a response time of 646 ms and a body size of 438 B. The response body is displayed in JSON format, showing an array of five category objects.

```
{
  "data": [
    {
      "id": 1,
      "name": "Elektronik"
    },
    {
      "id": 2,
      "name": "Fashion Pria"
    },
    {
      "id": 3,
      "name": "Fashion Wanita"
    },
    {
      "id": 4,
      "name": "Handphone & Tablet"
    },
    {
      "id": 5,
      "name": "Olahraga"
    }
  ]
}
```

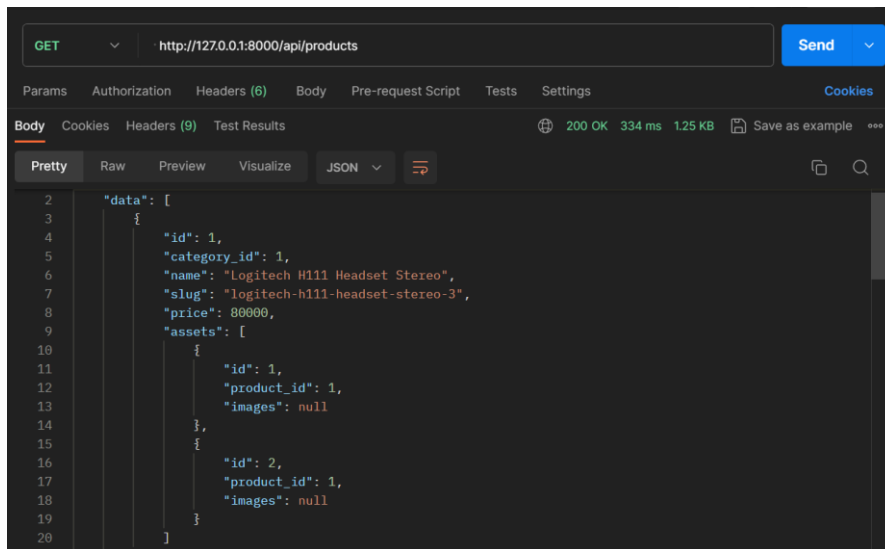
- Categories dan urutkan dari jumlah product terbanyak ke sedikit.



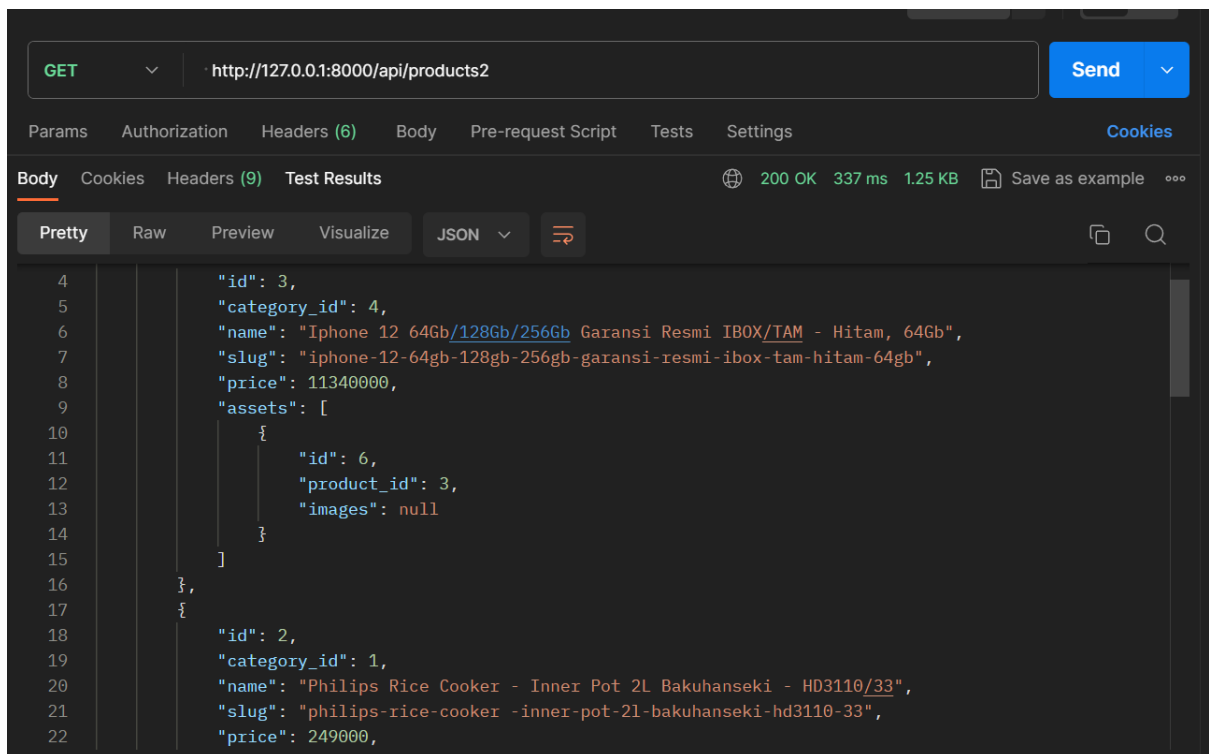
The screenshot shows a Postman interface for a GET request to `http://127.0.0.1:8000/api/categories2`. The response status is 200 OK, with a response time of 360 ms and a body size of 438 B. The response body is displayed in JSON format, showing an array of five category objects sorted by product count.

```
{
  "id": 1,
  "name": "Elektronik"
},
{
  "id": 4,
  "name": "Handphone & Tablet"
},
{
  "id": 5,
  "name": "Olahraga"
},
{
  "id": 2,
  "name": "Fashion Pria"
},
{
  "id": 3,
  "name": "Fashion Wanita"
}
```

