Autentikasi di API (Laravel Sanctum)

Tugas Pendahuluan

1. Apa fungsi dari Laravel Sanctum?
2. Bagaimana cara kerja autentikasi dengan Laravel Sanctum secara umum?
3. Tuliskan perintah artisan yang digunakan untuk menginstal Laravel Sanctum!
4. Untuk apa token digunakan dalam autentikasi API?
5. Apa fungsi middleware auth:sanctum dalam route API Laravel?

Dasar Teori (Laravel Sanctum)

a) Laravel Sanctum adalah package resmi dari Laravel yang digunakan untuk Autentikasi API menggunakan token (API Token Authentication).

b) Cara kerja Sanctum adalah memberi "kunci akses" (token) ke user yang sudah login, agar user bisa mengakses endpoint API (route API) tertentu tanpa harus login ulang setiap kali.

c) Alur proses Laravel Sanctum:
   - User melakukan login ke API menggunakan email dan password.
   - Jika login berhasil, sistem akan membuat token unik (biasanya berbentuk string panjang acak).
   - Token ini disimpan oleh user (biasanya di frontend atau aplikasi mobile).
   - Setiap kali user mengakses endpoint API (route API) yang dilindungi, token dikirim melalui Authorization Header:
     ```
     Authorization: Bearer <token>
     ```
   - Laravel akan memeriksa token tersebut:
     - Jika valid → akses diizinkan
     - Jika tidak valid → respon error "401 Unauthorized"
   - Contoh: Misalkan kita ingin melindungi route `/api/mahasiswa` agar hanya bisa diakses user yang login. Maka alurnya:
     - User login → dapatkan token
     - Token disertakan di header permintaan API
     - Laravel verifikasi token
     - Jika valid → kirim response data mahasiswa
     - Jika tidak valid → kirim error `401 Unauthorized`

d) Struktur Komponen Sanctum di Laravel
   Setelah Sanctum diinstal, beberapa komponen penting akan digunakan:
   - `config/sanctum.php` → konfigurasi Sanctum

- `personal_access_tokens` table → tempat menyimpan token user
- Middleware `auth:sanctum` → digunakan untuk melindungi route API
- Token Management Methods → disediakan otomatis oleh Laravel pada model `User`

e) Contoh respon JSON

- Saat Login:

```
{
  "status": "success",
  "user": {
    "id": 1,
    "name": "Admin",
    "email": "admin@gmail.com"
  },
  "token": "1|h5Zq9xLhfdK7E5H..."
}
```

- Saat Akses Data:
  - Header

```
Authorization: Bearer 1|h5Zq9xLhfdK7E5H...
```

  - Response

```
{
  "message": "Data Mahasiswa",
  "data": [
    {"nim": "2023001", "nama": "Andi", "prodi": "Sistem Informasi"},
    {"nim": "2023002", "nama": "Budi", "prodi": "Informatika"}
  ]
}
```

Jurnal (Auth API dengan Laravel Sanctum)

Step 1 — Instal Laravel Sanctum

- Install Laravel sactum

```
composer require laravel/sanctum
```

Perintah ini akan menginstal package resmi Laravel Sanctum ke dalam proyek.

- Salin file konfigurasi Sanctum ke folder `config/`. dengan perintah berikut:

```
php artisan vendor:publish --
provider="Laravel\Sanctum\SanctumServiceProvider"
```

Step 2 — Jalankan Migrasi Sanctum

- Setelah Sanctum terinstal, jalankan perintah berikut untuk membuat tabel `personal_access_tokens` di database:

```
php artisan migrate
```

Tabel ini berfungsi untuk menyimpan token yang dibuat setiap kali user login.

Step 3 — Tambahkan Trait ke Model User

- Buka file `app/Models/User.php`, lalu tambahkan trait berikut:

```
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
}
```

Trait `HasApiTokens` memberikan kemampuan pada model `User` untuk membuat, mengelola, dan menghapus token login pengguna.

