

# PRAKTIK PENGEMBANGAN BACK END

13 November 2024

## AUTENTIKASI DI LARAVEL DENGAN LARAVEL SANCTUM

---

a. Apa itu autentikasi

Autentikasi adalah proses verifikasi identitas pengguna atau sistem yang mencoba mengakses aplikasi atau layanan. Dalam konteks aplikasi web, autentikasi biasanya mengharuskan pengguna untuk memberikan kredensial (seperti username dan password) untuk membuktikan bahwa mereka adalah pemilik akun tersebut. Setelah berhasil diverifikasi, pengguna mendapatkan akses ke sistem atau sumber daya yang diizinkan.

b. Apa Itu bearer token

Bearer Token adalah jenis token autentikasi yang digunakan untuk memberi akses ke sumber daya API. Dengan Bearer Token, klien menyertakan token dalam header Authorization di setiap permintaan, dan server memverifikasi token tersebut untuk memberikan akses ke resource yang diminta. Penggunaan Bearer Token populer dalam aplikasi yang menggunakan protokol OAuth 2.0 dan API berbasis REST.

c. Apa itu Laravel sanctum

Laravel Sanctum adalah paket dari Laravel yang digunakan untuk mengelola autentikasi berbasis token dalam aplikasi Laravel, terutama untuk aplikasi SPA (Single Page Application), aplikasi mobile, atau API sederhana. Sanctum menyediakan personal access

tokens dan cookie-based session untuk memungkinkan autentikasi berbasis session yang aman.

d. Kelebihan Laravel sanctum

Kelebihan Laravel Sanctum antara lain:

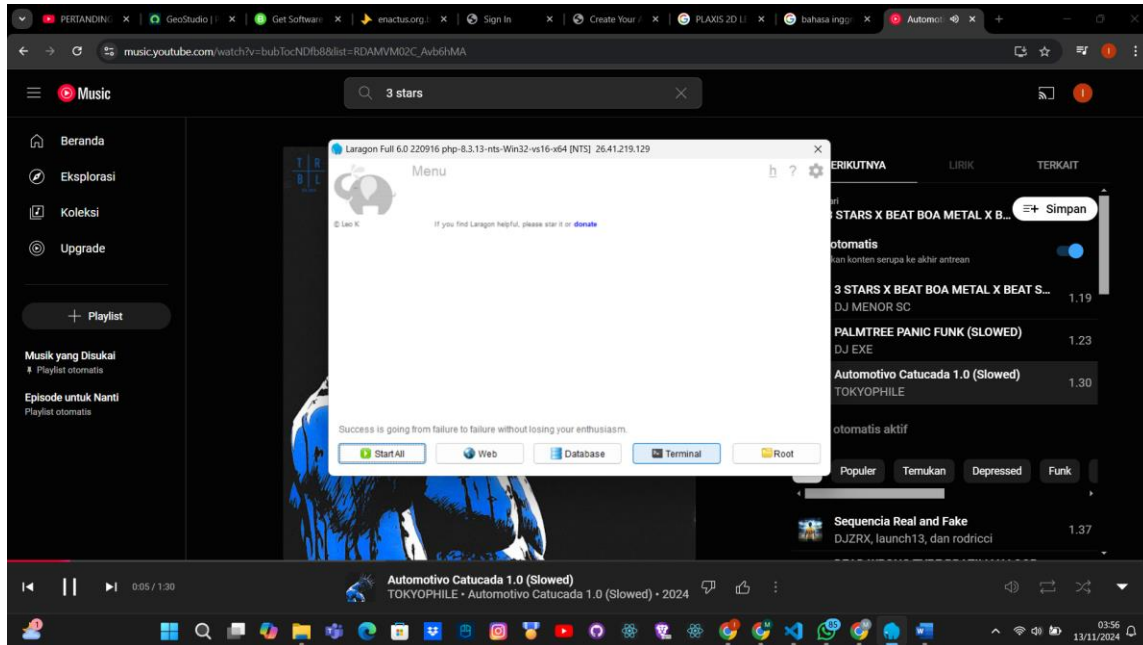
- **Kemudahan Penggunaan:** Sanctum mudah diintegrasikan dalam aplikasi Laravel tanpa konfigurasi kompleks.
- **Fleksibilitas:** Sanctum mendukung token berbasis API dan autentikasi berbasis cookie, sehingga dapat digunakan baik untuk API maupun aplikasi SPA.
- **Keamanan:** Sanctum menyediakan metode stateful yang aman untuk SPA dan autentikasi berbasis token yang cocok untuk aplikasi mobile.
- **Manajemen Token Sederhana:** Sanctum menyediakan token personal yang mudah dikelola dan dicabut tanpa mengharuskan aplikasi mengimplementasikan OAuth penuh.

e. Konsep Auth

Konsep Auth atau autentikasi adalah mekanisme yang memungkinkan aplikasi untuk memverifikasi identitas pengguna sebelum memberi akses ke sistem atau data tertentu. Di Laravel, konsep Auth meliputi beberapa fitur, seperti login, registrasi, reset password, dan manajemen session. Laravel mendukung berbagai metode autentikasi, seperti berbasis session, token (menggunakan Laravel Sanctum atau Passport), dan OAuth2.

## f. Proses Pembuatan Auth di Laravel dengan menggunakan Laravel sanctum

### 1. Pertama, bukalah laragon dan pergi ke console/Terminalnya



### 2. Kemudian pergilah ke project kalian masing masing di terminalnya:

```
Cmder

D:\laragon\www
λ cd D:\materi asdos\backend\P1

D:\materi asdos\backend\P1
λ |
```

### 3. Kemudian masukkan command ini, untuk memanggil “composer require laravel/sanctum”

```
D:\laragon\www
λ cd D:\materi asdos\backend\P1

D:\materi asdos\backend\P1
λ composer require laravel/sanctum
Using version ^4.0 for laravel/sanctum
./composer.json has been updated

Cmder
D:\laragon\www
λ cd D:\materi asdos\backend\P1
D:\materi asdos\backend\P1
λ composer require laravel/sanctum
Using version ^4.0 for laravel/sanctum
./composer.json has been updated
Running composer update laravel/sanctum
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/sanctum (v4.0.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/sanctum (v4.0.3)
- Installing laravel/sanctum (v4.0.3): Extracting archive
Generating optimized autoload files
> illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

[INFO] Discovering packages.

laravel/pail ..... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE

78 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

[INFO] No publishable resources for tag [laravel-assets].

Found 2 security vulnerability advisories affecting 2 packages.
Run composer audit for a full list of advisories.

D:\materi asdos\backend\P1
λ |
```

