Laporan

Pemograman API

Dosen Pengampu : Saiful Nur Budiman, S. Kom., M.Kom

Disusun Oleh :

Nafisa Ahlam Gisyeilla (23104410082)

Santi Kurniawati (23104410059)

Tia Neiska Puspita (23104410081)

Desi Widyawati (23104410090)

Annisa Kusuma Dewi (23104410100)

PRODI TEKNIK INFORMATIKA

FAKULTAS TEKNI DAN INFORMATIKA

UNIVERSITAS ISLAM BALITAR

November 2025

KATA PENGANTAR

Puji syukur diucapkan kehadirat Allah Swt. Atas segala rahmat-Nya sehingga makalah ini dapat tersusun sampai selesai. Tidak lupa penulis mengucapkan terima kasih terhadap bantuan dari pihak yang telah berkontribusi dengan memberikan sumbangan baik pikiran maupun materi.

Penulis sangat berharap semoga laporan ini dapat menambah pengetahuan pembaca. Bahkan penulis berharap lebih jauh lagi agar laporan ini dapat membuat pembaca lebih paham akan pembahasan yang dibahas di dalam makalah ini.

Penulis merasa bahwa masih banyak kekurangan dalam penyusunan laporan ini karena keterbatasan pengetahuan dan pengalama. Untuk itu penulis sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempurnaan laporan ini.

Blitar, 09 November 2025

Penulis

DAFTAR ISI

KATA PENGANTAR i
DAFTAR ISI ii
BAB I 1
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 1
1.3 Tujuan Penulisan 1
BAB II 2
LANDASAN TEORI 2
2.1 Laravel 2
2.2 Postman 2
2.3 Laravel Sanctum 2
2.4 Laravel Tinker 2
BAB III 3
PEMBAHASAN 3
3.1 Source Code 3
3.3 Hasil tampilan 9
BAB IV 11
PENUTUP 11
4.1 Kesimpulan 11
4.2 Saran 11
DAFTAR PUSTAKA 12

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan aplikasi modern, terutama pada arsitektur berbasis client-server, komunikasi antara bagian frontend (antarmuka pengguna) dan backend (server) menjadi hal yang sangat penting. Salah satu cara paling umum untuk melakukan komunikasi tersebut adalah dengan menggunakan Application Programming Interface (API). API berfungsi sebagai penghubung antara dua sistem yang berbeda agar dapat saling bertukar data dengan cara yang terstruktur. Salah satu komponen utama dalam API adalah endpoint. Endpoint merupakan alamat atau URL tertentu pada server yang menerima dan mengirimkan permintaan (request) dari klien.

Dengan adanya endpoint, sistem dapat mengatur jalur komunikasi untuk berbagai fungsi seperti mengambil data pengguna, menambahkan data baru, memperbarui data, hingga menghapus data. Misalnya, dalam aplikasi manajemen data siswa, endpoint dapat digunakan untuk mengakses daftar siswa, menambahkan siswa baru, atau mengubah informasi siswa yang sudah ada. Penggunaan endpoint API juga memberikan banyak keuntungan, seperti efisiensi dalam pengembangan aplikasi, fleksibilitas integrasi antar platform (misalnya web dan mobile), serta kemudahan dalam pemeliharaan dan pembaruan sistem. Namun, dalam penerapannya sering muncul berbagai permasalahan seperti keamanan endpoint, struktur data yang tidak konsisten, dan dokumentasi API yang kurang jelas.

1.2 Rumusan Masalah

1. Apa yang dimaksud dengan endpoint API dan bagaimana perannya dalam komunikasi antara client dan server?
2. Bagaimana cara merancang dan mengimplementasikan endpoint API yang efisien dan mudah digunakan?
3. Apa saja tantangan atau permasalahan yang sering muncul dalam pengembangan endpoint API, serta bagaimana solusinya?

1.3 Tujuan Penulisan

1. Untuk memahami pengertian dan fungsi endpoint API dalam sistem berbasis client-server.
2. Untuk mengetahui langkah-langkah perancangan dan implementasi endpoint API yang baik dan terstruktur.
3. Untuk mengidentifikasi dan mencari solusi terhadap permasalahan umum yang terjadi dalam pengembangan endpoint API.

BAB II

LANDASAN TEORI

2.1 Laravel

Laravel adalah framework PHP yang mengikuti pola arsitektur MVC (Model-View-Controller). Laravel menyediakan routing, ORM (Eloquent), Blade templating, queue, middleware, serta tooling frontend (Vite) yang mempercepat pengembangan web modern. Untuk integrasi payment gateway, struktur controller, service class, dan routing di Laravel memudahkan pemisahan logika server-side (membuat order, memanggil API Midtrans, menerima callback). Instalasi dan setup Laravel dapat dilakukan via Composer dan dokumentasi resmi Laravel menguraikan praktik instalasi, environment (.env), serta valet/valet/ngrok untuk pengembangan lokal.

