DOKUMENTASI TEST BACK-END DEVELOPER INSYST

Nama: Yabes Edward Sihombing
Tanggal: 22-24 November 2025
Project: insyst-backend

1. PENDAHULUAN

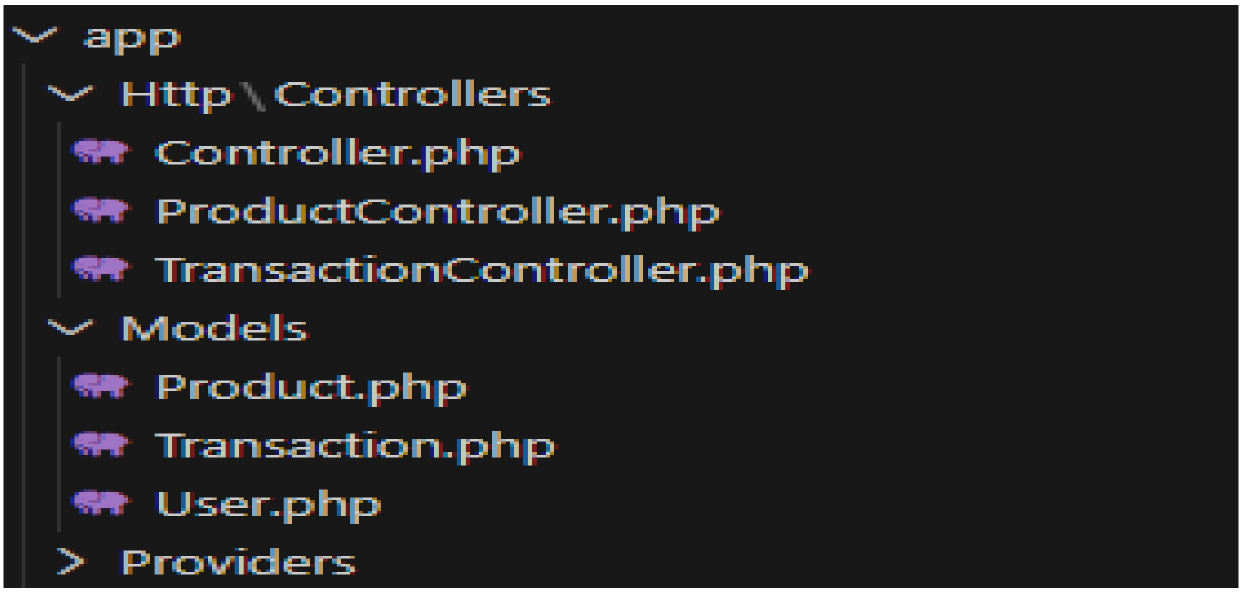
Dokumentasi ini berisi penjelasan implementasi RESTful API untuk manajemen produk dan transaksi pembelian menggunakan Laravel. API ini dibuat untuk menilai kemampuan logika backend, struktur kode, dan pemahaman framework Laravel.

2. STRUKTUR FILE & FOLDER

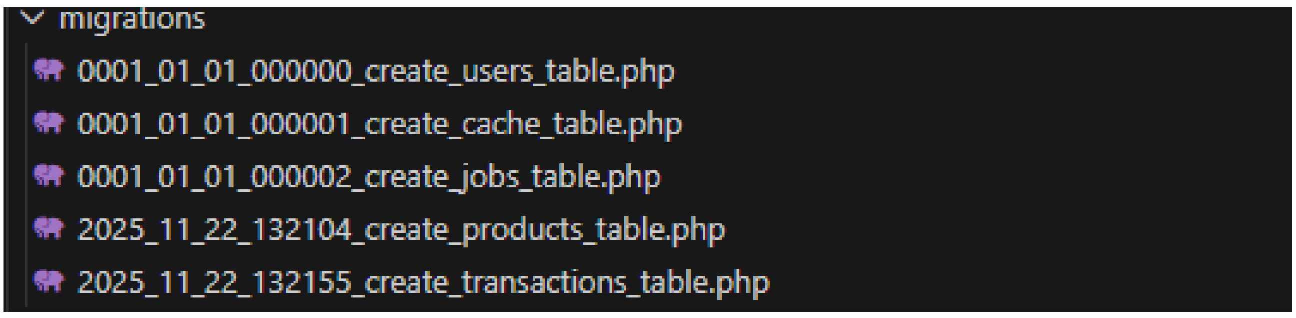
Project ini menggunakan struktur Laravel 11 dengan beberapa file yang dibuat sesuai requirement:

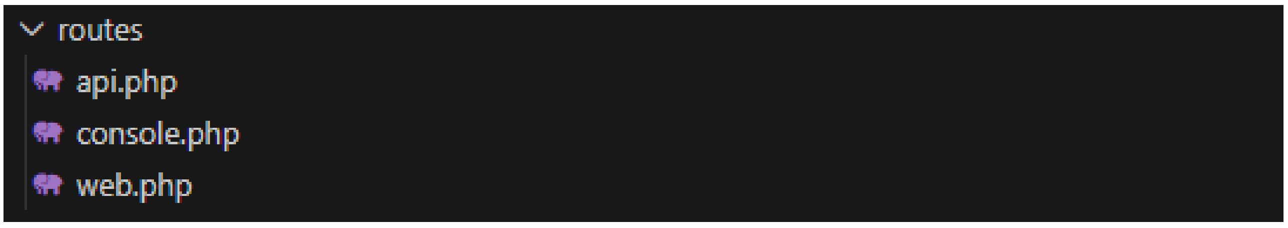
File-file yang dibuat:

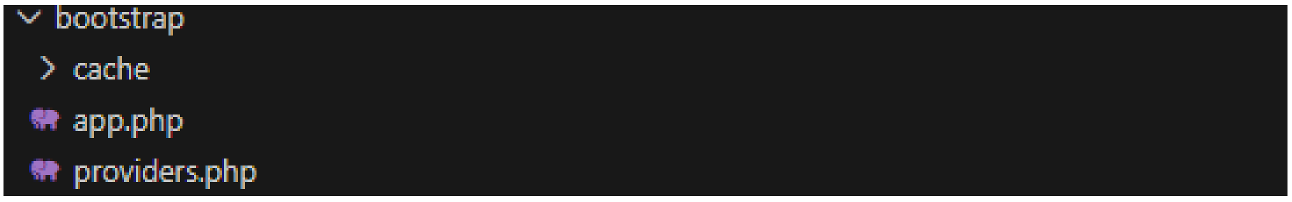
- app/Models/Product.php - Model untuk tabel products
- app/Models/Transaction.php - Model untuk tabel transactions
- app/Http/Controllers/ProductController.php - Controller untuk CRUD products
- app/Http/Controllers/TransactionController.php - Controller untuk transaksi
- database/migrations/xxxx_create_products_table.php - Migration tabel products
- database/migrations/xxxx_create_transactions_table.php - Migration tabel transactions
- routes/api.php - Route API endpoints
- bootstrap/app.php - Konfigurasi route API



```
∨ app
  ∨ Http \ Controllers
      Controller.php
      ProductController.php
      TransactionController.php
  ∨ Models
      Product.php
      Transaction.php
      User.php
  > Providers
```


```
migrations
  0001_01_01_000000_create_users_table.php
  0001_01_01_000001_create_cache_table.php
  0001_01_01_000002_create_jobs_table.php
  2025_11_22_132104_create_products_table.php
  2025_11_22_132155_create_transactions_table.php
```


```
routes
  api.php
  console.php
  web.php
```


```
bootstrap
  > cache
  app.php
  providers.php
```