4. Setelah itu, publikasikan konfigurasi Sanctum dan migrasikan tabel untuk menyimpan token, dengan command ini:

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

```
D:\materi asdos\backend\P1
λ php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

[INFO] Publishing assets.

Copying directory [D:\materi asdos\backend\P1\vendor\laravel\sanctum\database\Migrations] to [D:\materi asdos\backend\P1\database\Migrations] DONE
Copying file [D:\materi asdos\backend\P1\vendor\laravel\sanctum\config\sanctum.php] to [D:\materi asdos\backend\P1\config\sanctum.php] ..... DONE

D:\materi asdos\backend\P1
λ php artisan migrate

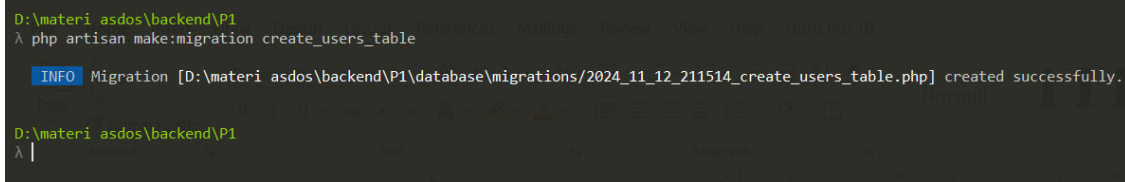
[INFO] Running migrations.

2024_11_12_210613_create_personal_access_tokens_table ..... 25.19ms DONE

D:\materi asdos\backend\P1
λ |
```

5. Kemudian buatlah table users nya di migration :

-----  
php artisan make:migration create\_users\_table  
-----

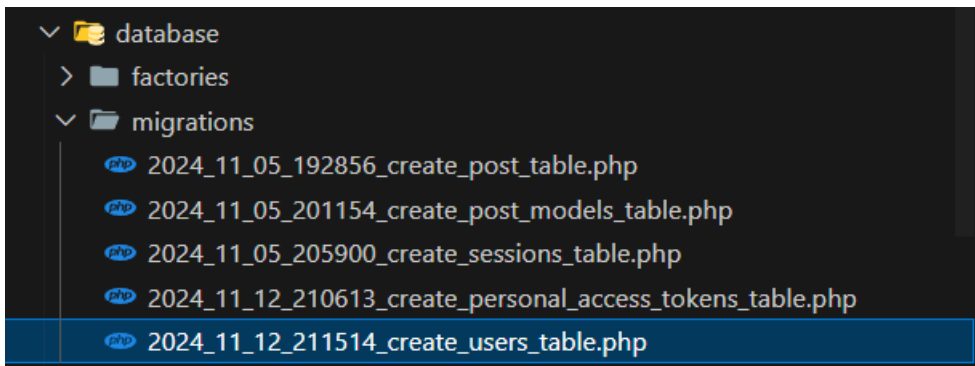


```
D:\materi asdos\backend\P1
λ php artisan make:migration create_users_table

[INFO] Migration [D:\materi asdos\backend\P1\database\Migrations\2024_11_12_211514_create_users_table.php] created successfully.

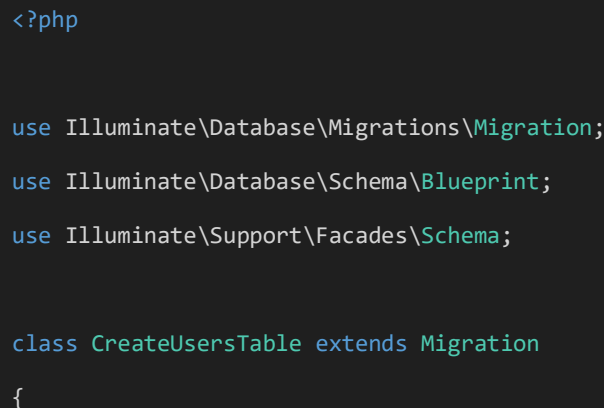
D:\materi asdos\backend\P1
λ |
```

6. Kemudian jika sudah dibuat, kemudian pergi ke filenya di database/migration



```
▼ database
  > factories
  ▼ migrations
    2024_11_05_192856_create_post_table.php
    2024_11_05_201154_create_post_models_table.php
    2024_11_05_205900_create_sessions_table.php
    2024_11_12_210613_create_personal_access_tokens_table.php
    2024_11_12_211514_create_users_table.php
```

7. Kemudian isialah dengan kode ini



```
<?php

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

class CreateUsersTable extends Migration
{
```

```

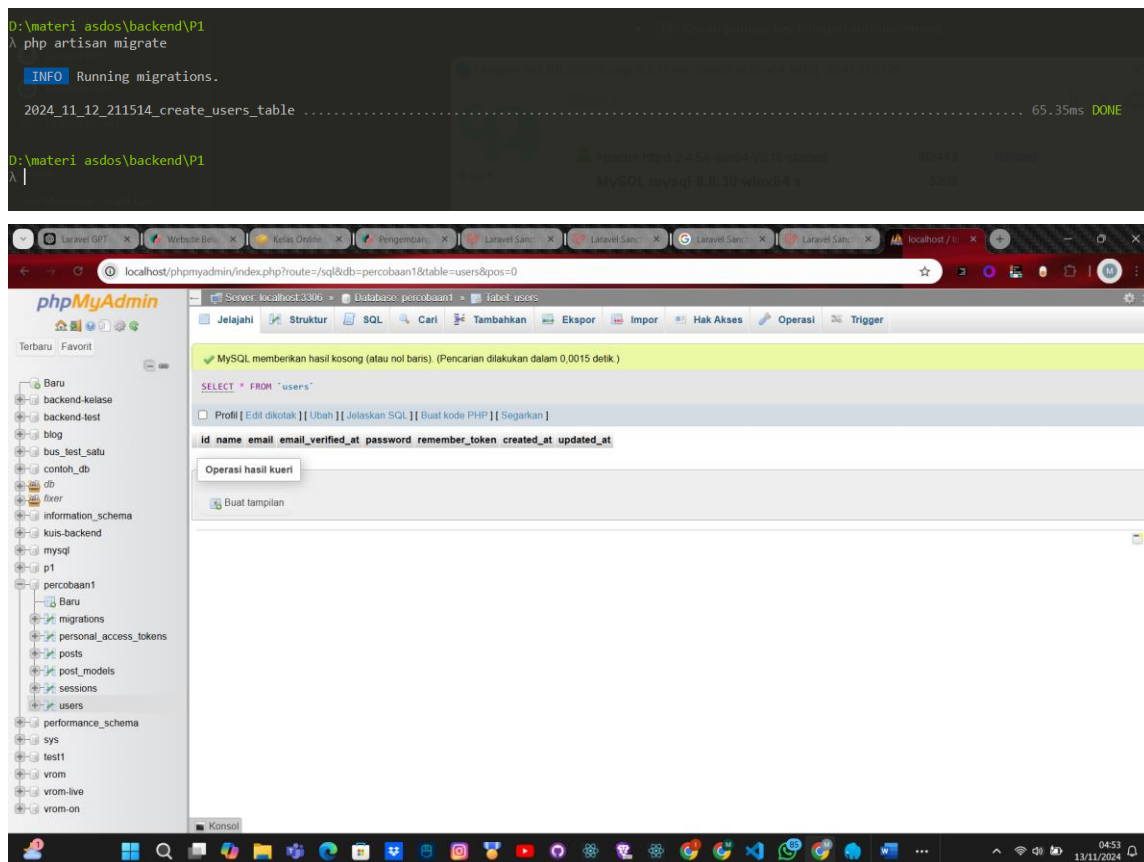
/**
 * Jalankan migrasi.
 *
 * @return void
 */
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id(); // Primary key ID auto-increment
        $table->string('name'); // Nama user
        $table->string('email')->unique(); // Email unik
        $table->timestamp('email_verified_at')->nullable(); // Waktu
verifikasi email
        $table->string('password'); // Password yang terenkripsi
        $table->rememberToken(); // Token untuk fitur "remember me"
        $table->timestamps(); // created_at dan updated_at otomatis
    });
}

/**
 * Balikkan migrasi.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('users');
}
}

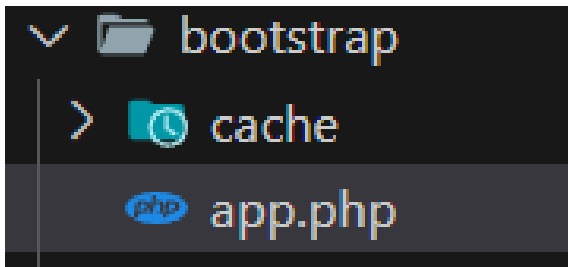
```

