Laporan Modul 9: RESTful API Laravel

Mata Kuliah: Workshop Web Lanjut
Nama: Maila Aziza
NIM: 2024573010024 **Kelas:** TI-2C

Abstrak

Modul ini berfokus pada implementasi RESTful API menggunakan Laravel, mencakup tahapan desain endpoint, manajemen resource, serta pengimplementasian protokol HTTP dengan berbagai metode (GET, POST, PUT, DELETE). Isi laporan mencakup pembuatan controller yang mengikuti pola API, pendefinisian route API, dan pengembalian respons data dalam format JSON yang terstruktur.

Pengujian dilakukan menggunakan Postman untuk memastikan setiap endpoint bekerja dengan baik, menguji pengiriman data, validasi respons, dan memverifikasi seluruh operasi CRUD berfungsi sesuai rencana. Dengan bantuan Postman, proses pengujian dan perbaikan error menjadi lebih sistematis dan mudah dipantau. Secara umum, pembelajaran ini memberikan wawasan komprehensif tentang cara mengembangkan API yang responsif, terjaga keamanannya, dan dapat terhubung dengan berbagai sistem aplikasi.

1. Dasar Teori

RESTful API merupakan pendekatan arsitektur web service yang menerapkan konsep REST untuk mengelola komunikasi antara klien dan server, menggunakan prinsip bahwa setiap informasi dipandang sebagai resource yang dapat diakses via URL dengan memanfaatkan metode HTTP (GET, POST, PUT, DELETE). Karakteristik utama meliputi: bersifat stateless artinya server tidak menyimpan sesi klien sebelumnya, menggunakan format data terstandar seperti JSON untuk pertukaran informasi, serta memiliki desain yang sederhana sehingga memungkinkan integrasi lintas platform yang lancar.

Peranan Metode HTTP dalam Operasi Resource

Setiap metode HTTP memiliki fungsi khusus dalam menangani resource pada API:

- **GET**: Mengambil atau membaca informasi dari server tanpa mengubah data, bisa untuk daftar lengkap atau detail tertentu
- **POST**: Membuat resource baru di sisi server berdasarkan data yang dikirim
- **PUT**: Mengganti seluruh resource yang ada, memerlukan pengiriman lengkap semua field yang dibutuhkan
- **PATCH**: Melakukan perubahan parsial pada resource, hanya field yang dimodifikasi yang dikirim
- **DELETE**: Menghilangkan resource tertentu dari penyimpanan data server

Di lingkungan Laravel, keseluruhan metode HTTP ini terhubung ke controller melalui sistem routing khusus untuk API, sehingga masing-masing metode dapat dialokasikan ke fungsi yang sesuai (index, store, show, update, destroy).

2. Langkah-Langkah Praktikum

Berikut ini adalah tahapan praktik yang telah dilaksanakan, dilengkapi dengan potongan kode program dan hasil tangkapan layar.

2.1 Praktikum 1 – Implementasi RESTful API dengan Laravel

Inisialisasi Project Laravel Baru

Buat proyek Laravel baru dengan nama `laravel-api` menggunakan command composer.

Konfigurasi Koneksi Database

Pengaturan koneksi database dilakukan di file `.env` dengan parameter berikut:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=authlab_db
DB_USERNAME=<username database anda>
DB_PASSWORD=<password database anda jika ada>
```

Setelah konfigurasi, jalankan migrasi awal dengan menjalankan command:

```
php artisan migrate
```

Pemasangan Laravel Breeze

Langkah pertama adalah mengunduh package Laravel Breeze:

```
composer require laravel/breeze --dev
```

Kemudian lakukan proses instalasi ke dalam project:

```
php artisan breeze:install
```

Selama setup, Anda akan dihadapkan pada beberapa pilihan konfigurasi:

- Untuk frontend framework, pilih opsi `api` (ini adalah persyaratan untuk API-only setup)

Setelah itu, pasang semua dependency frontend:

```
npm install
npm run dev
```

Pembuatan Model dan File Migrasi

Jalankan perintah artisan untuk membuat model beserta file migrasi sekaligus:

```
php artisan make:model Product -m
```

Perintah tersebut akan menghasilkan dua file: satu file model dan satu file migrasi.

Tahap Migrasi:

Buka file migrasi yang baru dibuat, kemudian modifikasi fungsi `up()` menjadi seperti berikut:

```
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->decimal('price', 10, 2);
        $table->text('description')->nullable();
        $table->integer('stock')->default(0);
        $table->timestamps();
    });
}
```

Setelah selesai, eksekusi migrasi:

```
php artisan migrate
```

Tahap Model:

Buka file model Product dan tambahkan implementasi berikut di dalam class:

```
use HasFactory;

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

Properti `$fillable` menentukan kolom-kolom mana saja yang diizinkan untuk diisi secara massal menggunakan method `create()` atau `update()`.

Pembentukan Request Validation Class

Buat file untuk menangani validasi request dengan command:

```
php artisan make:request ProductRequest
```

Command ini akan menghasilkan file bernama `ProductRequest.php` di folder `app/Http/Requests`. Buka file tersebut dan tambahkan kode di dalam class `ProductRequest`:

```
public function authorize(): bool
{
    return true;
}

public function rules(): array
{
    return [
        'name' => 'required|string|max:255',
        'price' => 'required|numeric',
        'description' => 'nullable|string',
        'stock' => 'required|integer|min:0',
    ];
}
```

Penjelasan Komponen:

- **Method authorize()**: Menentukan izin pengguna untuk melakukan request ini. Nilai `true` memungkinkan request diproses, nilai `false` akan menolaknya.
- **Method rules()**: Berisi daftar aturan validasi untuk masing-masing field input. Contohnya, field `name` harus diisi (required), bertipe string, dan maksimal 255 karakter.

Pembuatan Resource Class untuk Transformasi Data

Buat resource file dengan menjalankan command berikut:

```
php artisan make:resource ProductResource
```

Ini akan membuat file di `App\Http\Resources\ProductResource.php`. Tambahkan implementasi pada class `ProductResource`:

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'price' => $this->price,
        'description' => $this->description,
        'stock' => $this->stock,
        'created_at' => $this->created_at->format('Y-m-d H:i:s'),
        'updated_at' => $this->updated_at->format('Y-m-d H:i:s'),
```

```
        ];
    }
```

Resource ini berfungsi untuk mengubah model data menjadi array yang sesuai dengan format respons API.

Pembuatan Collection untuk Respons List

Buat collection resource dengan command:

```
php artisan make:resource ProductCollection
```

Ini menghasilkan file di `App\Http\Resources\ProductCollection.php`. Isi class `ProductCollection` dengan kode berikut:

```
public function toArray(Request $request): array
{
    return [
        'status' => true,
        'message' => 'Products retrieved successfully',
        'data' => $this->collection,
        'meta' => [
            'current_page' => $this->currentPage(),
            'last_page' => $this->lastPage(),
            'per_page' => $this->perPage(),
            'total' => $this->total(),
        ],
    ];
}
```

Collection ini memformat respons list data dengan menambahkan informasi status, pesan, dan metadata pagination.

Pengembangan Controller API

Buat controller khusus API dengan menjalankan command:

```
php artisan make:controller API\ProductController
```

Buka file `app/Http/Controllers/API/ProductController.php` dan definisikan method-method CRUD berikut:

```
public function index()
{
    $products = Product::latest()->paginate(10);
```

```
        return response()->json(new ProductCollection($products), Response::HTTP_OK);
    }

    public function store(ProductRequest $request)
    {
        $product = Product::create($request->validated());

        return response()->json([
            'status' => true,
            'message' => 'Product created successfully',
            'data' => new ProductResource($product),
        ], Response::HTTP_CREATED);
    }

    public function show(Product $product)
    {
        return response()->json([
            'status' => true,
            'message' => 'Product retrieved successfully',
            'data' => new ProductResource($product)
        ], Response::HTTP_OK);
    }

    public function update(ProductRequest $request, Product $product)
    {
        $product->update($request->validated());

        return response()->json([
            'status' => true,
            'message' => 'Product updated successfully',
            'data' => new ProductResource($product),
        ], Response::HTTP_OK);
    }

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

        return response()->json([
            'status' => true,
            'message' => 'Product deleted successfully',
        ], Response::HTTP_OK);
    }
```

Konfigurasi Routing API

Buka file `routes/api.php` dan daftarkan resource route berikut:

```
Route::apiResource('products', ProductController::class);
```

Setup Environment di Postman

Di panel Postman, navigasi ke bagian Environment untuk melakukan konfigurasi. Tambahkan variabel baru dengan nama `BASE_URL` dan isikan value dengan URL project Anda (misalnya `http://127.0.0.1:8000`).

 postman

Pengujian Method INDEX (Menampilkan Semua Data)

Method `index()` bertugas mengambil semua data produk dari database, mengurutkannya berdasarkan yang terbaru, dan mengirimkan respons dengan format pagination (10 item per halaman).

Pengujian dilakukan menggunakan Postman dengan endpoint `{{ BASE_URL }}/api/products` menggunakan metode GET:

 postman

Pengujian Method STORE (Membuat Data Baru)

Method `store()` digunakan untuk menyimpan data produk baru ke database melalui endpoint API. Buat request baru di Postman dengan label "CREATE DATA" menggunakan metode POST ke endpoint: `{{ BASE_URL }}/api/products`

Pada tab Body, pilih format "raw" dengan tipe JSON, kemudian masukkan data contoh:

```
{
    "name": "AIO Lenovo",
    "price": 50000000,
    "description": "PC AIO Lenovo.",
    "stock": 1
}
```

Setelah mengirim request, data berhasil disimpan ke database:

 postman

Pengujian Method UPDATE (Memperbarui Data)

Method `update()` memungkinkan perubahan data produk yang sudah ada. Laravel mendukung dua metode HTTP untuk update: PUT dan PATCH. PUT mengganti seluruh data (semua field harus dikirim), sementara PATCH hanya mengubah field tertentu.

Contoh melakukan update pada data produk dengan id = 4:

 postman

Pengujian Method DELETE (Menghapus Data)

Method `destroy()` bertanggung jawab untuk menghapus resource tertentu dari basis data. Endpoint DELETE memiliki struktur sama dengan update, yaitu memerlukan parameter id dari produk yang akan dihapus.

Setelah mengirim request DELETE ke endpoint `{{ BASE_URL }}/api/products/{id}`, data akan terhapus dari database:

 postman

Verifikasi penghapusan dapat dilakukan dengan mengambil data produk lagi untuk memastikan data sudah tidak ada:

 postman

---

3. Kesimpulan

Melalui praktikum ini, telah diperoleh pemahaman mendalam tentang penerapan arsitektur RESTful pada framework Laravel. Proses pembangunan API melibatkan berbagai komponen penting seperti routing, controller, model, resource, dan request validation yang bekerja secara sinergis membentuk sebuah layanan yang terstruktur.

Penggunaan HTTP method (GET, POST, PUT, DELETE) memungkinkan operasi CRUD dilakukan secara sistematis dan sesuai standar industri. Setiap metode memiliki peran spesifik dalam mengelola resource, sehingga klien dapat berkomunikasi dengan server dengan cara yang jelas dan terprediksi.

Postman terbukti menjadi alat yang sangat membantu dalam proses pengujian endpoint. Dengan fitur environment variable, custom headers, dan body testing yang lengkap, memastikan setiap endpoint berfungsi dengan baik sebelum diintegrasikan dengan aplikasi klien menjadi lebih mudah.

Konsep stateless pada RESTful API memberikan fleksibilitas tinggi dalam skalabilitas dan integrasi dengan berbagai platform. Respons dalam format JSON juga mempermudah parsing data di sisi klien. Secara keseluruhan, pembelajaran ini menunjukkan bahwa RESTful API merupakan fondasi penting dalam pengembangan aplikasi web modern yang memerlukan komunikasi antar sistem yang handal dan efisien.

---

4. Referensi

Cantumkan sumber yang Anda baca (buku, artikel, dokumentasi) — minimal 2 sumber. Gunakan format sederhana (judul — URL).

Tutorial Laravel 12 RESTful API — https://lagikoding.com/episode/tutorial-laravel-12-restful-api-1-install-laravel-12 Tutorial Laravel Rest API Untuk Pemula — https://www.rumahweb.com/journal/tutorial-laravel-rest-api/
