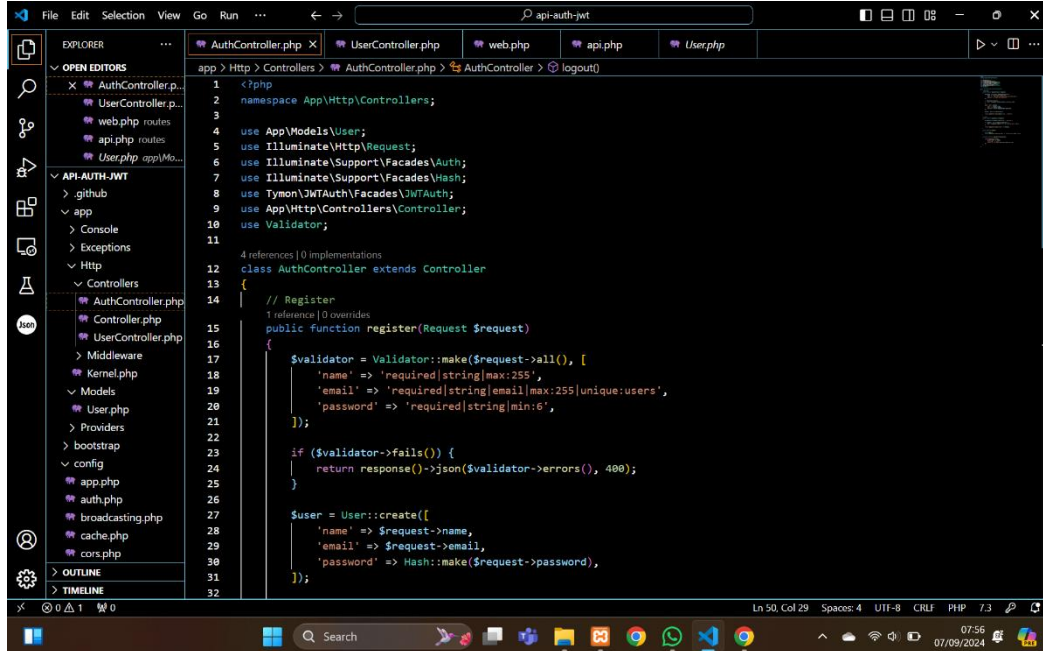


# Dokumentasi API Laravel Menggunakan Autentikasi JWT (Json Web Token)

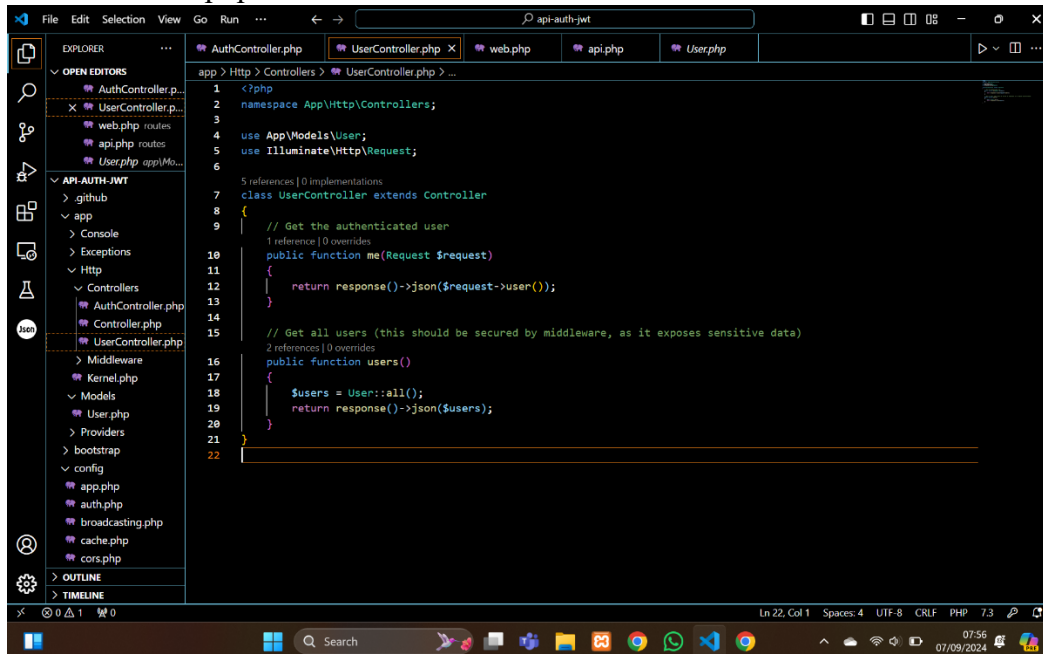
## A. Source Code

### 1. AuthController.php



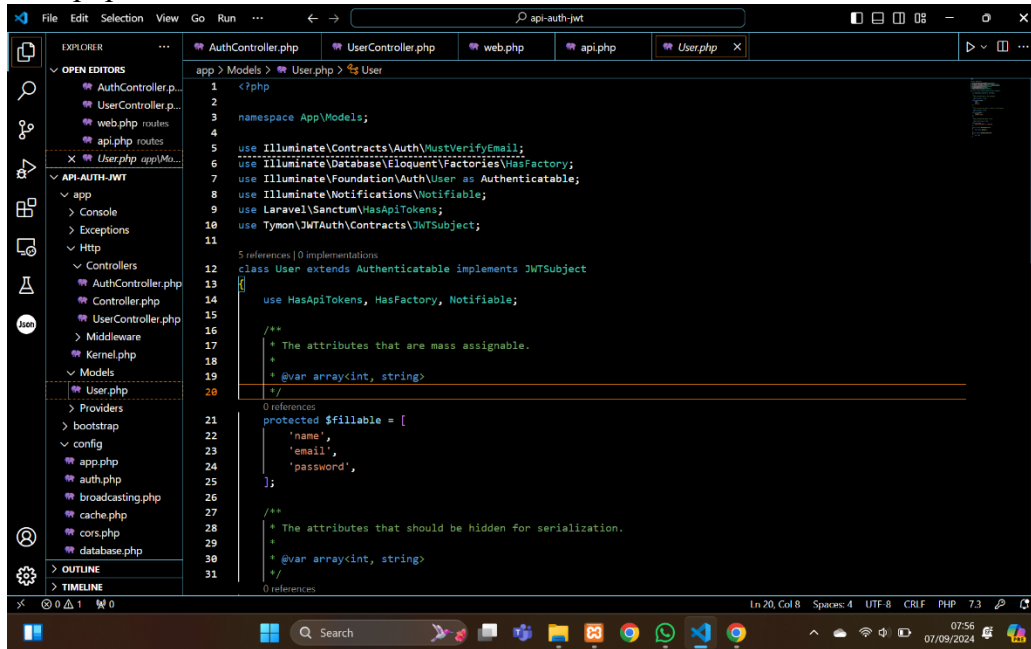
```
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Models\User;
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Auth;
7 use Illuminate\Support\Facades\Hash;
8 use Tymon\JWTAuth\Facades\JWTAuth;
9 use App\Http\Controllers\Controller;
10 use Validator;
11
12 class AuthController extends Controller
13 {
14     // Register
15     public function register(Request $request)
16     {
17         $validator = Validator::make($request->all(), [
18             'name' => 'required|string|max:255',
19             'email' => 'required|string|email|max:255|unique:users',
20             'password' => 'required|string|min:6',
21         ]);
22
23         if ($validator->fails()) {
24             return response()->json($validator->errors(), 400);
25         }
26
27         $user = User::create([
28             'name' => $request->name,
29             'email' => $request->email,
30             'password' => Hash::make($request->password),
31         ]);
32     }
33 }
```