2.2 Postman

Postman adalah sebuah alat (tool) yang digunakan untuk menguji, mengelola, dan mendokumentasikan API. Aplikasi ini memungkinkan pengembang untuk mengirim berbagai jenis permintaan (request) HTTP seperti GET, POST, PUT, dan DELETE ke server, serta melihat hasil respon yang dikembalikan. Dengan Postman, pengembang dapat dengan mudah melakukan pengujian endpoint API tanpa perlu membuat antarmuka pengguna terlebih dahulu. Postman juga mendukung autentikasi, parameter dinamis, serta koleksi API yang dapat disimpan dan dibagikan antar tim.

2.3 Laravel Sanctum

Laravel Sanctum adalah paket resmi dari Laravel yang digunakan untuk autentikasi API berbasis token yang sederhana dan ringan. Sanctum memungkinkan setiap pengguna aplikasi untuk membuat token API yang digunakan saat mengakses endpoint tertentu, sehingga keamanan komunikasi antara client dan server tetap terjaga.

2.4 Laravel Tinker

Laravel Tinker adalah tool interaktif berbasis command line yang digunakan untuk berinteraksi langsung dengan aplikasi Laravel. Tinker menyediakan antarmuka REPL (Read–Eval–Print Loop) yang memungkinkan pengembang menjalankan perintah PHP secara langsung dalam konteks Laravel.

2.5 Faker

Faker adalah library PHP yang digunakan untuk membuat data palsu (dummy data) seperti nama, alamat, email, nomor telepon, dan lain-lain. Dalam Laravel, Faker biasanya digunakan saat seeding database agar pengembang memiliki data simulasi untuk pengujian sistem. Faker membantu dalam proses pengembangan aplikasi karena pengembang dapat menguji fitur seperti pencarian, pagination, dan validasi tanpa harus memasukkan data manual.

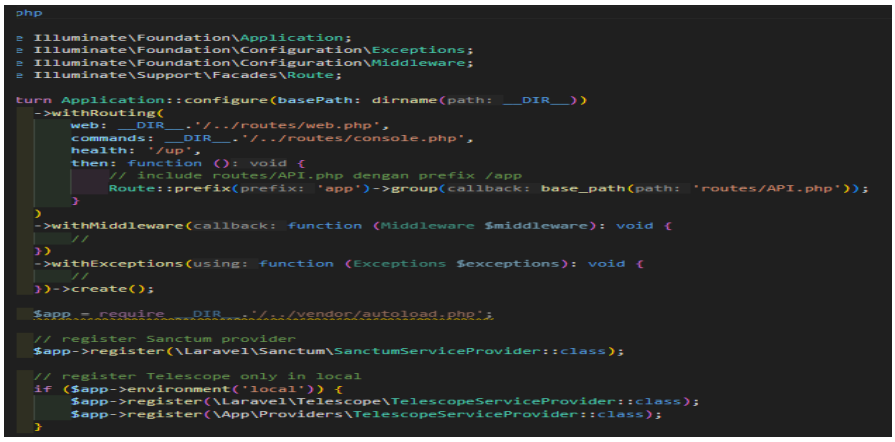
BAB III

PEMBAHASAN

3.1 Source Code

1. Booststrap

a. app.php



```php
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use Illuminate\Support\Facades\Route;

return Application::configure(basePath: dirname(path: __DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
        then: function (): void {
            // include routes/API.php dengan prefix /app
            Route::prefix(prefix: 'app')->group(callback: base_path(path: 'routes/API.php'));
        }
    )
    ->withMiddleware(callback: function (Middleware $middleware): void {
        //
    })
    ->withExceptions(using: function (Exceptions $exceptions): void {
        //
    })->create();

$app = require __DIR__.'/../vendor/autoload.php';

// register Sanctum provider
$app->register(\Laravel\Sanctum\SanctumServiceProvider::class);

// register Telescope only in local
if ($app->environment('local')) {
    $app->register(\Laravel\Telescope\TelescopeServiceProvider::class);
    $app->register(\App\Providers\TelescopeServiceProvider::class);
}
```

File bootstrap/app.php pada Laravel berfungsi sebagai konfigurasi utama aplikasi yang mengatur bagaimana framework dijalankan. Di dalamnya terdapat pengaturan Application::configure() yang menentukan direktori dasar proyek serta memuat konfigurasi routing, middleware, dan penanganan error. Bagian withRouting() menunjukkan lokasi file rute seperti routes/web.php untuk halaman web, routes/console.php untuk perintah Artisan, serta menambahkan pengecekan kesehatan server melalui endpoint /up. Pada fungsi tambahan then, terdapat perintah Route::prefix('app')->group(base_path('routes/api.php')); yang berarti semua rute dalam routes/api.php akan memiliki awalan /app, sehingga endpoint seperti /user menjadi /app/user. Selanjutnya, blok withMiddleware() disediakan untuk menambahkan middleware global seperti autentikasi atau pembatasan request, sedangkan withExceptions() digunakan untuk menangani kesalahan atau exception kustom. Akhirnya, metode ->create() membentuk instance aplikasi Laravel yang siap dijalankan. Dengan demikian, file ini menjadi inti dari proses konfigurasi awal framework Laravel yang menghubungkan seluruh komponen utama aplikasi.