- Products (beserta assetnya)



- Products dengan urutan dari harga termahal ke termurah

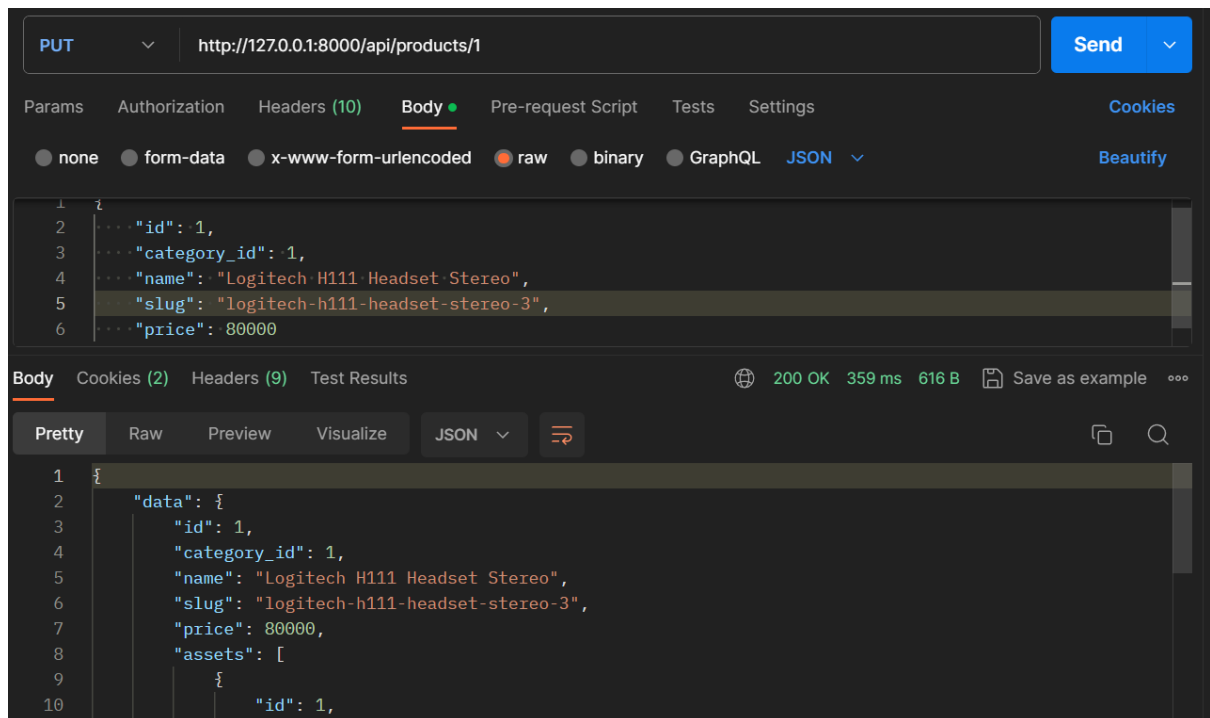


Sebenarnya ada product yang berharga 350.000 tapi terlanjur saya hapus.