3. PENJELASAN STRUKTUR DATABASE

3.1. Tabel: products

Kolom :

Kolom	Tipe Data	Keterangan
Id	Bigint	Primary Key (Auto Increment)
Name	String	Nama produk
Price	Integer	Harga produk
Stock	Integer	Jumlah stok tersedia
Created_at	Timestamp	Waktu dibuat (otomatis)
Updated_at	Timestamp	Waktu diupdate (otomatis)
Deleted_at	timestamp	Waktu dihapus (soft delete)

Fitur:

- Menggunakan SoftDeletes untuk soft delete
- Primary key auto increment

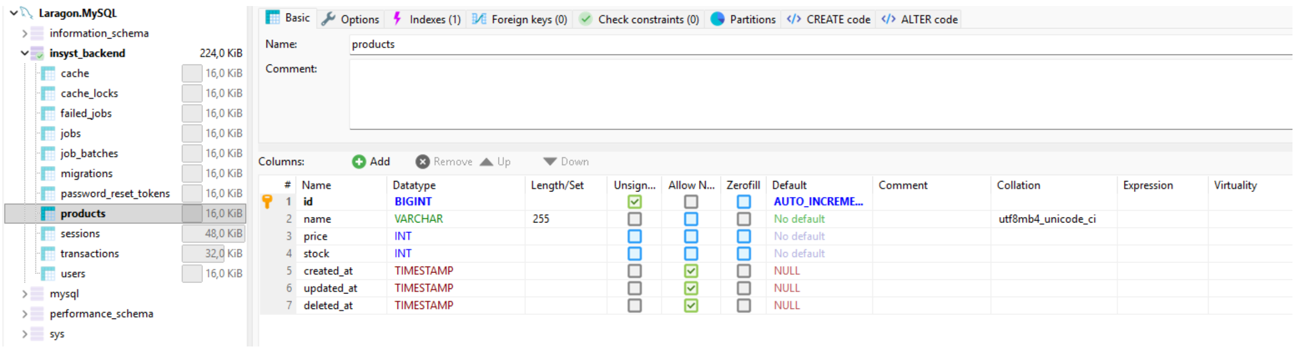
3.2. Tabel: transactions

Kolom :

Kolom	Tipe Data	Keterangan
Id	Bigint	Primary Key (Auto Increment)
Product_id	Bigint	Foreign key ke products.id
Quantity	Integer	Jumlah produk yang dibeli
Total_price	Integer	Total harga (quantity x price)
Created_at	Timestamp	Waktu dibuat (otomatis)
Updated_at	Timestamp	Waktu diupdate (otomatis)
Deleted_at	timestamp	Waktu dihapus (soft delete)

Relasi:

- Relasi foreign key ke tabel products dengan cascade delete
- Jika produk dihapus, transaksi terkait juga terhapus



Name: products

Comment:

Columns:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Comment	Collation	Expression	Virtuality
1	id	BIGINT		✓			AUTO_INCREME...				
2	name	VARCHAR	255				No default		utf8mb4_unicode_ci		
3	price	INT					No default				
4	stock	INT					No default				
5	created_at	TIMESTAMP			✓		NULL				
6	updated_at	TIMESTAMP			✓		NULL				
7	deleted_at	TIMESTAMP			✓		NULL				

4. PENJELASAN FILE FILE UTAMA

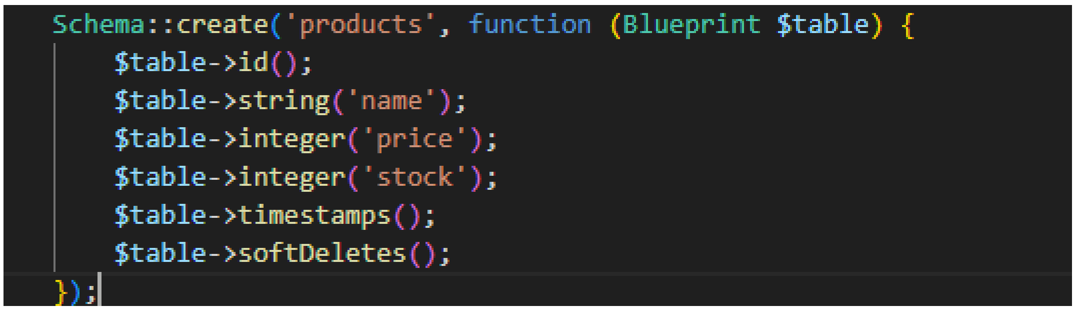
4.1. Migration:create_products_table.php

Lokasi: database/migrations/xxxx_create_products_table.php

Fungsi:

- Membuat tabel products di database
- Mendefinisikan struktur kolom tabel
- Method up() untuk membuat tabel
- Method down() untuk menghapus tabel
- Menggunakan softDeletes() untuk soft delete

Kode penting:



```php
Schema::create('products', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->integer('price');
    $table->integer('stock');
    $table->timestamps();
    $table->softDeletes();
});
```

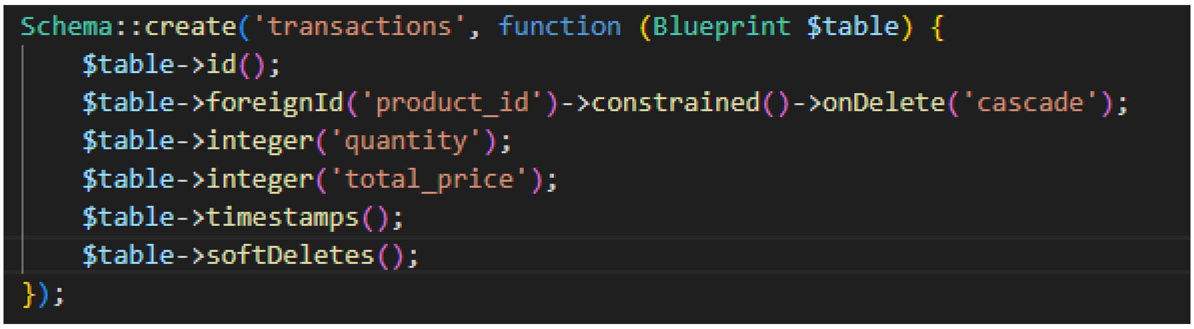
4.2. Migration:create_transactions_table.php

Lokasi: database/migrations/xxxx_create_transactions_table.php

Fungsi:

- Membuat tabel transactions di database
- Mendefinisikan relasi foreign key ke products
- Method up() untuk membuat tabel
- Method down() untuk menghapus tabel

Kode penting:



```php
Schema::create('transactions', function (Blueprint $table) {
    $table->id();
    $table->foreignId('product_id')->constrained()->onDelete('cascade');
    $table->integer('quantity');
    $table->integer('total_price');
    $table->timestamps();
    $table->softDeletes();
});
```