2. Routes

a. api.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\MahasiswaController;

// Endpoint Login (POST)
Route::post(uri: '/login', action: [AuthController::class, 'login']);

// Endpoint Mahasiswa (GET) - hanya bisa diakses dengan token
Route::middleware(middleware: 'auth:sanctum')->get(uri: '/mahasiswa', action: [MahasiswaController::class, 'index']);

Route::get(uri: '/test', action: function (): JsonResponse {
    return response()->json(data: ['message' => 'API works!']);
});
```

File api.php berfungsi untuk mendefinisikan rute API di Laravel, dimana terdapat endpoint POST /api/login untuk proses login melalui AuthController dan menghasilkan token, kemudian ada endpoint GET /api/mahasiswa yang hanya dapat diakses oleh user yang sudah login karena dilindungi middleware auth:sanctum dan akan mengambil data mahasiswa melalui MahasiswaController, serta endpoint GET /api/test yang digunakan untuk memastikan bahwa API berjalan dengan mengembalikan pesan JSON sederhana.

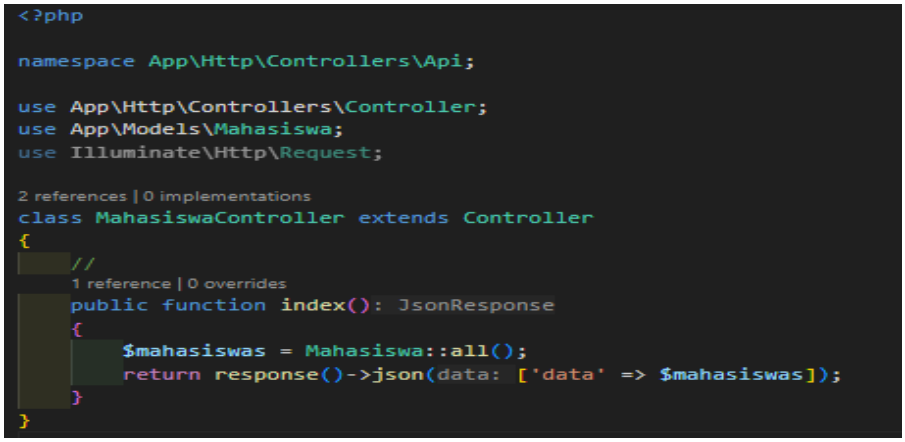
3. App

a. AuthController.php

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\ApiUser;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\ValidationException;

2 references | 0 implementations
class AuthController extends Controller
{
    //
    1 reference | 0 overrides
    public function login(Request $request): JsonResponse
    {
        $request->validate(rules: [
            'email' => 'required|email',
            'password' => 'required'
        ]);

        $user = ApiUser::where(column: 'email', operator: $request->email)->first();

        if (! $user || ! Hash::check(value: $request->password, hashedValue: $user->password)) {
            throw ValidationException::withMessages(messages: [
                'email' => ['The provided credentials are incorrect.'],
            ]);
        }

        $token = $user->createToken(name: 'api-token')->plainTextToken;

        return response()->json(data: [
            'access_token' => $token,
            'token_type' => 'Bearer',
            'user' => $user->only(attributes: ['id','name','email'])
        ]);
    }
}
```

AuthController.php berfungsi menangani proses login di API Laravel menggunakan Sanctum, dimana controller menerima input email dan password, memvalidasi data yang dikirim, mencari user berdasarkan email pada database, lalu memeriksa apakah password cocok; jika email tidak ditemukan atau password salah maka mengembalikan respons gagal, namun jika valid maka controller membuat token menggunakan Sanctum dan mengembalikan respons JSON berisi informasi user beserta token, yang nantinya digunakan untuk mengakses endpoint API yang membutuhkan autentikasi.