### 2. UserController.php

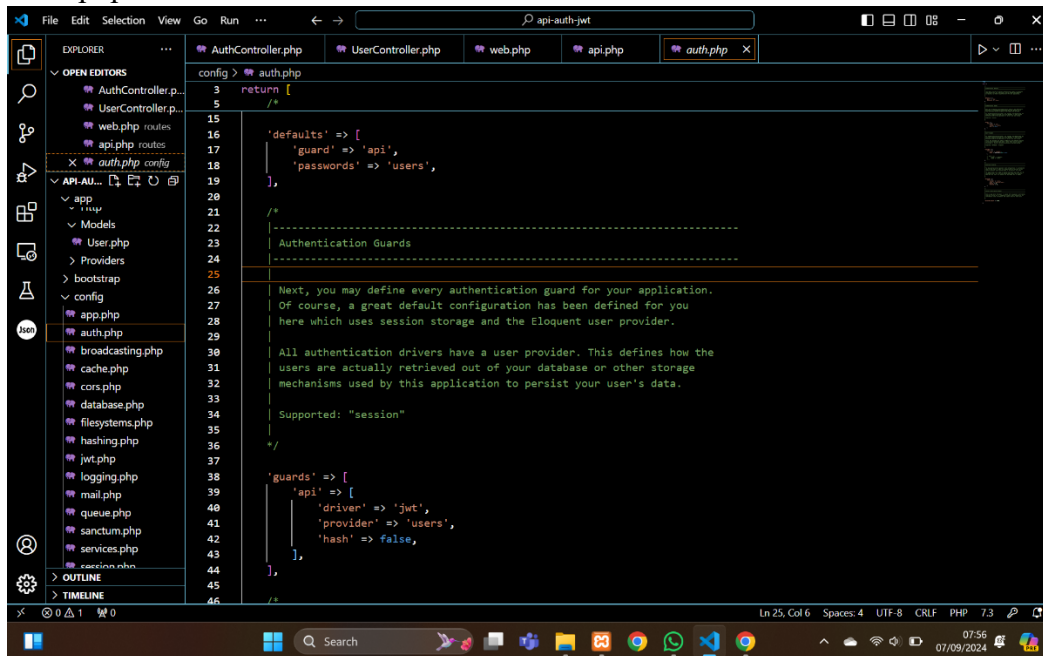


```
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Models\User;
5 use Illuminate\Http\Request;
6
7 class UserController extends Controller
8 {
9     // Get the authenticated user
10    public function me(Request $request)
11    {
12        return response()->json($request->user());
13    }
14
15    // Get all users (this should be secured by middleware, as it exposes sensitive data)
16    public function users()
17    {
18        $users = User::all();
19        return response()->json($users);
20    }
21 }
22
```