8. Kemudian lakukan migrations kembali di teminal laravelnya, dan cek databasenya apakah sudah terbuat atau belum:

-----  
php artisan migrate  
-----



9. Setelah itu, pergi ke Laravel Kembali, dan pergi ke bootstrap/app



a) Kemudian importlah laravel sanctumnya

```
use Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful;
```

```
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful;
```

b) Kemudian tambahkanlah "api" di bagian routingnya

```
api: __DIR__.'/../routes/api.php', // pastikan route api didefinisikan di sini
```

```
return Application::configure(basePath: dirname(path: __DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        api: __DIR__.'/../routes/api.php', // pastikan route api didefinisikan di sini
        health: 'up',
    ) <- #9-14 ->withRouting
```

c) Kemudian tambahkanlah middleware untuk menjalankan Laravel sanctumnya

```
// Tambahkan Sanctum middleware di grup 'api'
$middleware->group('api', [
    EnsureFrontendRequestsAreStateful::class,
    'throttle:api',
    \Illuminate\Routing\Middleware\SubstituteBindings::class
],
```

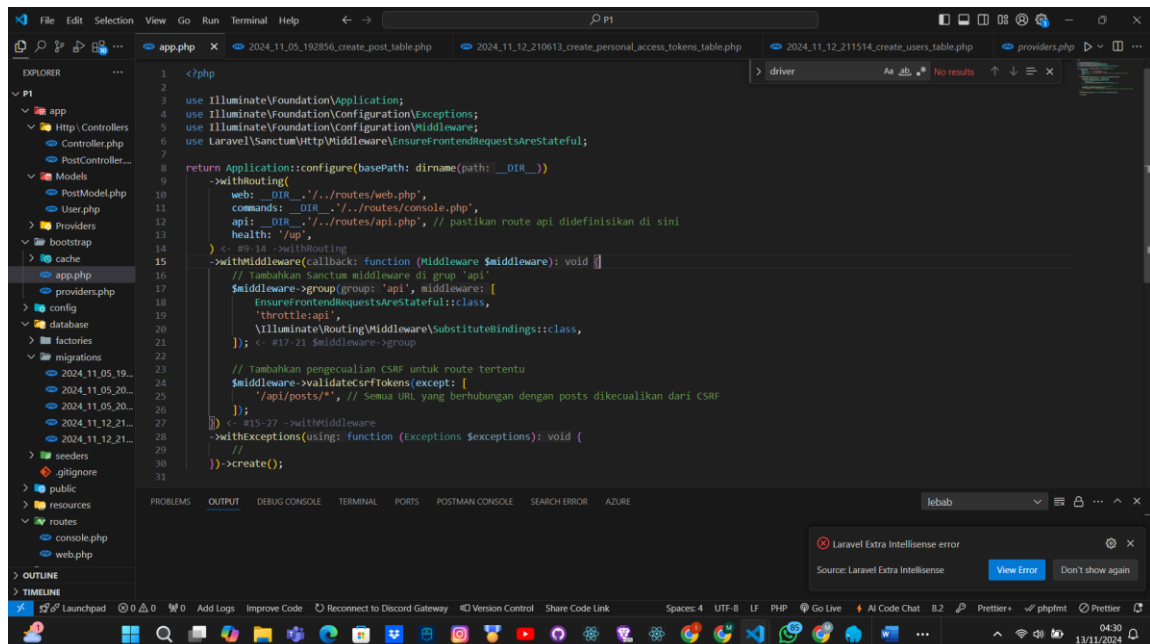


```

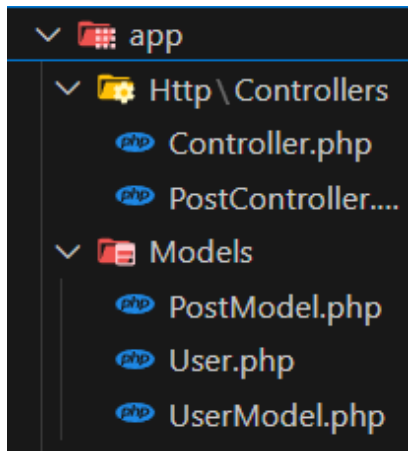
->withMiddleware(callback: function (Middleware $middleware): void {
    // Tambahkan Sanctum middleware di grup 'api'
    $middleware->group(group: 'api', middleware: [
        EnsureFrontendRequestsAreStateful::class,
        'throttle:api',
        \Illuminate\Routing\Middleware\SubstituteBindings::class,
    ]);
    <- #17-21 $middleware->group

```

Contoh :



10. Kemudian pergi ke app/models, dan buatlah UserModel disana



11. Setelah itu isilah UserModel.php dengan code ini

```
<?php

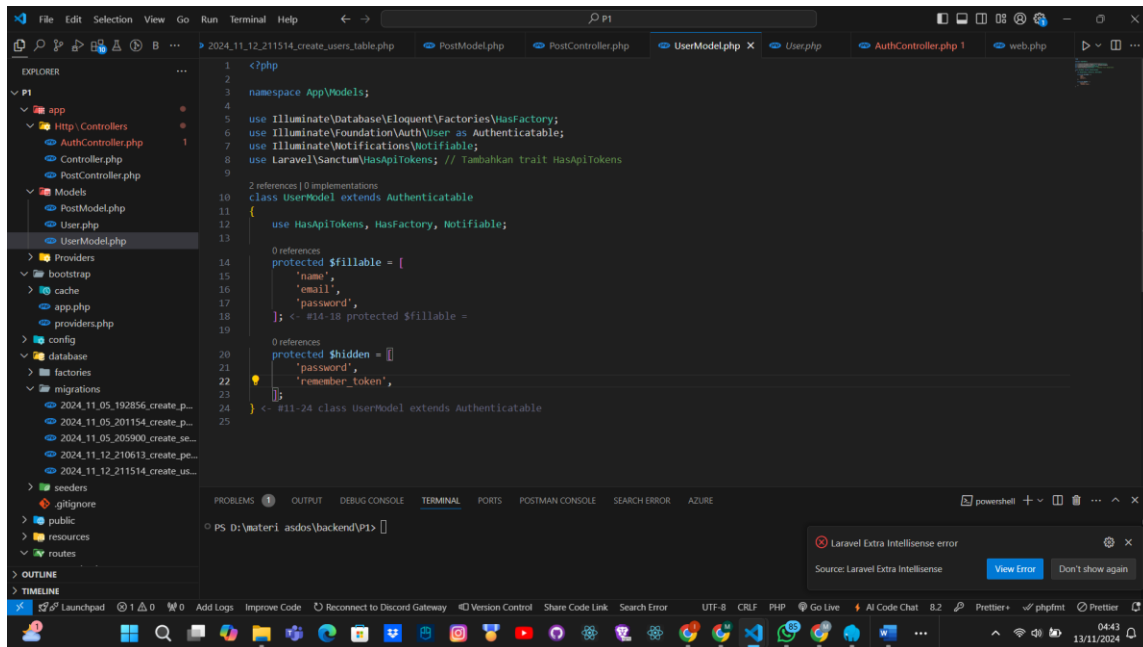
namespace App\Models;

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

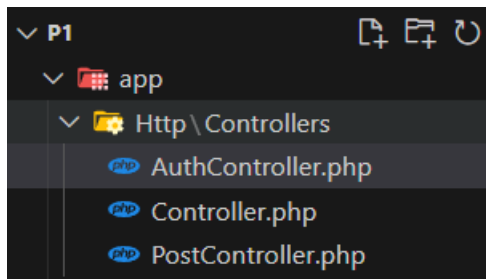
class UserModel extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

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

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



12. Setelah itu pergi ke app/http/controllerm, buatlah controllernya dengan nama “AuthController”



- Kemudian isikanlah dengan code ini