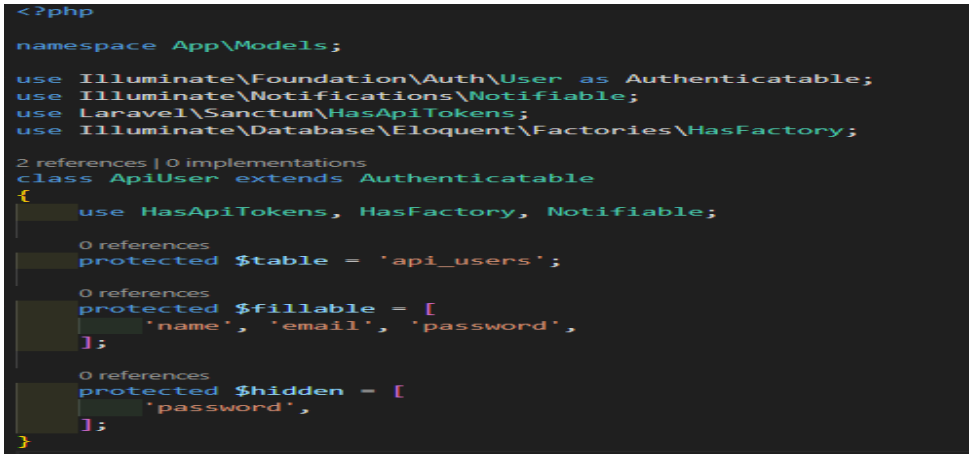
b. MahasiswaController.php



```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Mahasiswa;
use Illuminate\Http\Request;

2 references | 0 implementations
class MahasiswaController extends Controller
{
    //
    1 reference | 0 overrides
    public function index(): JsonResponse
    {
        $mahasiswas = Mahasiswa::all();
        return response()->json(data: ['data' => $mahasiswas]);
    }
}
```

MahasiswaController berfungsi untuk menangani permintaan API terkait data mahasiswa. Controller ini mengambil data dari model Mahasiswa (biasanya dari database) dan mengembalikannya dalam bentuk JSON. Di dalam method index(), controller menjalankan Mahasiswa::all() untuk mengambil seluruh data mahasiswa dari tabel, kemudian mengembalikan response dalam format JSON melalui return response()->json(). Controller ini biasanya dipanggil melalui endpoint GET /api/mahasiswa dan hanya bisa diakses jika user sudah login dan memiliki token (karena rutenya dilindungi middleware auth:sanctum). Dengan kata lain, controller ini adalah bagian dari API yang menyediakan data mahasiswa ke frontend, mobile, atau Postman.

c. ApiUser.php



```php
<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Illuminate\Database\Eloquent\Factories\HasFactory;

2 references | 0 implementations
class ApiUser extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    0 references
    protected $table = 'api_users';

    0 references
    protected $fillable = [
        'name', 'email', 'password',
    ];

    0 references
    protected $hidden = [
        'password',
    ];
}
```

File ApiUser.php adalah model di Laravel yang merepresentasikan tabel user khusus untuk keperluan API. Model ini digunakan untuk berinteraksi dengan database — seperti mengambil, menyimpan, atau mengupdate data user. Di dalamnya terdapat deklarasi kolom mana saja yang boleh di-insert atau di-update melalui proses mass assignment, biasanya melalui properti $fillable. Selain itu, jika model ini dipakai bersama Laravel Sanctum, model juga dapat membuat

token untuk autentikasi API. Intinya, model ApiUser.php berfungsi sebagai jembatan antara database dan controller, sehingga saat controller membutuhkan data user, controller cukup memanggil model ini. Model inilah yang mengeksekusi proses ke database dan mengembalikan hasilnya dalam bentuk data.

4. Migrations

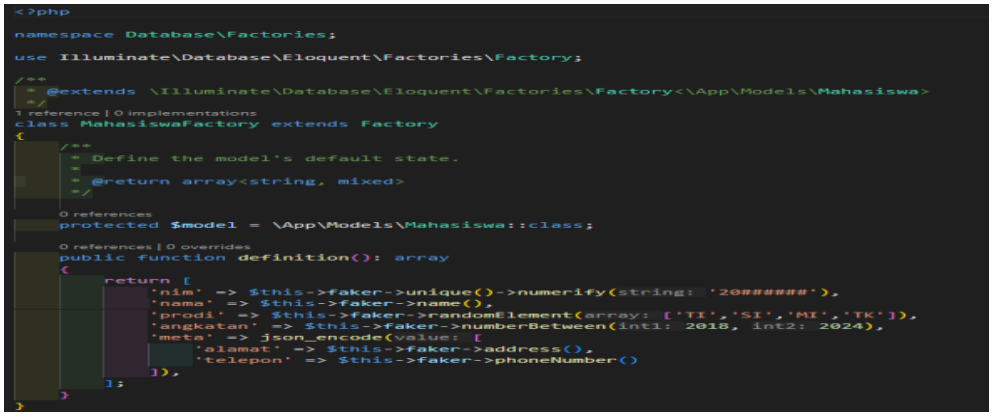
a. 2025_11_05_110221_create_personal_access_tokens_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'personal_access_tokens', callback: function (Blueprint $table): void {
            $table->id();
            $table->morphs(name: 'tokenable');
            $table->text(column: 'name');
            $table->string(column: 'token', length: 64)->unique();
            $table->text(column: 'abilities')->nullable();
            $table->timestamp(column: 'last_used_at')->nullable();
            $table->timestamp(column: 'expires_at')->nullable()->index();
            $table->timestamps();
        });
    }