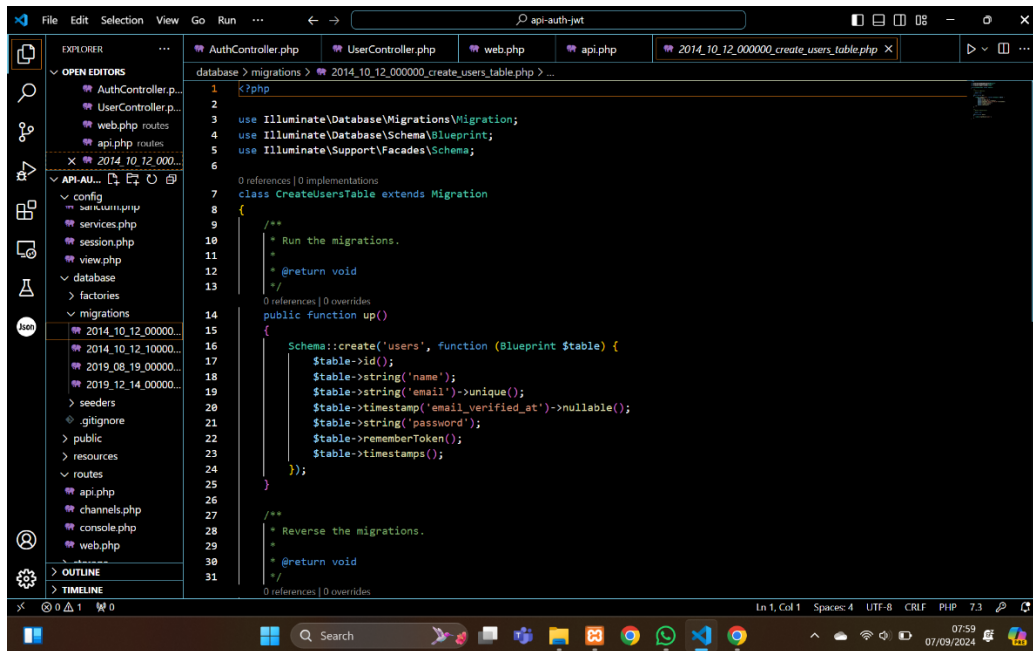
### 3. User.php



#### 4. Auth.php

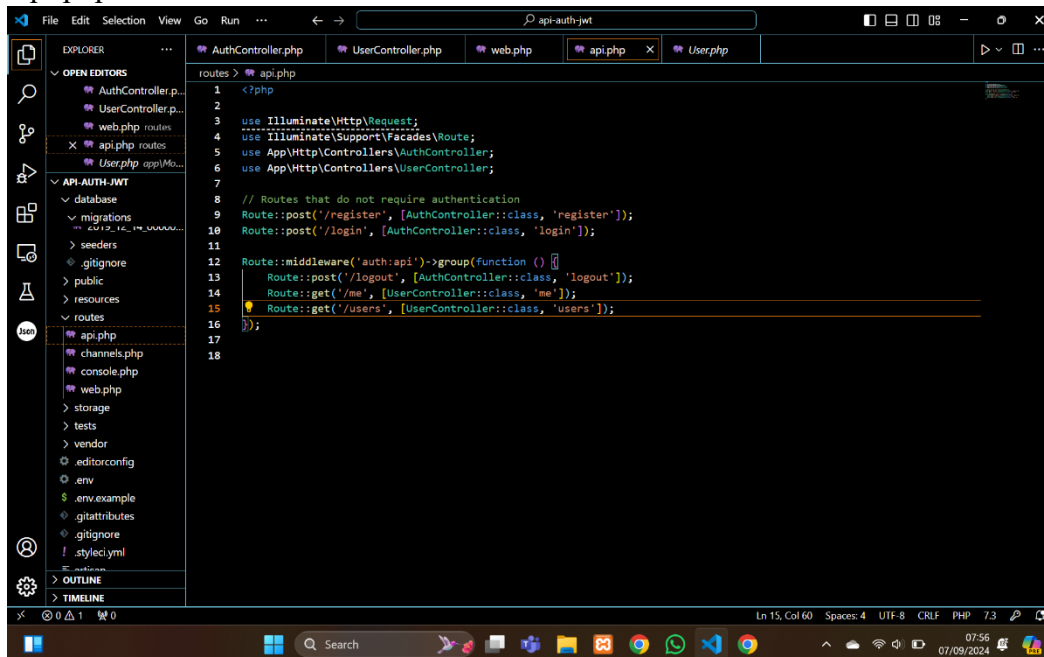


5. 2014\_10\_12\_000000\_create\_users\_table



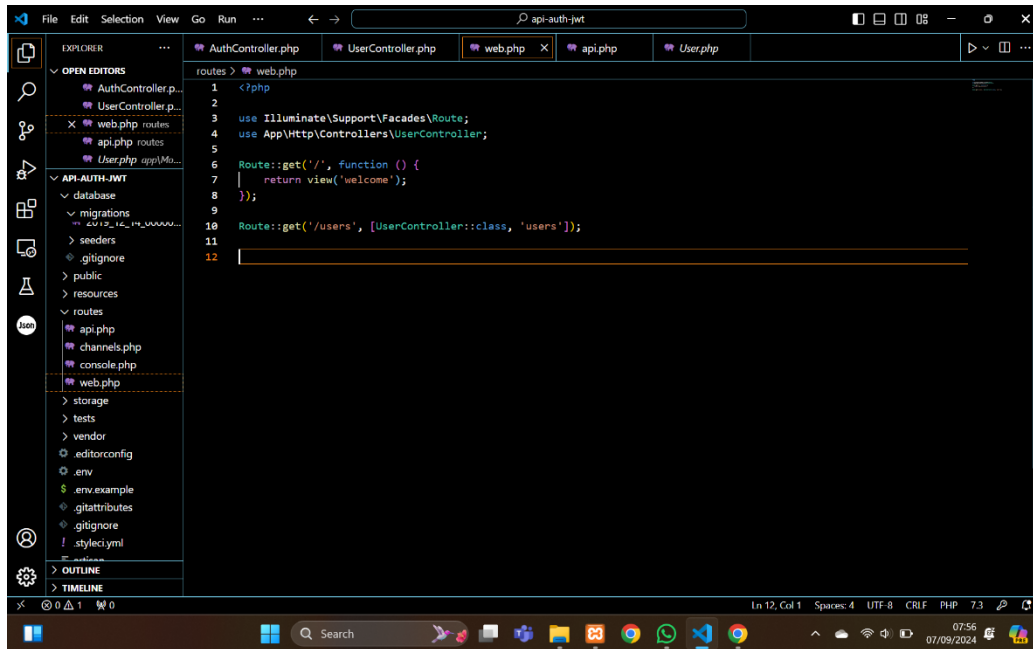
```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class CreateUsersTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->id();
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->timestamp('email_verified_at')->nullable();
21             $table->string('password');
22             $table->rememberToken();
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      *
30      * @return void
31      */
32 }
```