```
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    // Register User
    public function register(Request $request)
    {
        $request->validate([
```

```

        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:8|confirmed',
    ]);

    $user = UserModel::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
    ]);

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

    return response()->json([
        'message' => 'User registered successfully',
        'access_token' => $token,
        'token_type' => 'Bearer',
    ], 201);
}

// Login User
public function login(Request $request)
{
    $request->validate([
        'email' => 'required|string|email',
        'password' => 'required|string',
    ]);

    if (!Auth::attempt($request->only('email', 'password'))) {
        return response()->json(['message' => 'Invalid login
credentials'], 401);
    }

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

    return response()->json([
        'message' => 'Login successful',
        'access_token' => $token,
        'token_type' => 'Bearer',
    ], 200);
}

// Logout User
public function logout(Request $request)
{
    $request->user()->tokens()->delete();
}

```

```

return response()->json([
    'message' => 'Logout successful',
], 200);
}
}

```

```

<?php
namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    // Register User
    public function register(Request $request): JsonResponse|mixed
    {
        $request->validate(rules: [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
        ]); < #15-19 $request->validate

        $user = UserModel::create(attributes: [
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make(value: $request->password),
        ]); < #21-25 $user = UserModel::create

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

        return response()->json(data: [
            'message' => 'User registered successfully',
            'access_token' => $token,
            'token_type' => 'Bearer',
        ], status: 201); < #29-33 return response()->json
    } < #14-34 public function register(Request $request)

    // Login User
    public function login(Request $request): JsonResponse|mixed
    {
        $request->validate(rules: [
            'email' => 'required|string|email',
            'password' => 'required|string',
        ]);

        if (Auth::attempt(credentials: $request->only(keys: 'email', 'password'))) {
            return response()->json(data: ['message' => 'Invalid login credentials'], status: 401);
        }

        $user = Auth::user();
        $token = $user->createToken(name: 'auth_token')->plainTextToken; Undefined method 'createToken'.

        return response()->json(data: [
            'message' => 'Login successful',
            'access_token' => $token,
            'token_type' => 'Bearer',
        ], status: 200); < #51-55 return response()->json
    } < #38-56 public function login(Request $request)

    // Logout User
    public function logout(Request $request): JsonResponse|mixed
    {
        $request->user()->tokens()->delete();

        return response()->json(data: [
            'message' => 'Logout successful',
        ], status: 200);
    } < #60-66 public function logout(Request $request)
} < #11-67 class AuthController extends Controller

```

```

public function login(Request $request): JsonResponse|mixed
{
    $request->validate(rules: [
        'email' => 'required|string|email',
        'password' => 'required|string',
    ]);

    if (Auth::attempt(credentials: $request->only(keys: 'email', 'password'))) {
        return response()->json(data: ['message' => 'Invalid login credentials'], status: 401);
    }

    $user = Auth::user();
    $token = $user->createToken(name: 'auth_token')->plainTextToken; Undefined method 'createToken'.

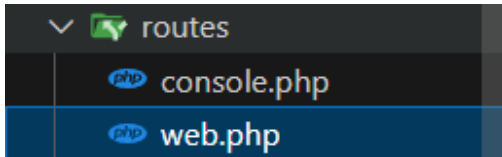
    return response()->json(data: [
        'message' => 'Login successful',
        'access_token' => $token,
        'token_type' => 'Bearer',
    ], status: 200); < #51-55 return response()->json
    } < #38-56 public function login(Request $request)

    // Logout User
    public function logout(Request $request): JsonResponse|mixed
    {
        $request->user()->tokens()->delete();

        return response()->json(data: [
            'message' => 'Logout successful',
        ], status: 200);
    } < #60-66 public function logout(Request $request)
} < #11-67 class AuthController extends Controller