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

File ini berfungsi untuk membuat tabel personal_access_tokens di database, yaitu tabel yang digunakan oleh Laravel Sanctum untuk menyimpan token autentikasi setiap kali user berhasil login melalui API; tabel ini berisi informasi seperti tokenable_type (jenis model yang menggunakan token, misalnya User), tokenable_id (ID user), name (nama token), token (token yang sudah disimpan dalam bentuk hash untuk keamanan), abilities (hak akses token), serta waktu token dibuat, digunakan, dan diperbarui, sehingga token tersebut dapat dipakai untuk mengakses endpoint API yang membutuhkan autentikasi.

5. Database

a. MahasiswaFactory.php

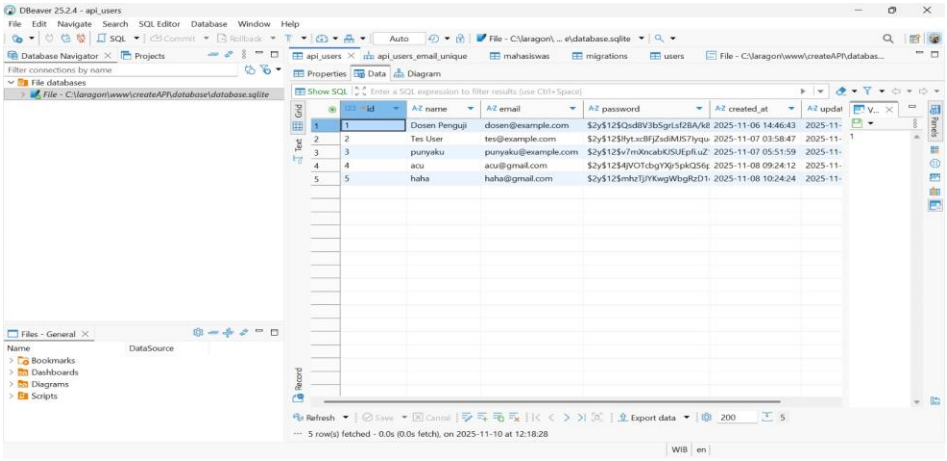


```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Mahasiswa>
 */
1 reference | 0 implementations
class MahasiswaFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    0 references
    protected $model = \App\Models\Mahasiswa::class;

    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'nim' => $this->faker->unique()->numerify(string: '20######'),
            'nama' => $this->faker->name(),
            'prodi' => $this->faker->randomElement(array: ['TI','SI','MI','TK']),
            'angkatan' => $this->faker->numberBetween(int1: 2018, int2: 2024),
            'meta' => json_encode(value: [
                'alamat' => $this->faker->address(),
                'telepon' => $this->faker->phoneNumber()
            ]),
        ];
    }
}
```

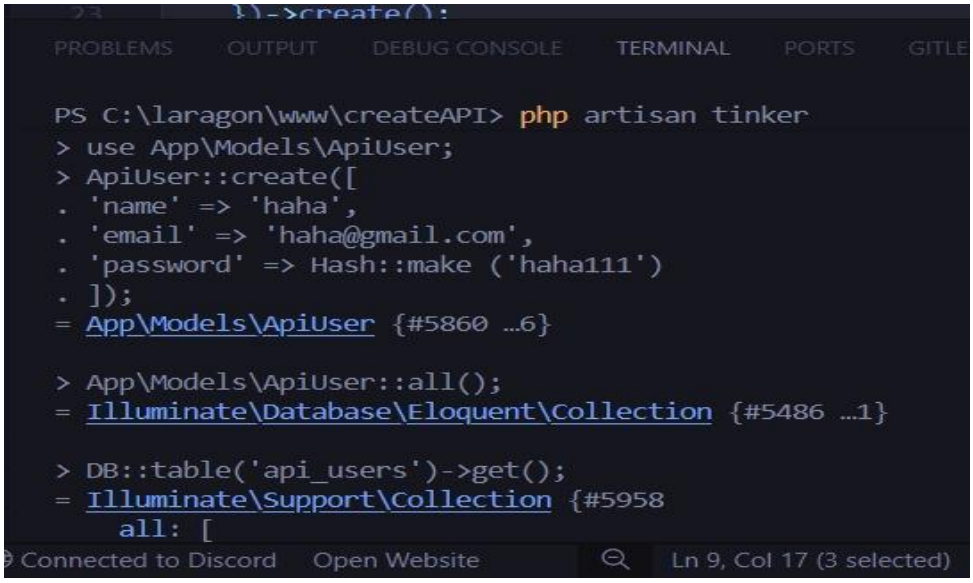
File ini digunakan untuk membuat data dummy atau data palsu ke dalam tabel mahasiswa, biasanya saat testing atau seeding database. File ini berada di