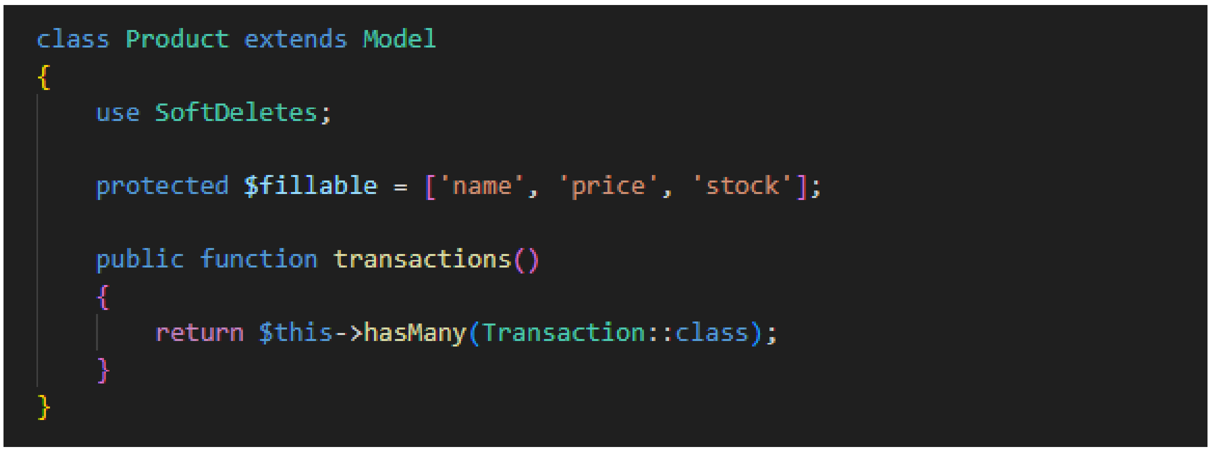
4.3. Model: Product.php

Lokasi: app/Models/Product.php

Fungsi:

- Merepresentasikan tabel products
- Mendefinisikan kolom yang bisa diisi mass assignment
- Relasi hasMany ke Transaction
- Menggunakan trait SoftDeletes

Kode:



```
class Product extends Model
{
    use SoftDeletes;

    protected $fillable = ['name', 'price', 'stock'];

    public function transactions()
    {
        return $this->hasMany(Transaction::class);
    }
}
```

Penjelasan:

- $fillable - Kolom yang boleh diisi secara mass assignment
- transactions() - Relasi one-to-many ke tabel transactions

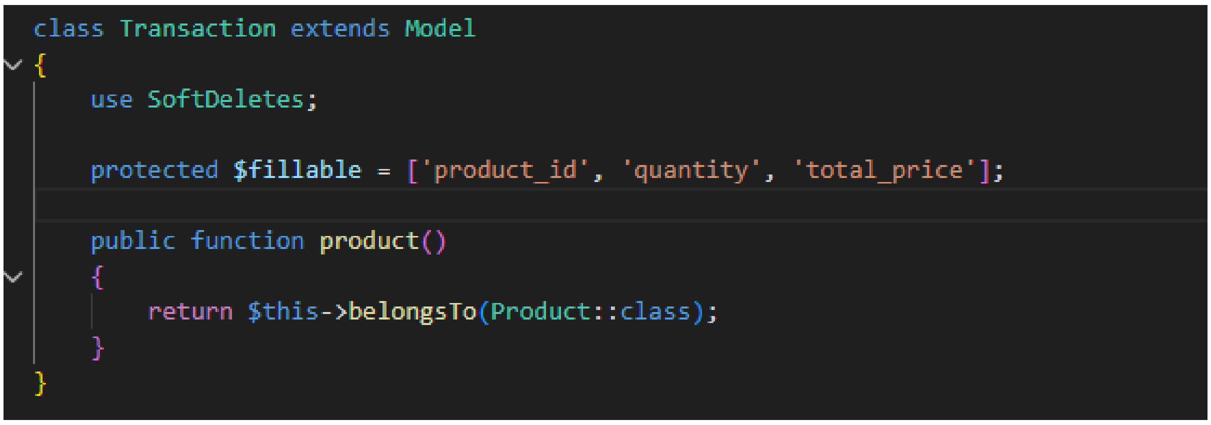
4.4. Model : Transaction.php

Lokasi: app/Models/Transaction.php

Fungsi:

- Merepresentasikan tabel transactions
- Mendefinisikan kolom yang bisa diisi mass assignment
- Relasi belongsTo ke Product
- Menggunakan trait SoftDeletes

Kode:



```
class Transaction extends Model
{
    use SoftDeletes;

    protected $fillable = ['product_id', 'quantity', 'total_price'];

    public function product()
    {
        return $this->belongsTo(Product::class);
    }
}
```

Penjelasan:

- product() - Relasi many-to-one ke tabel products

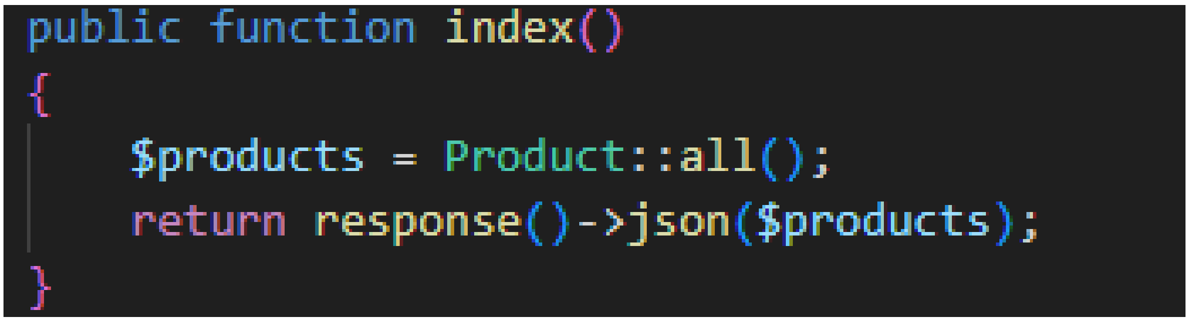
4.5. Controller: ProductController.php

Lokasi: app/Http/Controllers/ProductController.php

Fungsi:

- Menangani semua operasi CRUD untuk produk
- Method index() - Mengambil semua produk
- Method store() - Menambah produk baru dengan validasi
- Method update() - Mengupdate produk dengan validasi
- Method destroy() - Menghapus produk (soft delete)

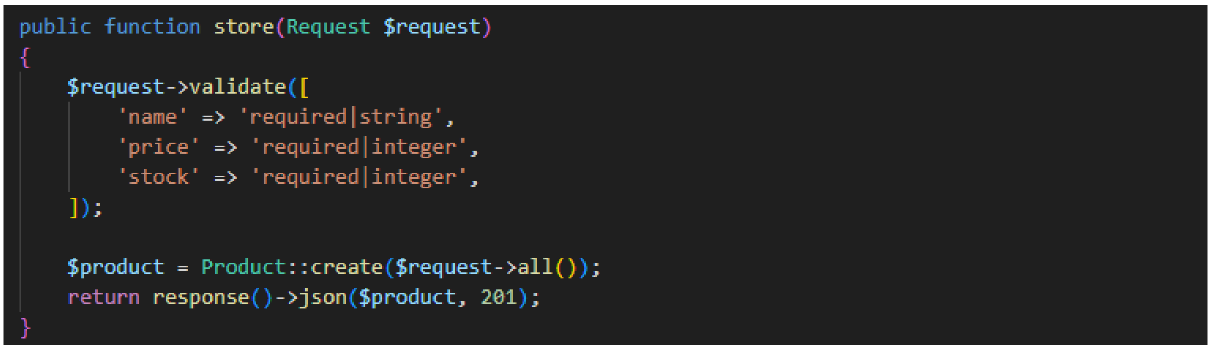
Method 1: index() - GET /api/products



```php
public function index()
{
    $products = Product::all();
    return response()->json($products);
}
```