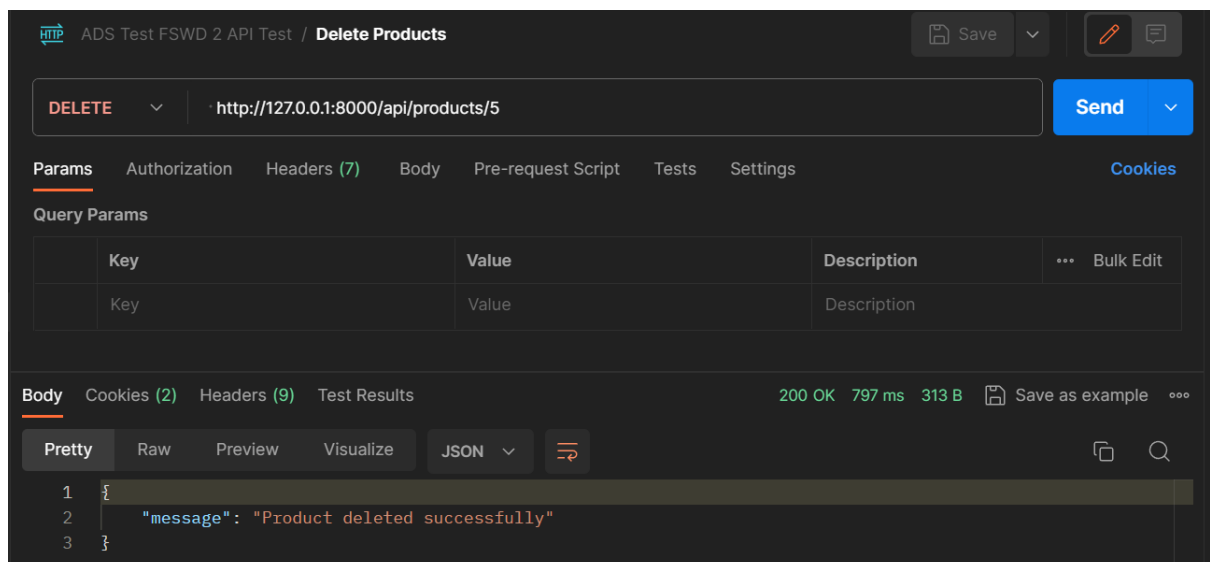
1. Buatlah endpoint untuk menambahkan data product berserta upload multiple image (ke table product_assets).

Mohon maaf karena screenshotan hilang dan waktu tidak mencukupi maka tidak bisa menyertakan ssnya.

2. Buatlah endpoint untuk edit product



3. Buatlah endpoint untuk hapus product beserta assetnya.



4. Buatlah endpoint untuk tambah asset

HTTP ADS Test FSWD 2 API Test / **Add Asset** Save

POST ▼ http://127.0.0.1:8000/api/products/2/add-asset Send ▼

Params Authorization Headers (10) **Body** • Pre-request Script Tests Settings Cookies

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

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	image	File ▼ logo wri.png 			
	Key	Text ▼ Value	Description		

Body Cookies (2) Headers (9) Test Results 200 OK 751 ms 309 B Save as example ...

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "message": "Asset added successfully"
3 }
```

5. Buatlah endpoint untuk hapus asset

HTTP ADS Test FSWD 2 API Test / **Delete Asset** Save

DELETE ▼ http://127.0.0.1:8000/api/products/assets/10 Send ▼

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies (2) Headers (9) Test Results 200 OK 603 ms 311 B Save as example ...

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "message": "Asset deleted successfully"
3 }
```