namespace Database\Factories dan menggunakan kelas dasar Illuminate\Database\Eloquent\Factories\Factory. Factory ini dihubungkan dengan model App\Models\Mahasiswa melalui properti $model. Di dalam metode definition(), factory mendefinisikan struktur data yang akan dibuat secara acak menggunakan library Faker. Nilai-nilai yang dihasilkan antara lain: nim berupa angka unik dengan pola 20######, nama berisi nama acak, prodi diambil secara acak dari empat pilihan (TI, SI, MI, TK), angkatan berupa angka antara 2018–2024, dan meta berisi data tambahan berupa JSON yang mencakup alamat dan telepon. Factory ini memungkinkan pembuatan data otomatis seperti dengan perintah Mahasiswa::factory()->count(10)->create(); sehingga sangat berguna untuk pengujian atau pengisian awal database tanpa harus membuat data secara manual.

b. Db'api_users'

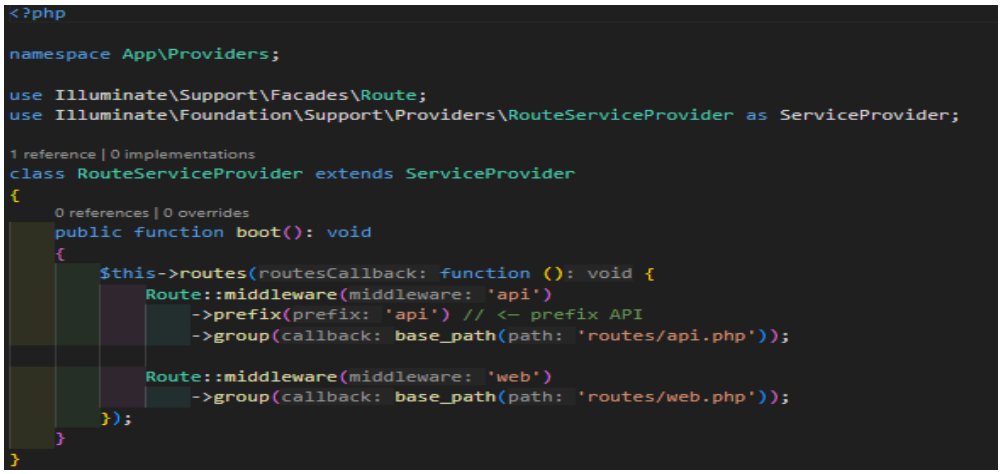


Gambar ini menghasilkan data dari database ApiUser, data didapatkan melalui tinker.



6. Providers

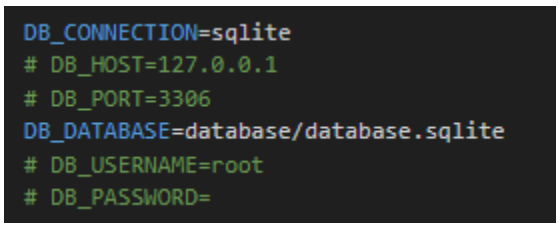
a. RouteServiceProvider.php



```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Route;
use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

1 reference | 0 implementations
class RouteServiceProvider extends ServiceProvider
{
    0 references | 0 overrides
    public function boot(): void
    {
        $this->routes(routesCallback: function (): void {
            Route::middleware(middleware: 'api')
                ->prefix(prefix: 'api') // <- prefix API
                ->group(callback: base_path(path: 'routes/api.php'));

            Route::middleware(middleware: 'web')
                ->group(callback: base_path(path: 'routes/web.php'));
        });
    }
}
```

File ini berfungsi sebagai pengatur utama semua rute (routes) dalam aplikasi Laravel, yaitu tempat menentukan bagaimana URL tertentu akan diarahkan ke controller. File ini berada di namespace App\Providers dan mewarisi (extends) kelas ServiceProvider dari Laravel. Di dalam metode boot(), terdapat konfigurasi pemuatan rute menggunakan $this->routes(function () { ... }); yang memisahkan dua jenis rute utama: rute API dan rute web. Pada bagian Route::middleware('api')->prefix('api')->group(base_path('routes/api.php'));,
Laravel memuat semua rute yang didefinisikan di file routes/api.php, memberikan prefix api di depan setiap URL-nya (misalnya api/users), serta menerapkan middleware api yang biasanya digunakan untuk komunikasi data tanpa sesi.Sementara itu, baris Route::middleware('web')->group(base_path('routes/web.php')); memuat rute yang terdapat di file routes/web.php dan menerapkan middleware web, yang mengaktifkan fitur seperti sesi, cookie, dan CSRF protection untuk halaman web biasa. Dengan demikian, file ini menjadi penghubung utama antara URL yang diakses oleh pengguna dan file definisi rutenya, serta memastikan setiap jenis rute dijalankan dengan konfigurasi middleware yang sesuai.