Step 4 — Registrasi Middleware Sanctum

- Buka file `config/sanctum.php` dan pastikan pengaturan defaultnya tidak diubah dulu. Lalu buka file `app/Http/Kernel.php` dan pastikan middleware berikut sudah terdaftar di grup `api`:

```
'api' => [
    \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
    'throttle:api',
    \Illuminate\Routing\Middleware\SubstituteBindings::class,
],
```

Ket: Sejak Laravel 11, file `Kernel.php` sudah dihapus dan middleware diatur otomatis menggunakan `bootstrap/app.php` jadi codenya seperti berikut:

```
->withMiddleware(function (Middleware $middleware): void {
    $middleware->api(prepend: [

\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
    ]);

    $middleware->alias([
        'role' => \App\Http\Middleware\RoleMiddleware::class,
    ]);
})
```

Step 5 — Buat Route untuk Auth di API

- Buka file `routes/api.php` lalu tambahkan:

```
use App\Http\Controllers\Api\AuthController;

Route::post('/register', [AuthController::class, 'register']);
Route::post('/login', [AuthController::class, 'login']);
Route::middleware('auth:sanctum')->get('/profile',
[AuthController::class, 'profile']);
Route::middleware('auth:sanctum')->post('/logout',
[AuthController::class, 'logout']);
```

Ket:

- `/register` → membuat akun baru
- `/login` → user login dan mendapatkan token

- `/profile` → hanya bisa diakses jika user login
- `/logout` → menghapus token (logout)

Step 6 — Buat Controller Auth

- Buat controller dengan perintah:

```
php artisan make:controller Api/AuthController
```

- Kemudian buka file `app/Http/Controllers/Api/AuthController.php` dan isi dengan kode berikut:

```php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class AuthController extends Controller
{
    // REGISTER
    public function register(Request $request)
    {
        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        return response()->json([
            'status' => 'success',
            'message' => 'User berhasil dibuat',
            'user' => $user
        ]);
    }

    // LOGIN
    public function login(Request $request)
    {
        $user = User::where('email', $request->email)->first();

        if (!$user || !Hash::check($request->password, $user->password))
{
            return response()->json(['message' => 'Login gagal'], 401);
        }

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json([
            'status' => 'success',
            'message' => 'Login berhasil',
            'token' => $token,
            'user' => $user
        ]);
    }
```

```php
    // PROFILE (Hanya bisa diakses jika login)
    public function profile(Request $request)
    {
        return response()->json([
            'status' => 'success',
            'user' => $request->user()
        ]);
    }

    // LOGOUT
    public function logout(Request $request)
    {
        $request->user()->currentAccessToken()->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Logout berhasil'
        ]);
    }
}
```

Step 7 — Uji API di Postman

- Register
  Method : POST
  URL : `http://127.0.0.1:8000/api/register`
  Body (form-data / JSON):