Fungsi: Mengambil semua data produk dan return dalam format JSON

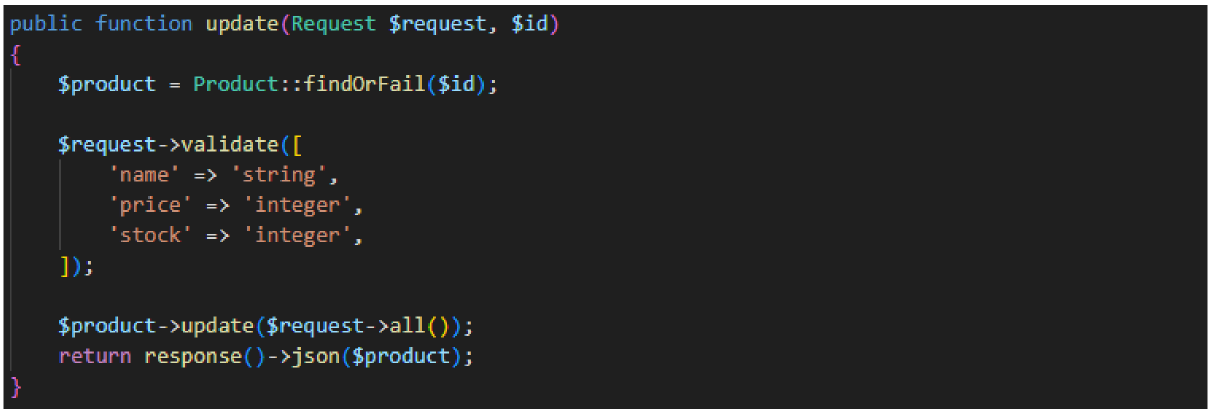
Method 2: store() - POST /api/products



```php
public function store(Request $request)
{
    $request->validate([
        'name' => 'required|string',
        'price' => 'required|integer',
        'stock' => 'required|integer',
    ]);

    $product = Product::create($request->all());
    return response()->json($product, 201);
}
```

Fungsi: Validasi input dan simpan produk baru Validasi: name, price, dan stock wajib diisi

Method 3: update() - PUT /api/products/{id}

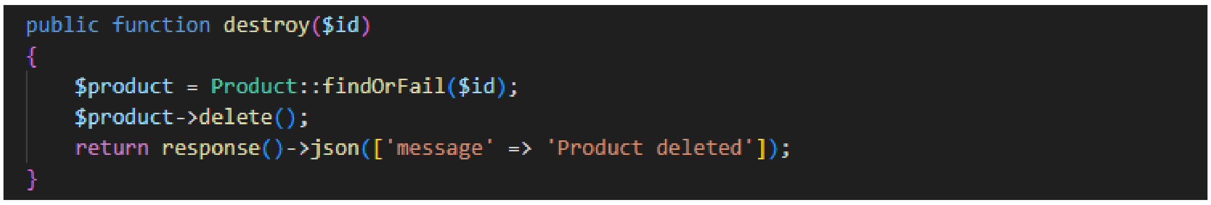


```php
public function update(Request $request, $id)
{
    $product = Product::findOrFail($id);

    $request->validate([
        'name' => 'string',
        'price' => 'integer',
        'stock' => 'integer',
    ]);

    $product->update($request->all());
    return response()->json($product);
}
```

Fungsi: Mencari produk berdasarkan ID, validasi input, dan update data

Catatan: Semua field bersifat optional, hanya field yang dikirim yang akan diupdate

Method 4: destroy() - DELETE /api/products/{id}



```php
public function destroy($id)
{
    $product = Product::findOrFail($id);
    $product->delete();
    return response()->json(['message' => 'Product deleted']);
}
```

Fungsi: Soft delete produk berdasarkan ID

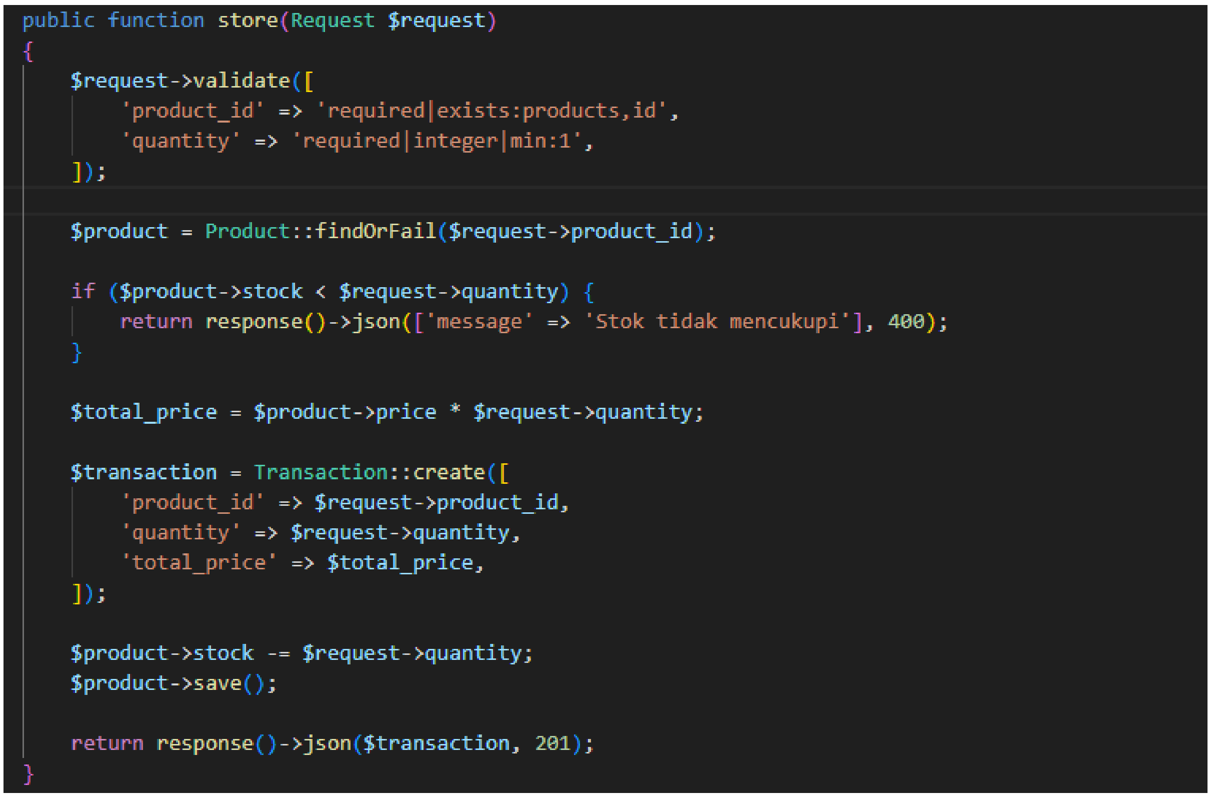
4.6. Controller: TransactionController.php

Lokasi: app/Http/Controllers/TransactionController.php

Fungsi:

- Menangani pembuatan transaksi pembelian
- Validasi stok produk
- Menghitung total harga otomatis
- Mengurangi stok produk setelah transaksi