7. .env



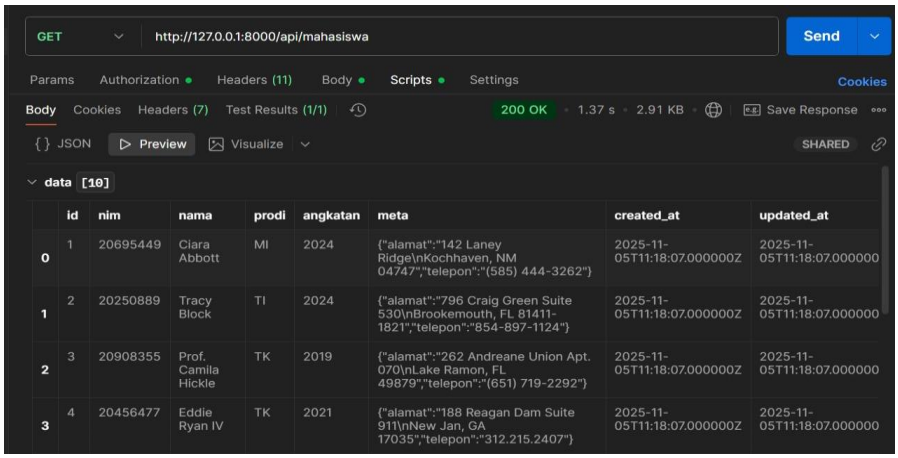
```
DB_CONNECTION=sqlite
# DB_HOST=127.0.0.1
# DB_PORT=3306
DB_DATABASE=database/database.sqlite
# DB_USERNAME=root
# DB_PASSWORD=
```

File .env ini berfungsi untuk menyimpan pengaturan utama aplikasi Laravel. Isinya mengatur nama dan mode aplikasi (APP_NAME, APP_ENV, APP_DEBUG),

alamat URL, serta kunci enkripsi (APP_KEY). Database menggunakan SQLite yang tersimpan di database/database.sqlite. Sistem sesi, cache, dan antrian menggunakan database, sedangkan file dan log disimpan secara lokal. Redis diatur sebagai penyimpanan tambahan, dan mailer menggunakan mode log (email tidak dikirim sungguhan). Secara keseluruhan, file ini mengatur bagaimana aplikasi Laravel berjalan di lingkungan pengembangan lokal.