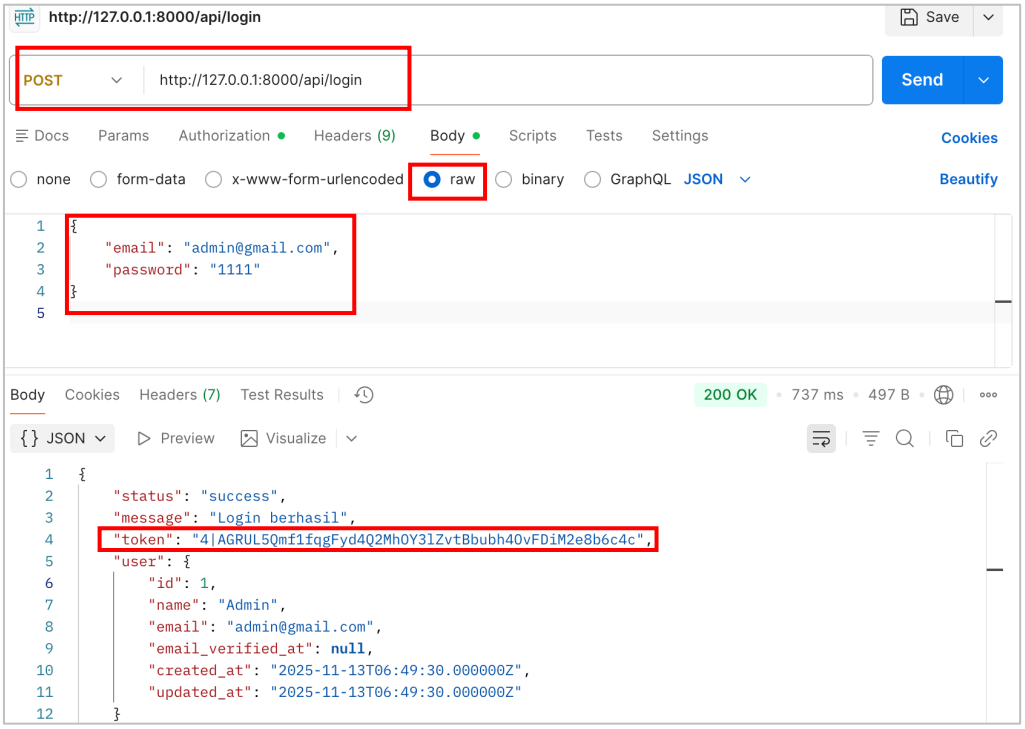
```json
{
    "name": "Admin",
    "email": "admin@gmail.com",
    "password": "12345678"
}
```

- Login
  Method : POST
  URL : `http://127.0.0.1:8000/api/login`
  Body :

```json
{
    "email": "admin@gmail.com",
    "password": "12345678"
}
```

  Response :

```json
{
    "status": "success",
    "token": "1|h9s8HDk3929HsK...",
    "user": {
        "id": 1,
        "name": "Admin",
        "email": "admin@gmail.com"
    }
}
```