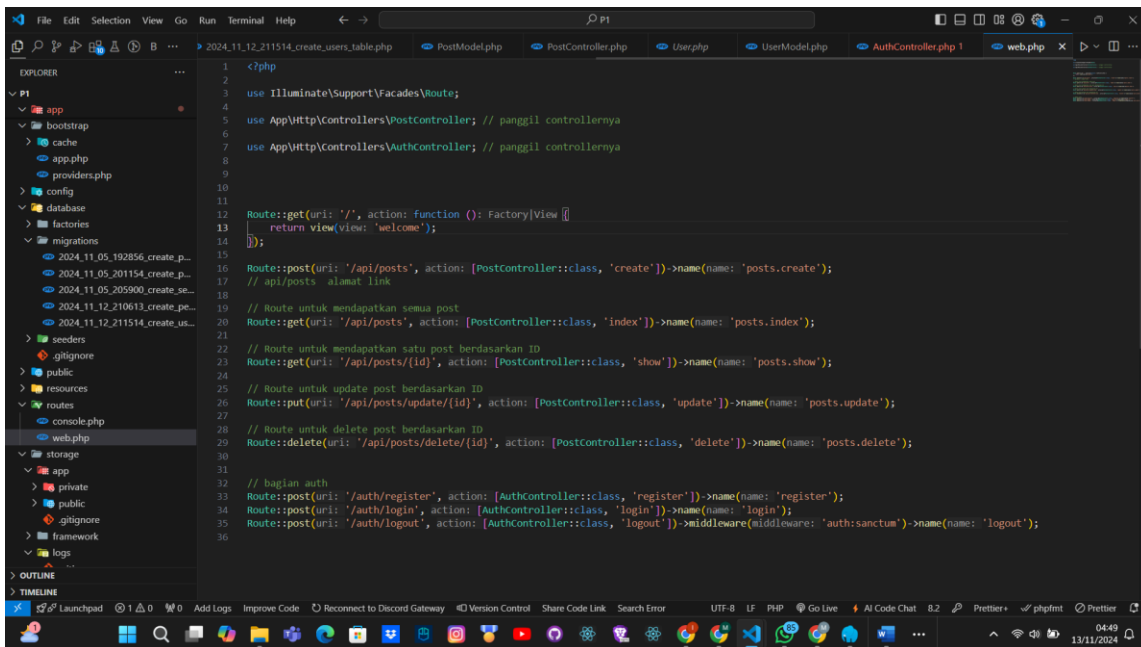
```

13. Kemudian setelah controller dibuat, kemudian buatlah routesnya, buka file routes di routes\web.php

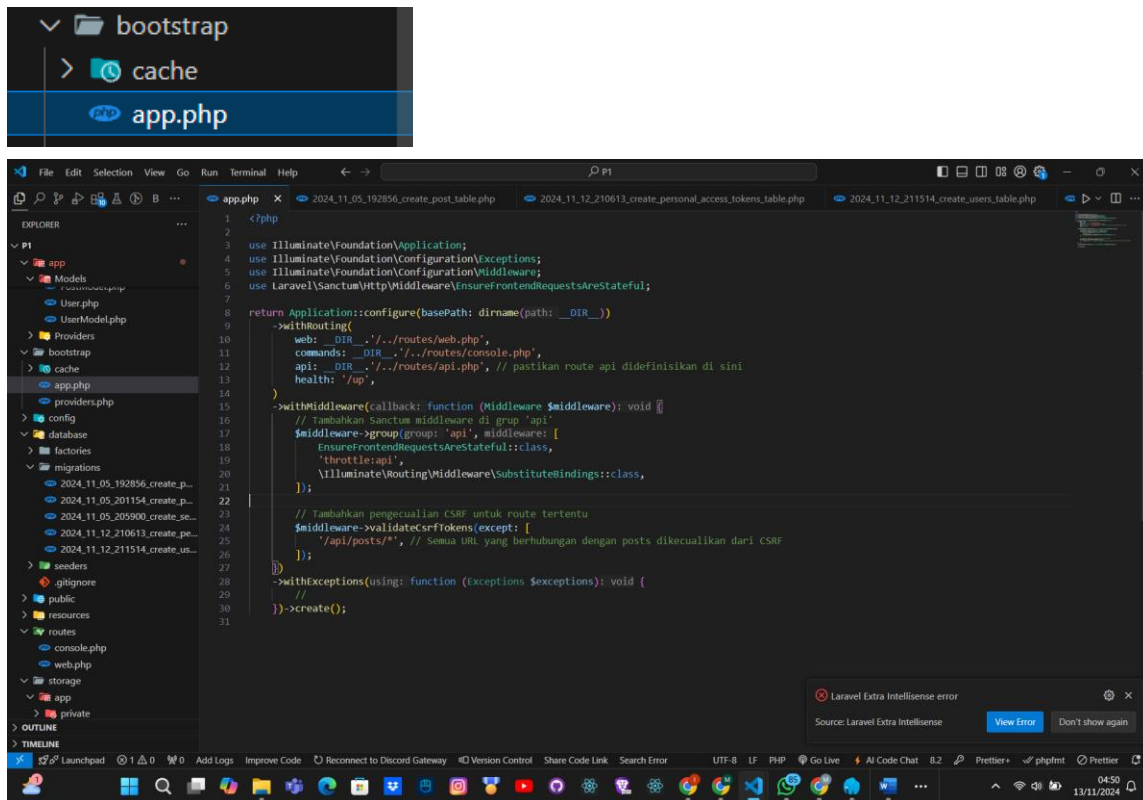


14. Kemudian jika sudah dibuat, bisa ditambahkan code routes untuk delete berdasarkan dari controller yang dibuat, contoh seperti ini :

```
use App\Http\Controllers\AuthController; // panggil controllernya
// bagian auth
Route::post('/auth/register', [AuthController::class, 'register'])->
>name('register');
Route::post('/auth/login', [AuthController::class, 'login'])->
>name('login');
Route::post('/auth/logout', [AuthController::class, 'logout'])->
>middleware('auth:sanctum')->name('logout');
```



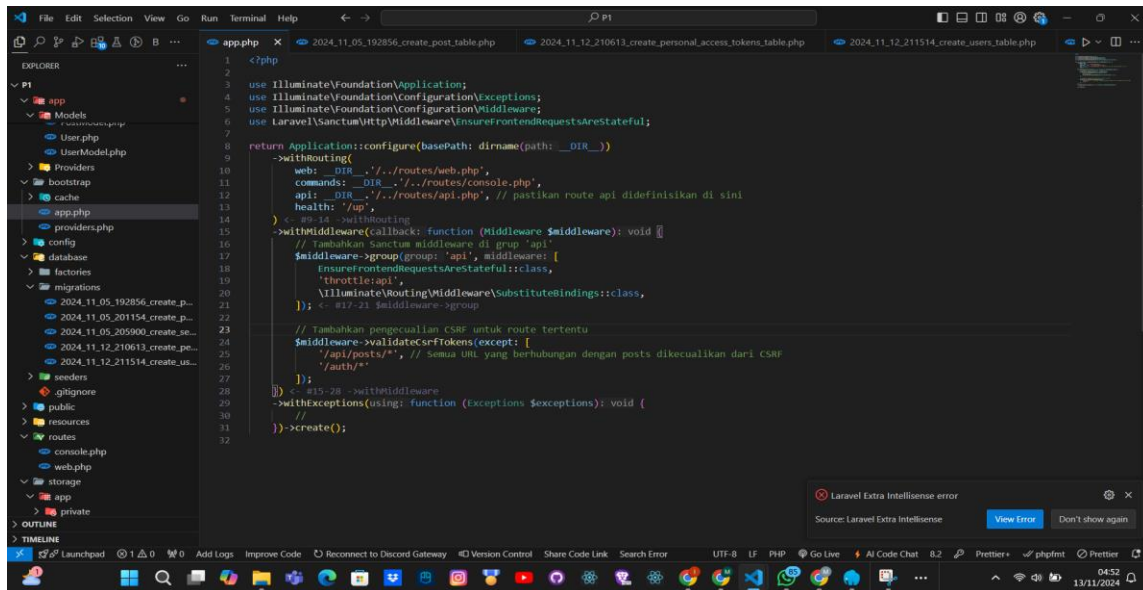
15. Setelah membuat routesnya, kemudian buka bootstrap\app.php



16. Setelah itu isillah bagian return yang withMiddleware, dan tambahkanlah routes yang baru saja dibuat untuk delete nya :