Method: store() - POST /api/transactions



```php
public function store(Request $request)
{
    $request->validate([
        'product_id' => 'required|exists:products,id',
        'quantity' => 'required|integer|min:1',
    ]);

    $product = Product::findOrFail($request->product_id);

    if ($product->stock < $request->quantity) {
        return response()->json(['message' => 'Stok tidak mencukupi'], 400);
    }

    $total_price = $product->price * $request->quantity;

    $transaction = Transaction::create([
        'product_id' => $request->product_id,
        'quantity' => $request->quantity,
        'total_price' => $total_price,
    ]);

    $product->stock -= $request->quantity;
    $product->save();

    return response()->json($transaction, 201);
}
```

Logika penting:

1. Validasi product_id harus exists di tabel products
2. Validasi quantity minimal 1
3. Cek apakah stok mencukupi
4. Hitung total_price = price x quantity
5. Simpan transaksi
6. Kurangi stok produk
7. Return response transaksi

4.7. Routes:api.php

Lokasi: routes/api.php

Fungsi:

- Mendefinisikan semua endpoint API
- Menghubungkan URL dengan controller method

Kode:

```php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProductController;
use App\Http\Controllers\TransactionController;

Route::get('/products', [ProductController::class, 'index']);
Route::post('/products', [ProductController::class, 'store']);
Route::put('/products/{id}', [ProductController::class, 'update']);
Route::delete('/products/{id}', [ProductController::class, 'destroy']);

Route::post('/transactions', [TransactionController::class, 'store']);
```

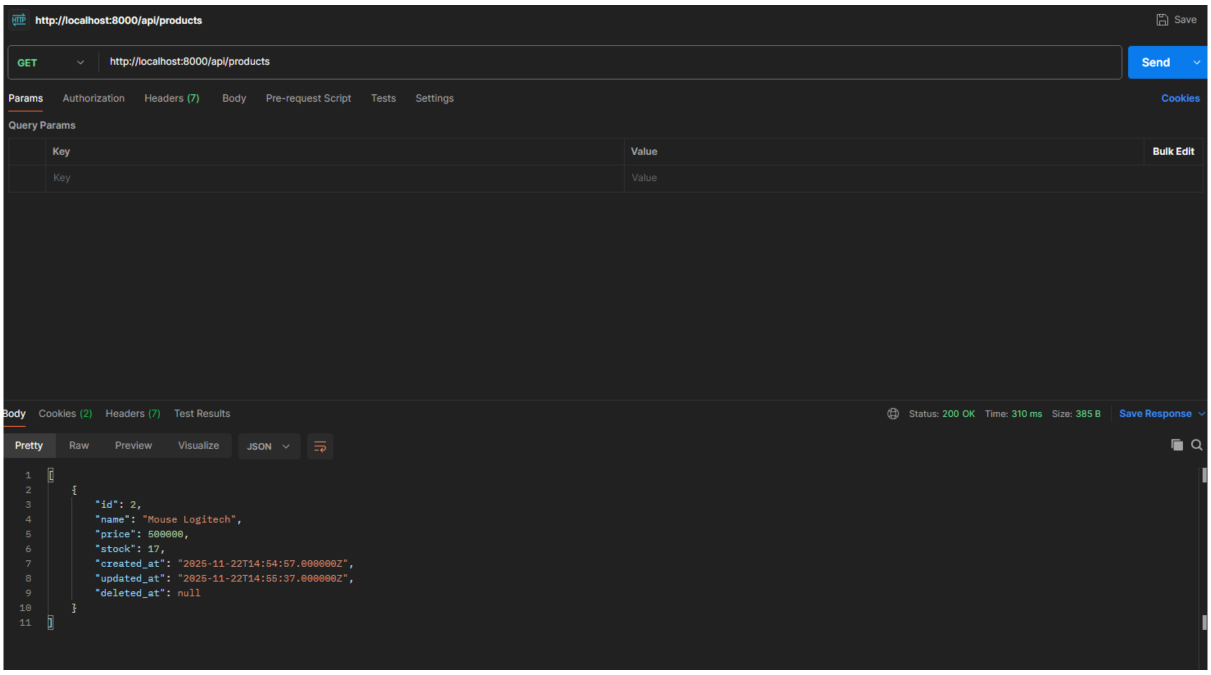
Endpoint yang tersedia:

- GET /api/products - Ambil semua produk
- POST /api/products - Tambah produk baru
- PUT /api/products/{id} - Update produk
- DELETE /api/products/{id} - Hapus produk
- POST /api/transactions - Buat transaksi

5. TESTING API DENGAN POSTMAN

5.1. GET /api/products - Ambil Semua Produk

Endpoint: GET http://localhost:8000/api/products



http://localhost:8000/api/products

GET http://localhost:8000/api/products

Query Params

Status: 200 OK Time: 310 ms Size: 385 B

```
[
    {
        "id": 2,
        "name": "Mouse Logitech",
        "price": 500000,
        "stock": 17,
        "created_at": "2025-11-22T14:54:57.000000Z",
        "updated_at": "2025-11-22T14:55:37.000000Z",
        "deleted_at": null
    }
]
```

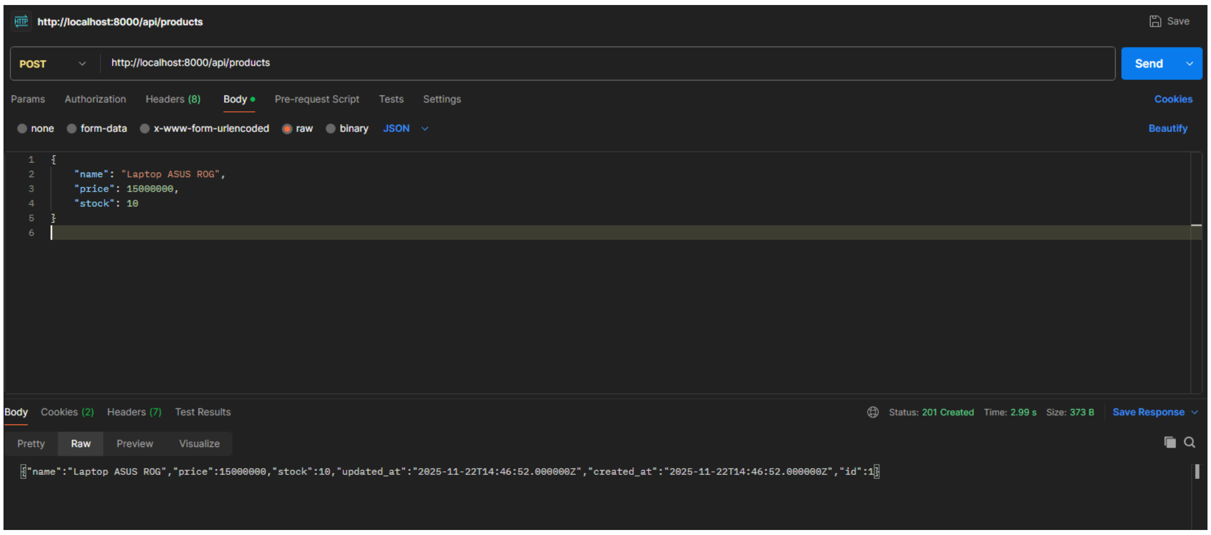
Status Code: 200 OK

Deskripsi: Endpoint ini mengembalikan semua data produk dalam format JSON array. Produk yang sudah di-soft delete tidak akan muncul.