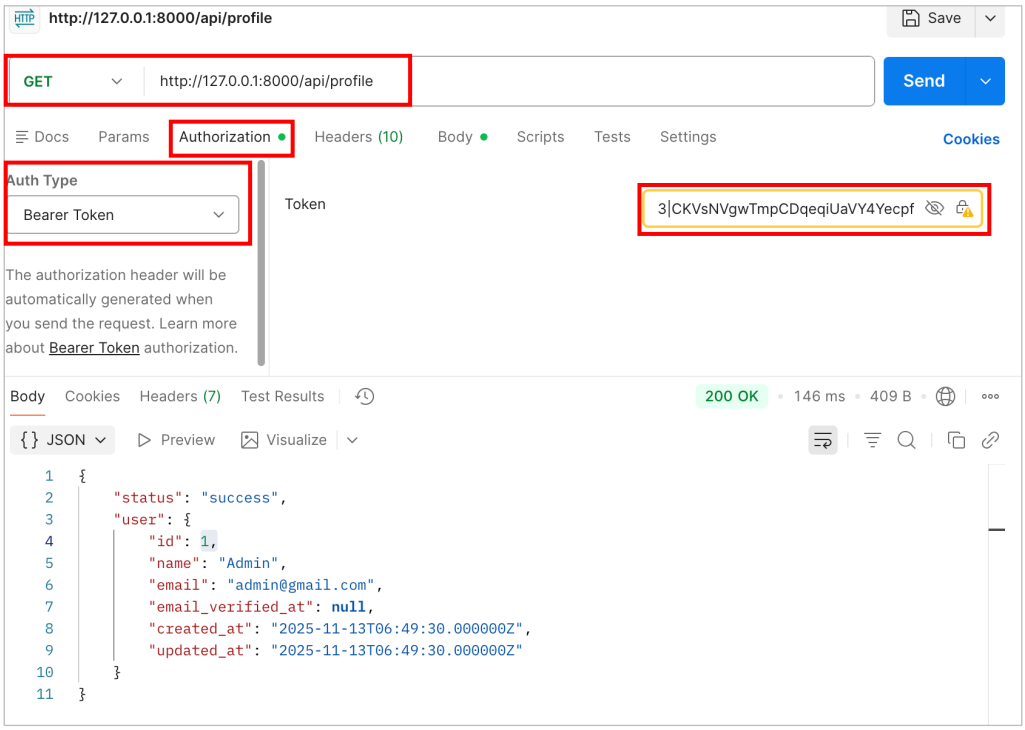


– Profile (Cek Login)

Method : **GET**

URL : `http://127.0.0.1:8000/api/profile`

Authorization : 4|AGRUL5Qmf1fqgFyd4Q2MhOY3lZvtBbubh4OvFDiM2e8b6c4c



- Logout
  Method : **POST**
  URL : `http://127.0.0.1:8000/api/logout`
  Authorization : 4|AGRUL5Qmf1fqgFyd4Q2MhOY3lZvtBbubh4OvFDiM2e8b6c4c

Latihan (Multi Role & Hak Akses di API)

Pada Latihan ini kita akan membelajari bagaimana caranya membuat sistem login API berbasis token menggunakan Sanctum dan menambahkan role pada user (misal: admin, dosen, mahasiswa) sehingga nantinya penggunaan API dibatasi berdasarkan role.

Step 1 - Tambahkan Kolom Role ke Tabel User (jika belum ada)

- Buka file migrasi user di:

```
database/migrations/2014_10_12_000000_create_users_table.php
```

- Tambahkan kolom `role` seperti ini:

```
$table->string('role')->default('mahasiswa');
```

Setiap user sekarang punya atribut `role` yang secara default bernilai mahasiswa.

- Lalu jalankan migrasi ulang:

```
php artisan migrate:refresh
```

Step 2 - Modifikasi Controller Auth

- Tambahkan logika role ke `AuthController` dengan menambahkan kode berikut:
  `app/Http/Controllers/Api/AuthController.php`

```
public function register(Request $request)
{
    $user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'role' => $request->role ?? 'mahasiswa', // default
mahasiswa
    ]);

    return response()->json([
        'status' => 'success',
        'message' => 'User berhasil dibuat',
        'user' => $user
    ]);
}
```

Ket:
User bisa mendaftar dengan role tertentu, misalnya `admin`, `dosen`, atau `mahasiswa`. Jika role tidak dikirim, otomatis akan menjadi **mahasiswa**.

Step 3 - Buat Middleware untuk Role

- Buat middleware agar API bisa membatasi akses sesuai role.

```
php artisan make:middleware RoleMiddleware
```

- Isi middleware dengan kode berikut:
  `app/Http/Middleware/RoleMiddleware.php`

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class RoleMiddleware
{
    public function handle(Request $request, Closure $next,
...$roles)
    {
        $user = $request->user();

        if (!$user || !in_array($user->role, $roles)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Anda tdk ada hak akses.'
            ], 403);
        }

        return $next($request);
    }
}
```

Ket*:*
Middleware ini mengecek apakah user yang sedang login memiliki role yang diizinkan. Jika tidak, maka API mengembalikan respon `403 Forbidden`.

Step 4 - Registrasikan Middleware di Kernel

- Buka file `app/Http/Kernel.php` dan tambahkan baris berikut di dalam `$routeMiddleware`:

```
protected $routeMiddleware = [
    // ...
    'role' => \App\Http\Middleware\RoleMiddleware::class,
];
```

Ket: Sejak Laravel 11, file `Kernel.php` sudah dihapus dan middleware diatur otomatis menggunakan `bootstrap/app.php` sehingga codenya seperti berikut:

```
->withMiddleware(function (Middleware $middleware): void {
    $middleware->api(prepend: [
```

```
\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::
class,
    ]);