```
$middleware->validateCsrfTokens(except: [
    '/auth/*'
]);
```

Contoh penerapan :



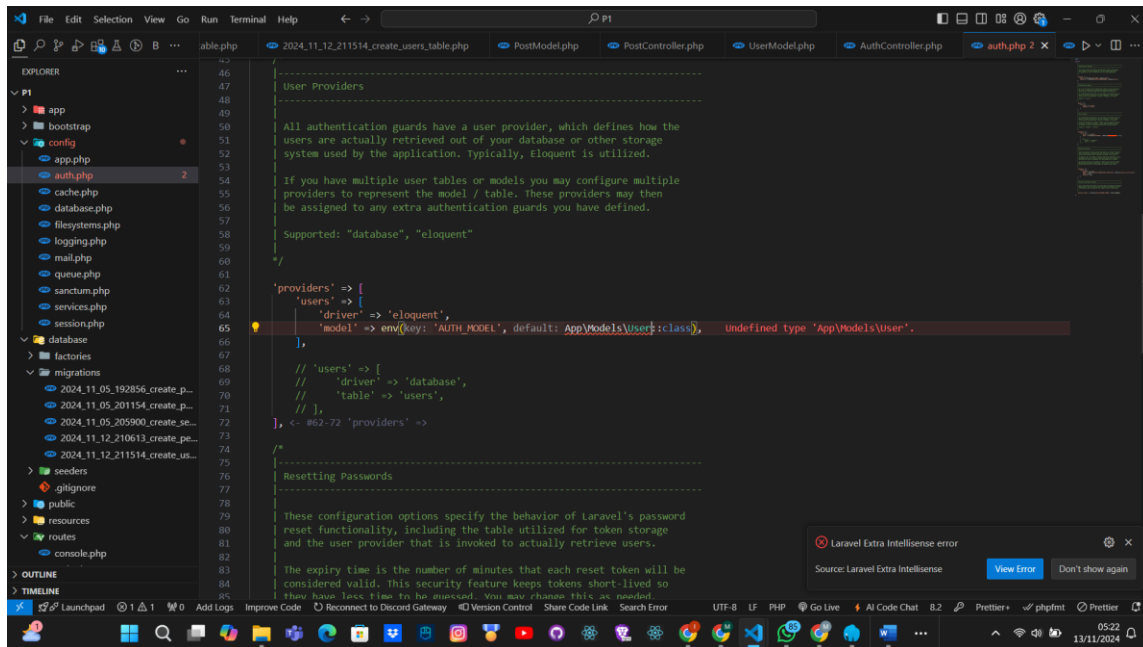
17. Kemudian gantilah Lokasi model di config/Auth :

```
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model' => env('AUTH_MODEL', App\Models\UserModel::class),
    ],

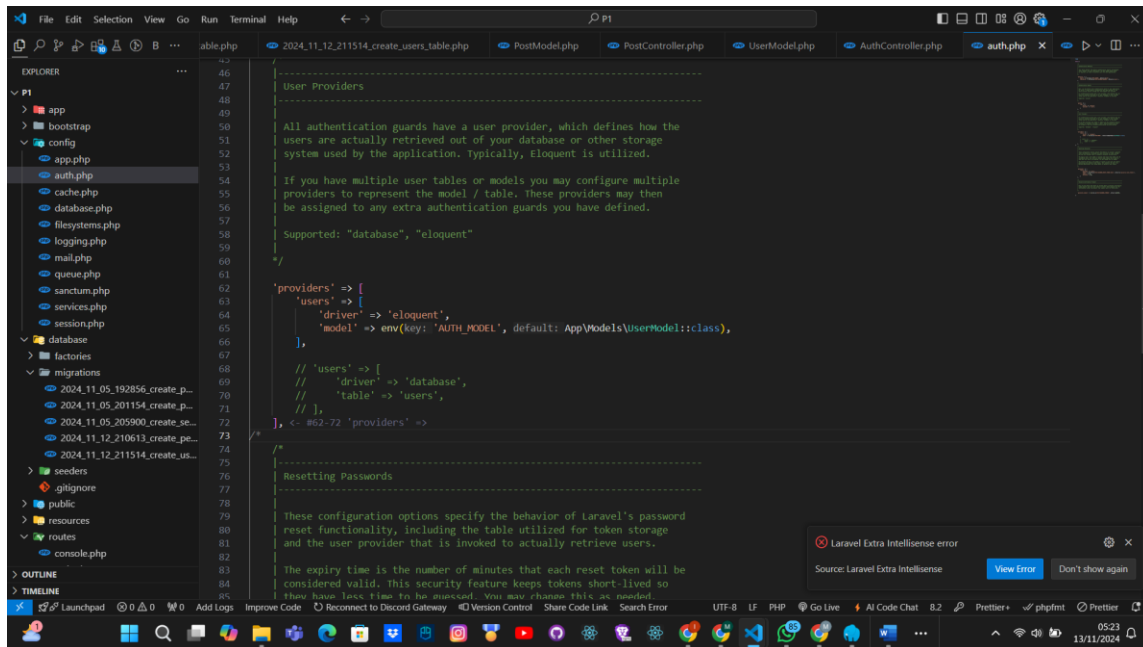
    // 'users' => [
    //     'driver' => 'database',
    //     'table' => 'users',
    // ],
],
```

Sebelum :





Sesudah :



18. Kemudian buka kembali terminal dan nyalakan project laravelnya dengan command :  
php artisan serve

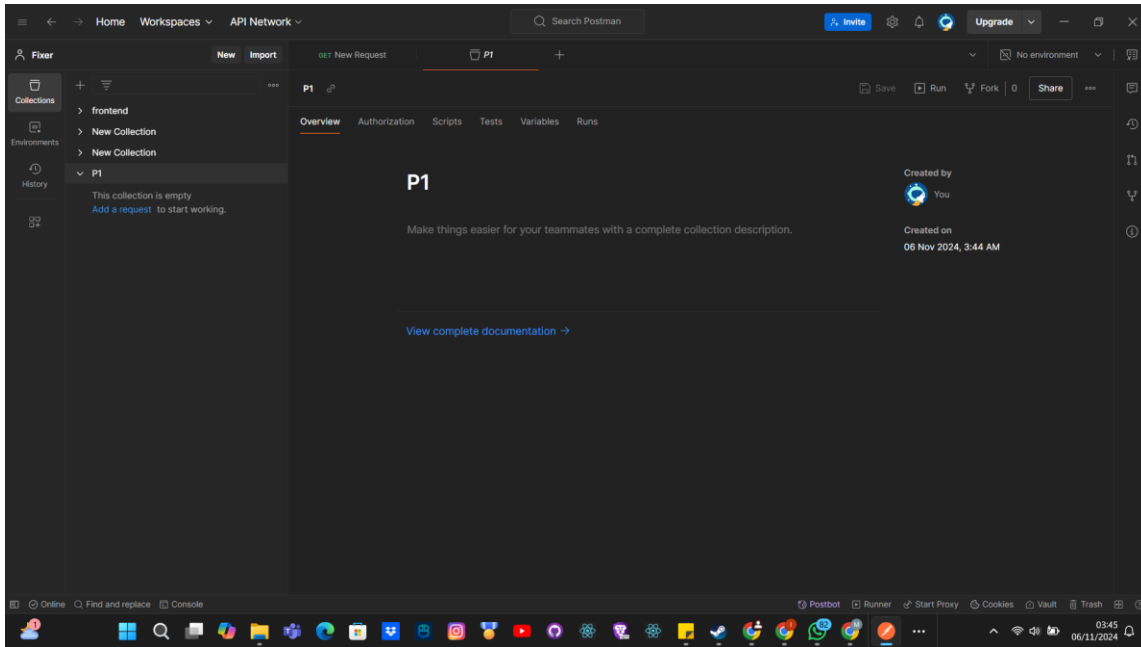
```
D:\materi asdos\backend\P1
λ php artisan serve
forking is not supported on this platform

