Tutorial Laboratorium: Membangun RESTful API Perpustakaan Universitas

Informatika (Kampus Kota Makassar)

18 November, 2025

Abstract

Dokumen ini berisi panduan langkah demi langkah untuk membangun **RESTful API** sederhana menggunakan **Laravel 11** dengan studi kasus Sistem Informasi Perpustakaan Universitas. Mahasiswa akan mengimplementasikan operasi CRUD (Create, Read, Update, Delete) pada dua entitas, yaitu `Book` (Buku) dan `Member` (Anggota), serta menguji setiap endpoint menggunakan Postman.

1 Tujuan Pembelajaran

1. Mampu mengatur proyek Laravel 11 untuk pengembangan API.
2. Mampu mengimplementasikan operasi CRUD yang sesuai dengan prinsip RESTful (penggunaan HTTP Method dan Status Code yang tepat).
3. Mampu menggunakan Postman untuk menguji dan memverifikasi fungsionalitas API.

2 Persiapan Awal

2.1 1.1 Inisialisasi Proyek dan Konfigurasi

1. Buka terminal dan buat proyek Laravel baru: `composer create-project laravel/laravel library-api-lab`
2. Masuk ke direktori proyek: `cd library-api-lab`
3. Instalasi API dan Sanctum: `php artisan install:api`
4. Konfigurasi koneksi database Anda di file `.env`.

2.2 1.2 Pembuatan Model dan Migrasi

Buat dua model beserta migrasinya:

```
php artisan make:model Member -m
php artisan make:model Book -m
```

2.2.1 Entitas Member

Edit `database/migrations/*_create_members_table.php`:

```
public function up(): void
{
    Schema::create('members', function (Blueprint $table) {
        $table->id();
        $table->string('student_id')->unique();
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamps();
    });
}
```

2.2.2 Entitas Book

Edit `database/migrations/*_create_books_table.php`:

```
public function up(): void
{
    Schema::create('books', function (Blueprint $table) {
        $table->id();
        $table->string('isbn')->unique();
        $table->string('title');
        $table->string('author');
        $table->integer('published_year');
        $table->integer('stock')->default(0);
        $table->timestamps();
    });
}
```

2.3 1.3 Menjalankan Migrasi dan Konfigurasi Model

1. Jalankan migrasi: `php artisan migrate`
2. Tambahkan `$fillable` di `app/Models/Member.php`: `protected $fillable = ['student_id', 'name', 'email'];`
3. Tambahkan `$fillable` di `app/Models/Book.php`: `protected $fillable = ['isbn', 'title', 'author', 'published_year', 'stock'];`

2.4 1.4 Pembuatan Controller dan Route

1. Buat dua API Resource Controller:

   ```
   php artisan make:controller MemberController --api
   php artisan make:controller BookController --api
   ```

2. Daftarkan routes di `routes/api.php`:

   ```
   use App\Http\Controllers\BookController;
   use App\Http\Controllers\MemberController;
   use Illuminate\Support\Facades\Route;

   Route::apiResource('members', MemberController::class);
   Route::apiResource('books', BookController::class);
   ```

3 Implementasi CRUD Entitas Buku

Implementasikan logika berikut di dalam `app/Http/Controllers/BookController.php`.

3.1 2.1. CREATE (POST)

```
// BookController.php

use App\Models\Book;
use Illuminate\Http\Request;
use Illuminate\Validation\Rule;

public function store(Request $request)
{
    $validated = $request->validate([
        'isbn' => 'required|unique:books|max:20',
        'title' => 'required|max:255',
        'author' => 'required|max:255',
        'published_year' => 'required|integer|max:' . date('Y'),
        'stock' => 'required|integer|min:0',
    ]);

    $book = Book::create($validated);
    // RESTful status code 201 Created
    return response()->json(['message' => 'Book added successfully', 'data' => $book], 20
}
```

3.2 2.2. READ (GET)

```
// BookController.php

public function index()
{
    // Status 200 OK (default)
    return response()->json(Book::all());
}

// Route Model Binding memastikan 404 jika resource tidak ditemukan
public function show(Book $book)
{
    // Status 200 OK
    return response()->json($book);
}
```

3.3 2.3. UPDATE (PATCH)

```
// BookController.php

public function update(Request $request, Book $book)
{
```

```
    $validated = $request->validate([
        // Ignore ID buku saat ini agar tetap bisa menggunakan ISBN yang sama
        'isbn' => ['sometimes', 'max:20', Rule::unique('books')->ignore($book->id)],
        'title' => 'sometimes|max:255',
        'author' => 'sometimes|max:255',
        'published_year' => 'sometimes|integer|max:' . date('Y'),
        'stock' => 'sometimes|integer|min:0',
    ]);

    $book->update($validated);
    // Status 200 OK (default)
    return response()->json(['message' => 'Book updated successfully', 'data' => $book]);
}
```

3.4 2.4. DELETE (DELETE)

```
// BookController.php

public function destroy(Book $book)
{
    $book->delete();
    // RESTful status code 204 No Content
    return response()->json(null, 204);
}
```

4 Tugas: Implementasi Anggota dan Pengujian

4.1 3.1. Tugas Implementasi MemberController

Mahasiswa diminta untuk melengkapi semua method (`index, store, show, update, destroy`) di `app/Http/Controllers/MemberController.php`.

Aturan Validasi Khusus Anggota:

- Kolom `student_id` dan `email` harus selalu `unique`.
- Saat `update`, gunakan `Rule::unique('members')->ignore($member->id)` untuk mengabaikan `student_id` dan `email` milik anggota tersebut.

4.2 3.2. Panduan Pengujian Postman

Jalankan `php artisan serve` dan uji semua `endpoint` menggunakan Postman.

Setiap request di Postman harus menyertakan header: `Accept: application/json`

5 Pelaporan dan Penyerahan

Mahasiswa wajib menyerahkan:

1. File `BookController.php` dan `MemberController.php` yang telah diisi lengkap.
2. **Screenshot Postman** yang menunjukkan hasil pengujian untuk kasus-kasus kritis berikut:
    - `POST /api/books` sukses dengan respons **201 Created**.

Table 1: Daftar Endpoint dan Status Code yang Diharapkan

Entitas	Metode	Endpoint	Body (Contoh)	Status Diharapkan
Buku	`POST`	`/api/books`	`{"isbn": "...", "title": "..."}`	**201 Created**
Buku	`GET`	`/api/books`	*None*	**200 OK**
Buku	`GET`	`/api/books/1`	*None*	**200 OK** / **404 Not Found**
Buku	`PATCH`	`/api/books/1`	`{"stock": 12}`	**200 OK** / **404 Not Found**
Buku	`DELETE`	`/api/books/1`	*None*	**204 No Content**
Anggota	`POST`	`/api/members`	`{"student_id": "...", "name": "..."}`	**201 Created**
Anggota	`POST`	`/api/members`	(Duplikat Student ID)	**422 Unprocessable**
Anggota	`GET`	`/api/members/1`	*None*	**200 OK** / **404 Not Found**
Anggota	`DELETE`	`/api/members/1`	*None*	**204 No Content**

- `POST /api/members` gagal karena kesalahan validasi (misalnya email duplikat) dengan respons **422 Unprocessable Content**.
- `DELETE /api/books/ID` sukses dengan respons **204 No Content**.
- `GET /api/books/ID` untuk ID yang sudah dihapus, menunjukkan respons **404 Not Found**.