5.2. POST /api/products - Tambah Produk Baru

Endpoint: POST http://localhost:8000/api/products

Request Body (JSON) & Response:



```
{
    "name": "Laptop ASUS ROG",
    "price": 15000000,
    "stock": 10
}
```

```
{"name":"Laptop ASUS ROG","price":15000000,"stock":10,"updated_at":"2025-11-22T14:46:52.000000Z","created_at":"2025-11-22T14:46:52.000000Z","id":1}
```

Status Code: 201 Created

Validasi:

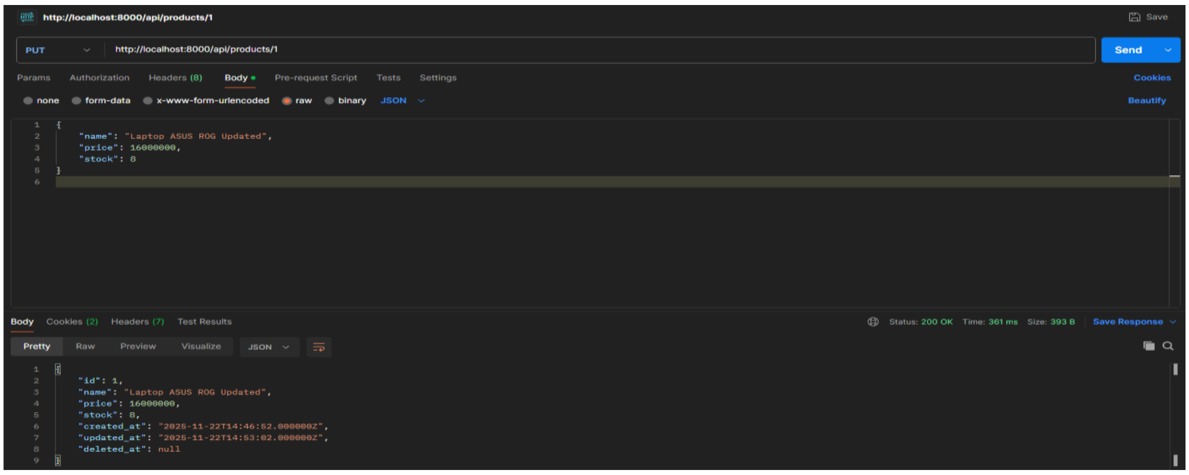
- name: required|string - Wajib diisi, tipe string
- price: required|integer - Wajib diisi, tipe integer
- stock: required|integer - Wajib diisi, tipe integer

Deskripsi: Endpoint ini menambahkan produk baru ke database. Semua field wajib diisi, jika tidak akan return error 422 Unprocessable Entity.

5.3. PUT /api/products/{id} - Update Produk

Endpoint: PUT http://localhost:8000/api/products/1

Request Body (JSON) & Response:



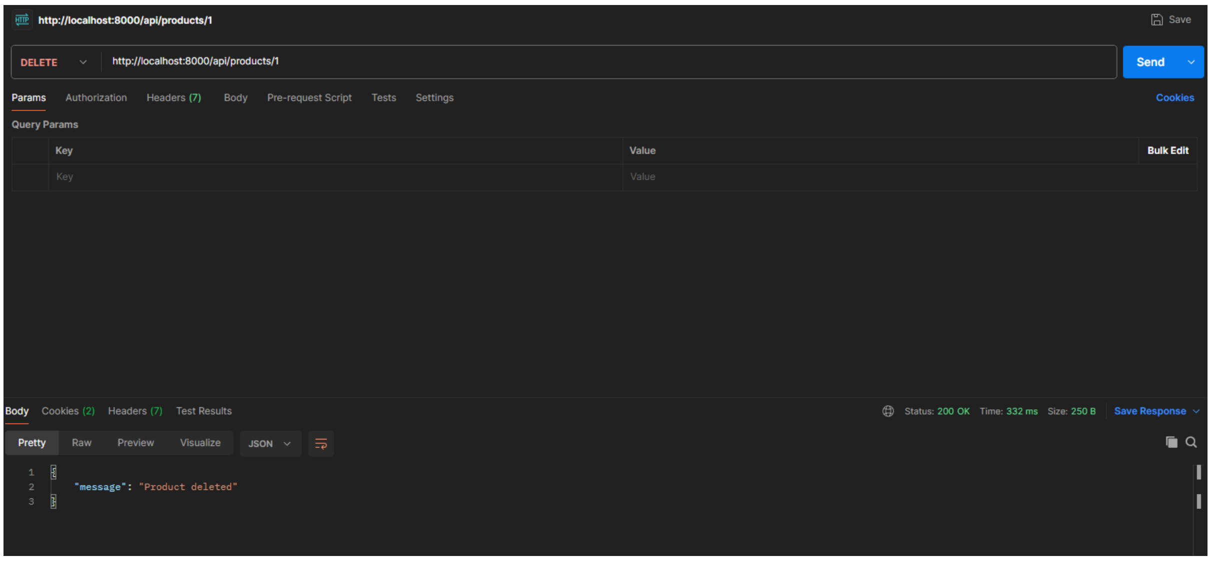
```
{
    "name": "Laptop ASUS ROG Updated",
    "price": 16000000,
    "stock": 8
}
```

```
{
    "id": 1,
    "name": "Laptop ASUS ROG Updated",
    "price": 16000000,
    "stock": 8,
    "created_at": "2025-11-22T14:46:52.000000Z",
    "updated_at": "2025-11-22T14:53:02.000000Z",
    "deleted_at": null
}
```

Status Code: 200 OK

Deskripsi: Endpoint ini mengupdate data produk berdasarkan ID. Semua field bersifat optional, hanya field yang dikirim yang akan diupdate. Field updated_at otomatis berubah.

5.4. DELETE /api/products/{id} - Hapus Produk



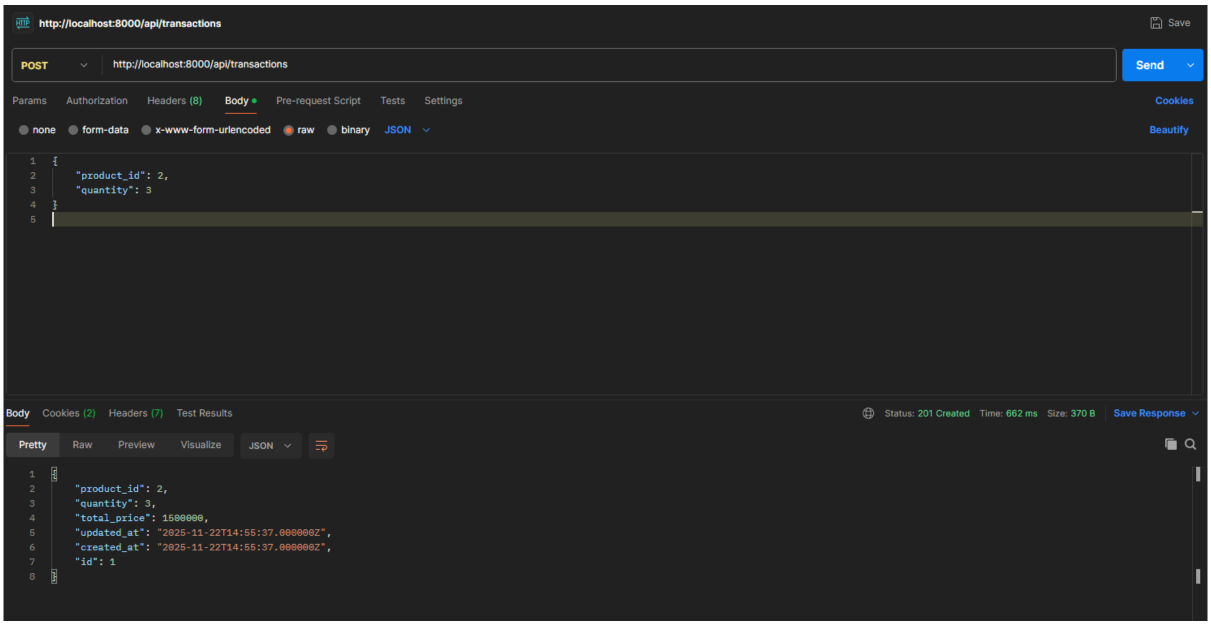
```
{
    "message": "Product deleted"
}
```