    $middleware->alias([
        'role' => \App\Http\Middleware\RoleMiddleware::class,
    ]);
})
```

Step 5 - Buat Controller API

- Buat controller API untuk mengatur akses API sesuai role

```
php artisan make:controller Api/AccessController
```

kemudian isi controller tersebut dengan beberapa method seperti berikut:
app/Http/Controllers/Api/AccessController.php

```
<?php

namespace App\Http\Controllers\Api;

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

class AccessController extends Controller
{
    // Hanya Mahasiswa
    public function mahasiswa(Request $request)
    {
        $user = $request->user();

        if ($user->role !== 'mahasiswa') {
            return response()->json(['message' => 'Akses hanya untuk
mahasiswa!'], 403);
        }

        $data = Mahasiswa::all();
        return response()->json([
            'status' => 'success',
            'data' => $data
        ]);
    }

    // Hanya Dosen
    public function dosen(Request $request)
    {
        $user = $request->user();
```

```
        if ($user->role !== 'dosen') {
            return response()->json(['message' => 'Akses hanya untuk
dosen!'], 403);
        }

        $data = Dosen::all();
        return response()->json([
            'status' => 'success',
            'data' => $data
        ]);
    }

    // Hanya Admin
    public function admin(Request $request)
    {
        $user = $request->user();

        if ($user->role !== 'admin') {
            return response()->json(['message' => 'Akses hanya untuk
admin!'], 403);
        }

        $mahasiswa = Mahasiswa::all();
        $dosen = Dosen::all();

        return response()->json([
            'status' => 'success',
            'mahasiswa' => $mahasiswa,
            'dosen' => $dosen
        ]);
    }
}
```

Step 6 - Tambahkan Route API

– Tambahkan route berikut ini di dalam `routes/api.php`

```
use App\Http\Controllers\Api\AccessController;

// Mahasiswa hanya bisa akses tabel mahasiswa
Route::middleware(['auth:sanctum', 'role:mahasiswa'])-
>get('/mahasiswa', [AccessController::class, 'mahasiswa']);

// Dosen hanya bisa akses tabel dosen
Route::middleware(['auth:sanctum', 'role:dosen'])->get('/dosen',
[AccessController::class, 'dosen']);

// Admin bisa akses semuanya
Route::middleware(['auth:sanctum', 'role:admin'])->get('/admin',
[AccessController::class, 'admin']);
```

Step 7 - Simulasi Pengujian di Postman

- Login dulu untuk mendapatkan token

  Endpoint:

```
POST http://127.0.0.1:8000/api/login
```

  Misal login menggunakan role admin maka masukan user dan password admin pada Body(JSON):

```
{
  "email": "admin@gmail.com",
  "password": "12345678"
}
```

  Maka ketika di send akan menghasilkan response berikut:

```
{
  "token": "1|XyzAbc123..."
}
```

  Gunakan token tersebut di Header setiap request:

```
Authorization: Bearer 1|XyzAbc123...
```

- Contoh hasil JSON sesuai role

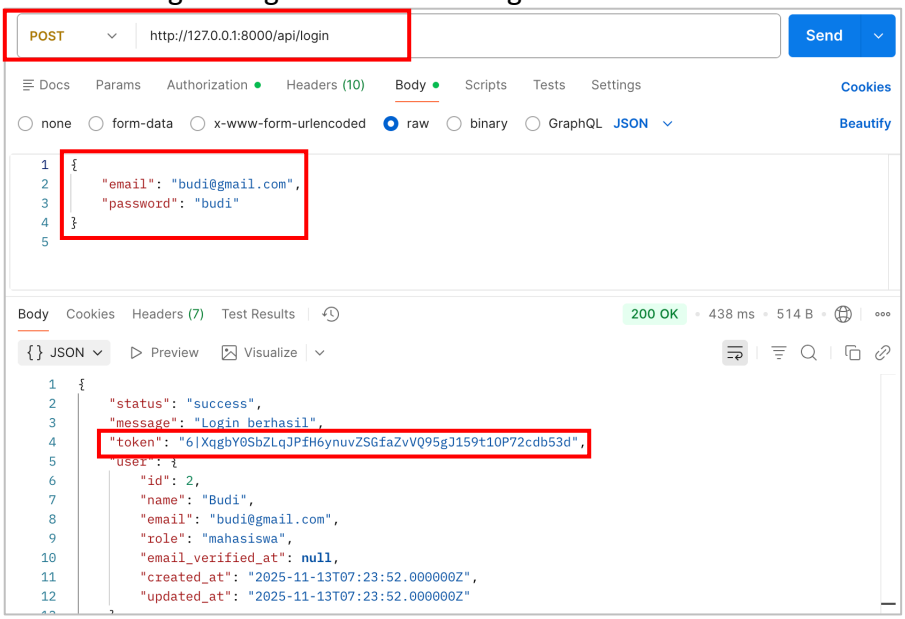
Role	Endpoint	Hasil
Admin	GET /api/admin	Dapat data mahasiswa dan dosen
Dosen	GET /api/dosen	Dapat data tabel dosen
Mahasiswa	GET /api/mahasiswa	Dapat data tabel mahasiswa
Role berbeda	Coba akses endpoint yang bukan miliknya	akan muncul pesan *403 Forbidden*

Step 8 – Percobaan Login dan Akses Data

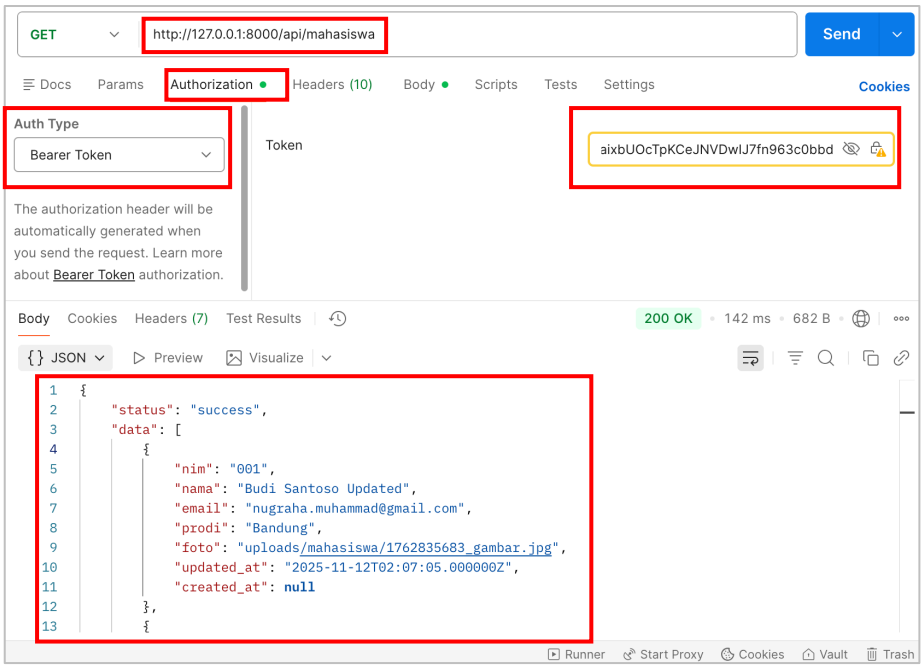
- Register 3 data dengan role yang berbeda-beda. Cara register seperti pada percobaan sebelumnya.

Modify	id	name	email	role
edit	1	Admin	admin@gmail.com	admin
edit	2	Budi	budi@gmail.com	mahasiswa
edit	5	Bill	bill@gmail.com	dosen

- Kemudian login dengan user budi sebagai mahasiswa:



- Setelah berhasil login akses data mahasiswa seperti gambar berikut:



– Kemudian coba untuk mengakses data dosen, maka akses di tolak karena role mahasiswa tidak punya akses ke data dosen.