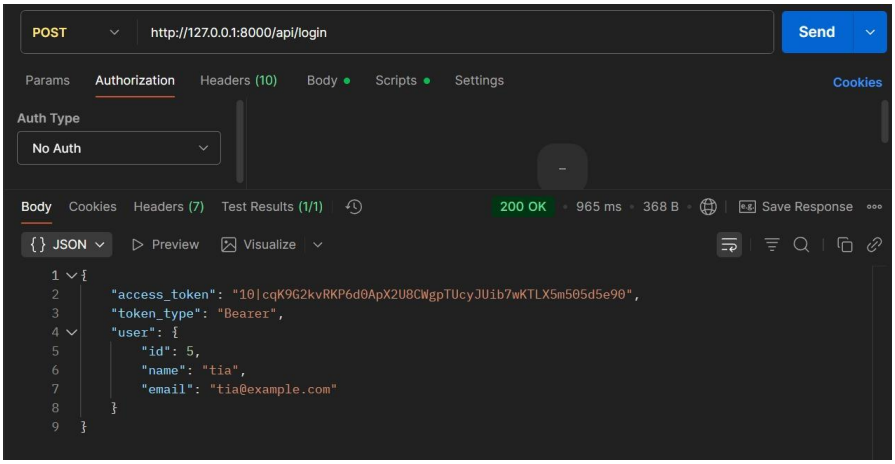
3.3. Hasil tampilan

1. GET



	id	nim	nama	prodi	angkatan	meta	created_at	updated_at
0	1	20695449	Ciara Abbott	MI	2024	{"alamat":"142 Laney Ridge\nKochhaven, NM 04747","telepon":"(585) 444-3262"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
1	2	20250889	Tracy Block	TI	2024	{"alamat":"796 Craig Green Suite 530\nBrookemouth, FL 81411-1821","telepon":"854-897-1124"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
2	3	20908355	Prof. Camila Hickle	TK	2019	{"alamat":"262 Andreane Union Apt. 070\nLake Ramon, FL 49879","telepon":"(651) 719-2292"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
3	4	20456477	Eddie Ryan IV	TK	2021	{"alamat":"188 Reagan Dam Suite 911\nNew Jan, GA 17035","telepon":"312.215.2407"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
4	5	20666632	Sallie Legros	TI	2018	{"alamat":"2792 Rachael Point\nPort Dell, AZ 91114-0927","telepon":"+1 (510) 944-0896"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
5	6	20892325	Dr. Payton McKenzie DVM	SI	2019	{"alamat":"863 Weissnat Radial Apt. 174\nPort Mable, MT 32966","telepon":"+1-737-818-4017"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
6	7	20101296	Prof. Marlen Champlin MD	MI	2022	{"alamat":"79483 Ankunding Inlet Apt. 440\nLisettemouth, PA 32408","telepon":"+1.386.555.8883"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
7	8	20466321	Anya Fritsch	SI	2023	{"alamat":"49193 Mertie Plains Apt. 457\nEast Carey, NV 25552-7826","telepon":"+1.915.479.4585"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
8	9	20758759	Prof. Kurt Parker	TI	2021	{"alamat":"84490 Hickle Light\nNorth Melissa, ID 21626-0537","telepon":"510.402.6617"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000
9	10	20595185	Miss Shea Luettgen IV	TK	2018	{"alamat":"27515 Larson Street\nMaximusview, CO 58346-1523","telepon":"(270) 406-3434"}	2025-11-05T11:18:07.000000Z	2025-11-05T11:18:07.000000

2. POST


POST
http://127.0.0.1:8000/api/login
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
Auth Type
No Auth
Body
Cookies
Headers (7)
Test Results (1/1)
200 OK
965 ms
368 B
Save Response
JSON
Preview
Visualize
{
"access_token": "10|cqK9G2kvRKP6d0ApX2U8CWgpTUcyJUib7wKTLX5m505d5e90",
"token_type": "Bearer",
"user": {
"id": 5,
"name": "tia",
"email": "tia@example.com"
}
}

BAB IV

PENUTUP

4.1 Kesimpulan

Dari pembahasan dapat disimpulkan bahwa pengembangan sistem berbasis API (Application Programming Interface) memiliki peranan penting dalam membangun komunikasi antara client dan server secara efisien. Endpoint API berfungsi sebagai jalur utama pertukaran data yang harus dirancang dengan struktur yang jelas, aman, dan mudah diakses. Dalam proses pengembangannya, berbagai alat bantu seperti Postman, Sanctum, Tinker, dan Faker berperan besar dalam meningkatkan efisiensi dan keandalan sistem. Postman digunakan untuk menguji dan memastikan fungsi setiap endpoint API, Laravel Sanctum berfungsi untuk mengamankan akses API melalui sistem token, Tinker memudahkan pengembang berinteraksi langsung dengan data dan logika aplikasi, sedangkan Faker digunakan untuk membuat data uji palsu (dummy data) yang membantu dalam proses pengujian dan pengembangan. Dengan penerapan konsep dan alat tersebut, sistem API dapat berjalan lebih aman, terstruktur, dan mudah dipelihara.

4.2 Saran

Dalam mengembangkan aplikasi berbasis API, pengembang disarankan untuk merancang struktur endpoint secara konsisten agar mudah dipahami dan digunakan oleh tim lain, serta menerapkan sistem autentikasi dan otorisasi yang kuat menggunakan Laravel Sanctum untuk menjaga keamanan data. Pengujian rutin menggunakan Postman perlu dilakukan sebelum API diintegrasikan ke aplikasi utama guna memastikan bahwa setiap endpoint berjalan sesuai fungsinya. Selain itu, pemanfaatan Tinker dan Faker dapat membantu proses pengembangan dengan mempercepat pengujian serta penyediaan data tiruan yang realistis. Terakhir, setiap endpoint API sebaiknya didokumentasikan dengan baik agar memudahkan proses pemeliharaan dan pengembangan sistem di masa mendatang.

DAFTAR PUSTAKA

Andrianto, L. D., & Suyatno, D. F. (2024). Analisis Performa Load Testing Antara MySQL dan NoSQL MongoDB pada REST API NodeJS menggunakan Postman. Journal of Emerging Information Systems and Business Intelligence (JEISBI), 5(1), 18-26. DOI:10.26740/jeisbi.v5i1.58157.

Baudry, B., Etemadi, K., Fang, S., Gamage, Y., Liu, Y., … et al. (2024). Generative AI to Generate Test Data Generators. arXiv preprint. (Menyinggung perpustakaan “faking” data seperti Faker).

Kore, P. P., Lohar, M. J., Surve, M. T., & Jadhav, S. (2022). API Testing Using Postman Tool. International Journal for Research in Applied Science and Engineering Technology, 10(12), 841-843. DOI:10.22214/ijraset.2022.48030.

Ndaru, A. D., Aljawari, M., & Setiawan, M. Y. (2025). Implementasi REST API Pengelolaan Data Penduduk Multi-Desa Berbasis Laravel dengan Autentikasi Sanctum. Merkurius: Jurnal Riset Sistem Informasi dan Teknik Informatika, 3(4), 325-335. DOI:10.61132/merkurius.v3i4.997.