Status Code: 200 OK

Deskripsi: Endpoint ini melakukan soft delete produk berdasarkan ID. Produk tidak dihapus permanen, tetapi kolom deleted_at akan terisi dengan timestamp. Produk yang sudah di-soft delete tidak akan muncul di GET /api/products.

5.5. POST /api/transactions - Buat Transaksi

Endpoint: POST http://localhost:8000/api/transactions



```
{
    "product_id": 2,
    "quantity": 3
}
```

```
{
    "product_id": 2,
    "quantity": 3,
    "total_price": 1500000,
    "updated_at": "2025-11-22T14:55:37.000000Z",
    "created_at": "2025-11-22T14:55:37.000000Z",
    "id": 1
}
```

Status Code: 201 Created

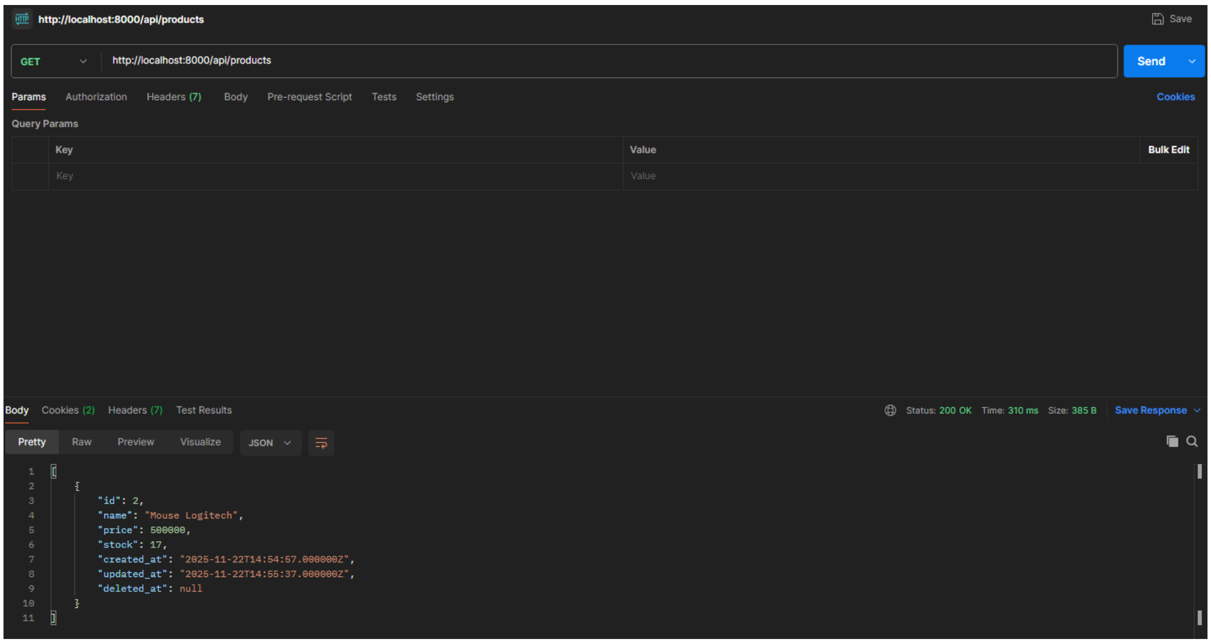
Logika yang terjadi:

1. Validasi product_id dan quantity
2. Cek stok produk (contoh: 20 unit tersedia)
3. Hitung total_price = 500000 x 3 = 1500000
4. Simpan transaksi
5. Kurangi stok produk dari 20 menjadi 17

Deskripsi: Endpoint ini membuat transaksi pembelian. Total harga dihitung otomatis dari price x quantity. Stok produk otomatis berkurang setelah transaksi berhasil.

5.6. GET /api/products - Cek Stok Setelah Transaksi

Endpoint: GET http://localhost:8000/api/products

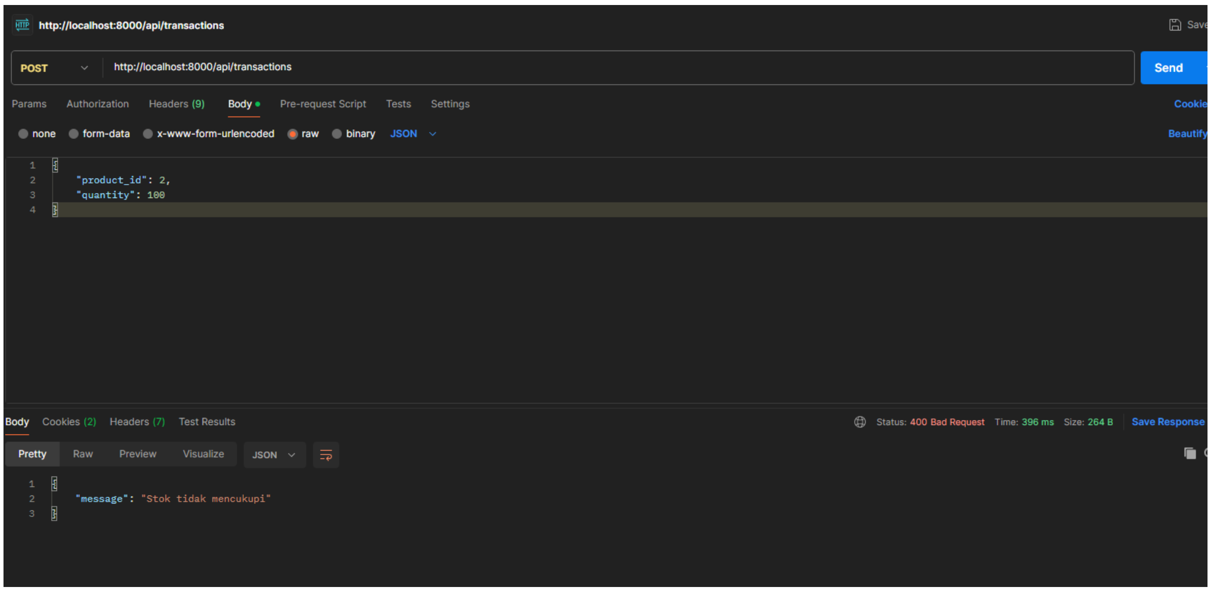


```
[
    {
        "id": 2,
        "name": "Mouse Logitech",
        "price": 500000,
        "stock": 17,
        "created_at": "2025-11-22T14:54:57.000000Z",
        "updated_at": "2025-11-22T14:55:37.000000Z",
        "deleted_at": null
    }
]
```

Bukti: Stok berkurang dari 20 menjadi 17 setelah transaksi 3 unit. Field updated_at juga berubah menunjukkan ada perubahan data.