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

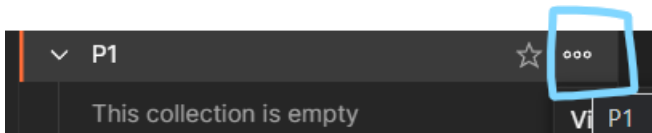
Press Ctrl+C to stop the server
```

19. Kemudian buka postman, setelah itu bukalah collection yang kalian sudah buat sebelumnya

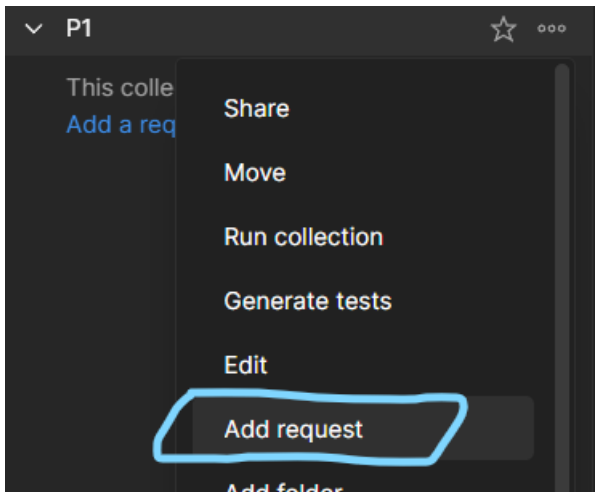
Contoh :



20. Setelah terbuat, klik titik tiga yang berada di sebelah nama project

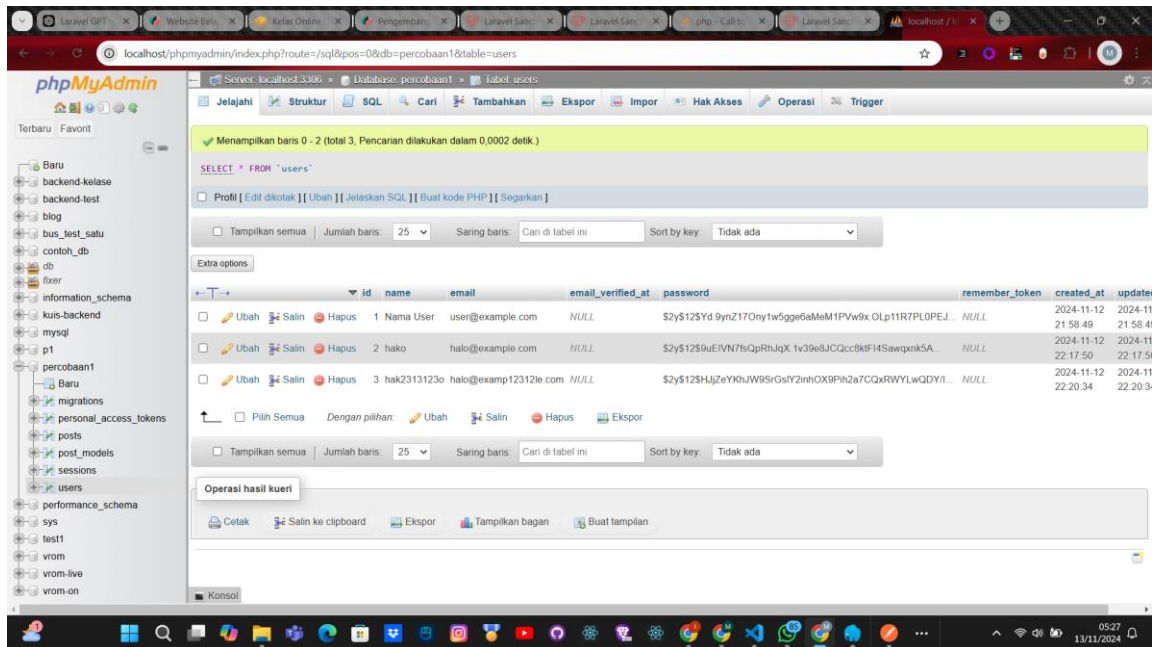


21. Kemudian klik add request

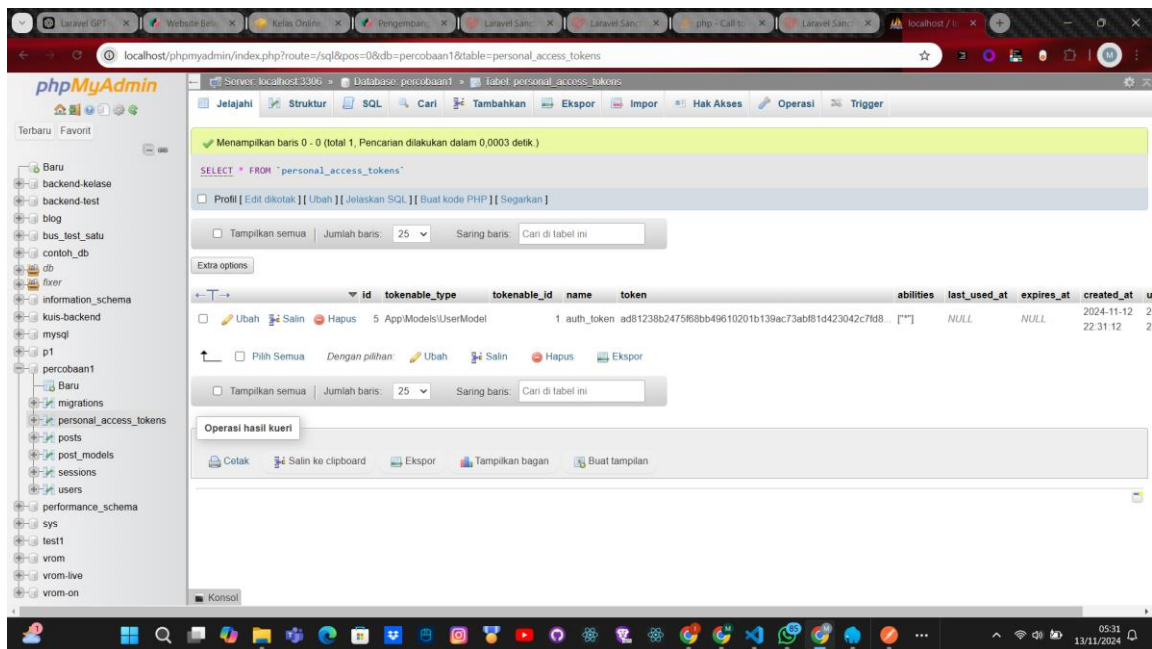
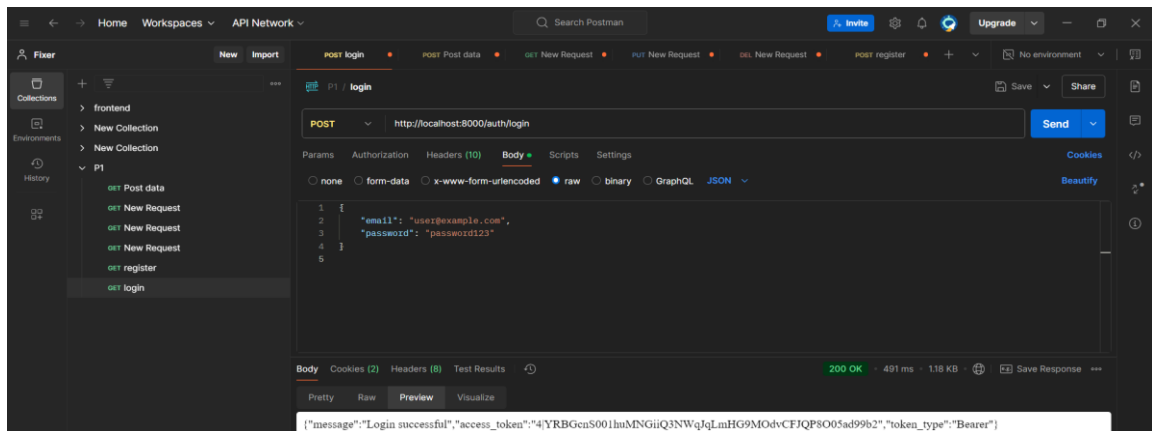


22. Setelah itu gantilah requestnya sesuaikan dengan request yang diinginkan dan routesnya, kemudian klik send:

- URL:** `http://localhost:8000/auth/register`



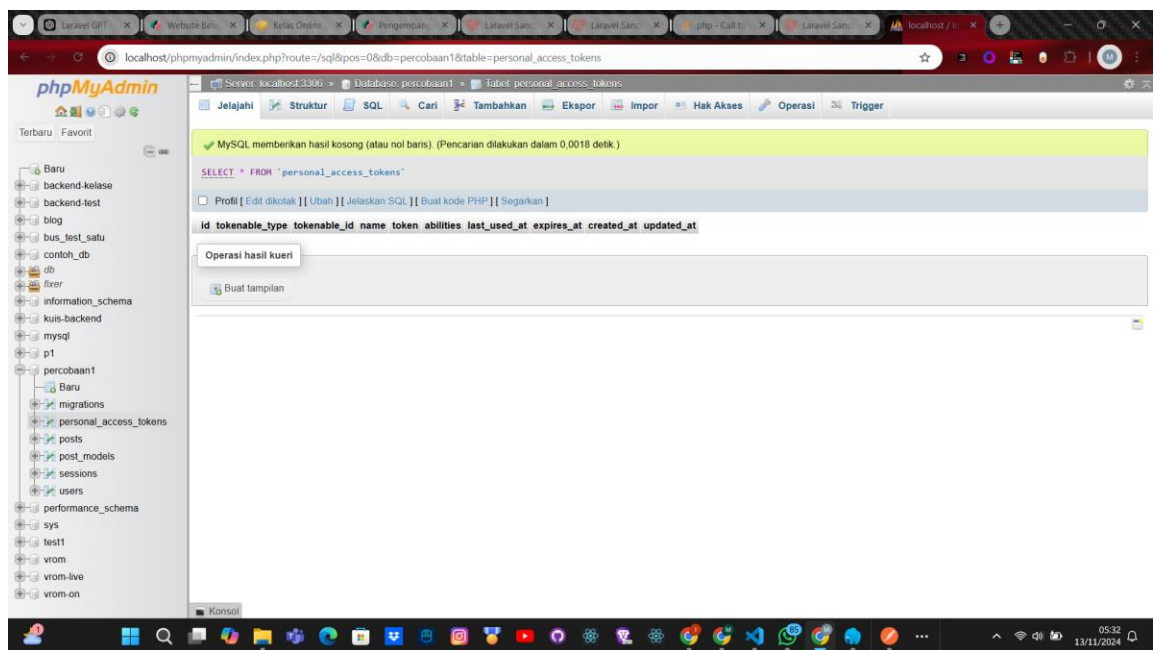
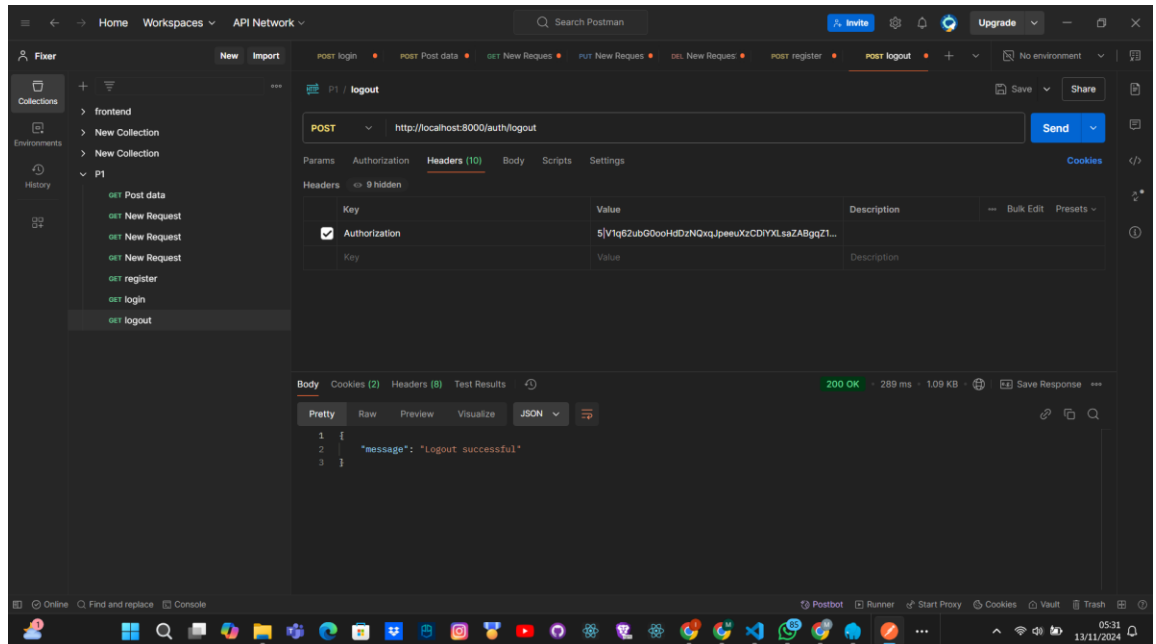
- URL:** <http://localhost:8000/auth/login>



- Function logout

Untuk bagian ini menggunakan header dengan value token berdasarkan akun yang sudah melakukan login

URL: <http://localhost:8000/auth/logout>



## 2. TUGAS

Jawablah Pertanyaan dari Soal ini |

1. Cobalah untuk melakukan semua percobaan diatas
2. Kemudian buatlah versi project kalian sendiri
3. table user sudah harus bisa melakukan proses register, login, dan logout

#### PENGUMPULAN TUGAS DAN DEADLINE

- a. Pengumpulan Tugas di Google Drive
- b. Deadline = 13 November 2024 | 23.00 WIB

## DAFTAR PUSTAKA

<https://id.wikipedia.org/wiki/Autentikasi>

<https://api.aodocs.com/20-authentication/20-access-apis-with-bearer-tokens/>

<https://laravel.com/docs/11.x/sanctum>

<https://laravel.com/docs/11.x/sanctum#introduction>

<https://laravel.com/docs/11.x/authentication>