5.7. POST /api/transactions - Validasi Stok Tidak Cukup

Endpoint: POST http://localhost:8000/api/transactions



```
POST http://localhost:8000/api/transactions
```

```
{
    "product_id": 2,
    "quantity": 100
}
```

```
Status: 400 Bad Request   Time: 396 ms   Size: 264 B
```

```
{
    "message": "Stok tidak mencukupi"
}
```

Deskripsi : Transaksi ditolak karena quantity (100) melebihi stok yang tersedia (17). Sistem melakukan pengecekan stok sebelum membuat transaksi untuk menjaga integritas data.

6. CARA MENJALANKAN APLIKASI

6.1. Setup Database

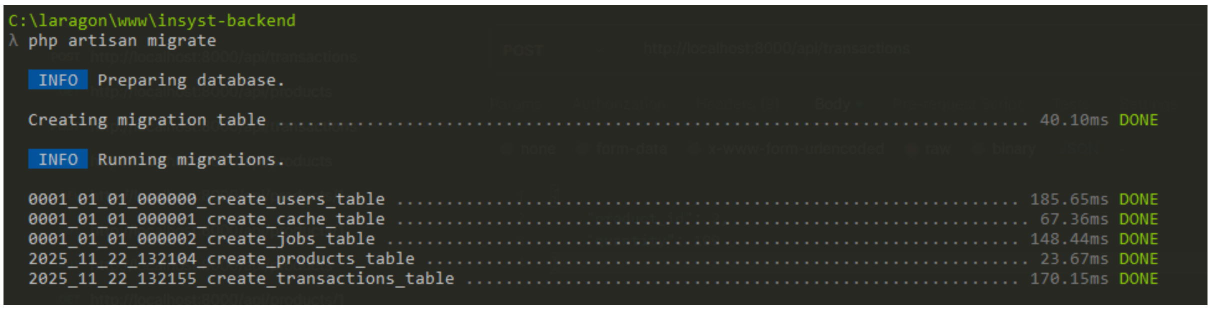
1. Buat database baru di phpMyAdmin/HeidiSQL dengan nama `insyst_backend`
2. Update file `.env`:

- DB_CONNECTION=mysql
  DB_HOST=127.0.0.1
  DB_PORT=3306
  DB_DATABASE=insyst_backend
  DB_USERNAME=root
  DB_PASSWORD=

6.2. Jalankan Migration

Buka terminal di folder project, jalankan :

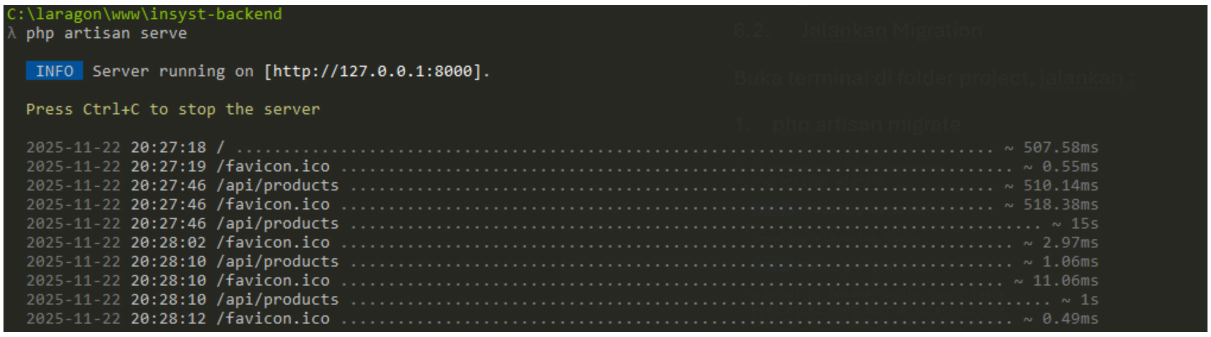
php artisan migrate



```
C:\laragon\www\insyst-backend
λ php artisan migrate

  INFO  Preparing database.

  Creating migration table ........................................................... 40.10ms DONE

  INFO  Running migrations.

  0001_01_01_000000_create_users_table ............................................. 185.65ms DONE
  0001_01_01_000001_create_cache_table .............................................. 67.36ms DONE
  0001_01_01_000002_create_jobs_table .............................................. 148.44ms DONE
  2025_11_22_132104_create_products_table ........................................... 23.67ms DONE
  2025_11_22_132155_create_transactions_table ...................................... 170.15ms DONE
```

6.3. Jalankan server

php artisan serve



```
C:\laragon\www\insyst-backend
λ php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server

  2025-11-22 20:27:18 / .............................................................................. ~ 507.58ms
  2025-11-22 20:27:19 /favicon.ico ...................................................................... ~ 0.55ms
  2025-11-22 20:27:46 /api/products ................................................................. ~ 510.14ms
  2025-11-22 20:27:46 /favicon.ico .................................................................. ~ 518.38ms
  2025-11-22 20:27:46 /api/products ....................................................................... ~ 15s
  2025-11-22 20:28:02 /favicon.ico ...................................................................... ~ 2.97ms
  2025-11-22 20:28:10 /api/products ..................................................................... ~ 1.06ms
  2025-11-22 20:28:10 /favicon.ico ..................................................................... ~ 11.06ms
  2025-11-22 20:28:10 /api/products ........................................................................ ~ 1s
  2025-11-22 20:28:12 /favicon.ico ...................................................................... ~ 0.49ms
```

Server akan berjalan di http://localhost:8000

6.4. Test API

Gunakan Postman untuk testing endpoint API sesuai dokumentasi di bagian 5.

7. TEKNOLOGI YANG DIGUNAKAN

- **Laravel 11** - PHP Framework
- **MySQL 8.4.3** - Database Management System
- **Eloquent ORM** - Database abstraction layer
- **RESTful API** - Architecture pattern
- **JSON** - Data interchange format
- **Postman** - API testing tool

8. FITUR YANG TELAH DIIMPLEMENTASIKAN

- CRUD Products (GET, POST, PUT, DELETE)
- Validasi input di setiap endpoint
- Soft Delete untuk products dan transactions
- Relasi foreign key antara products dan transactions
- POST Transactions dengan validasi stok
- Perhitungan total_price otomatis
- Pengurangan stok otomatis setelah transaksi
- Error handling untuk stok tidak cukup
- Response JSON di semua endpoint
- HTTP Status Code yang sesuai (200, 201, 400, 404, 422)

9. CATATAN TAMBAHAN

Keamanan

- Semua input divalidasi menggunakan Laravel validation
- Menggunakan findOrFail() untuk error handling 404
- Foreign key constraint untuk integritas data
- Mass assignment protection dengan $fillable

Database

- Menggunakan soft delete untuk audit trail
- Timestamp otomatis (created_at, updated_at)
- Relasi cascade delete untuk menjaga konsistensi data
- Foreign key constraint mencegah orphan data

Best Practices

- Separation of concerns (Model, Controller, Route terpisah)
- RESTful naming convention
- Consistent JSON response format
- Proper HTTP status codes
- Eloquent relationships untuk query yang efisien
- Single Responsibility Principle pada setiap method

Dokumentasi dibuat oleh: Yabes Edward Sihombing
Tanggal: 22-24 November 2025