## 6. Api.php



```
1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\AuthController;
6 use App\Http\Controllers\UserController;
7
8 // Routes that do not require authentication
9 Route::post('/register', [AuthController::class, 'register']);
10 Route::post('/login', [AuthController::class, 'login']);
11
12 Route::middleware('auth:api')->group(function () {
13     Route::post('/logout', [AuthController::class, 'logout']);
14     Route::get('/me', [UserController::class, 'me']);
15     Route::get('/users', [UserController::class, 'users']);
16 });
17
18
```

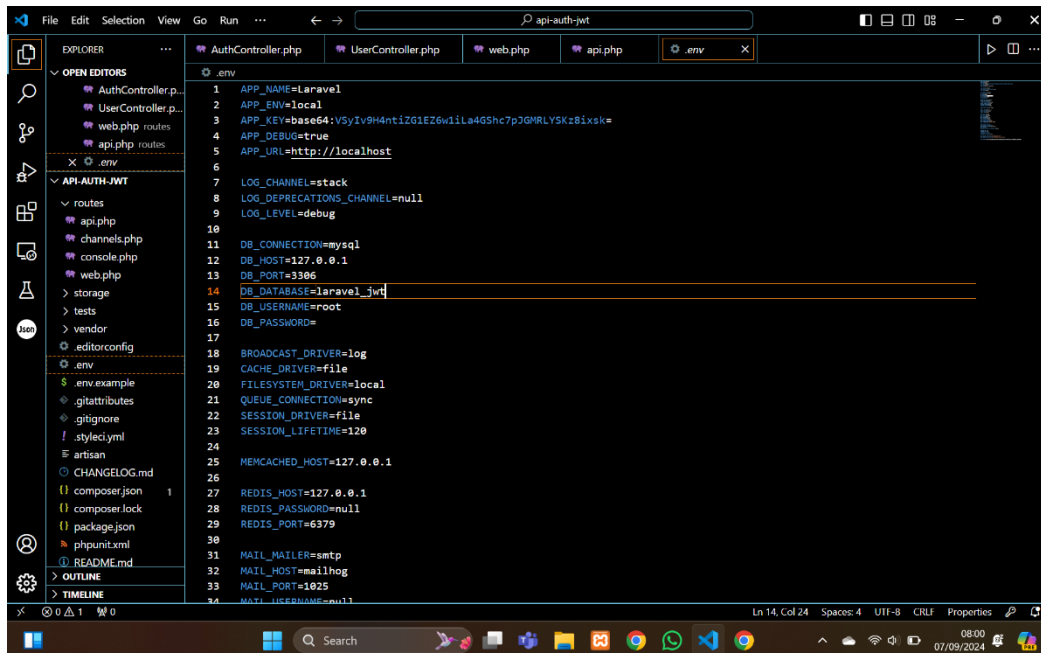
## 7. Web.php



The screenshot shows the Visual Studio Code editor with the 'api-auth-jwt' project open. The Explorer sidebar on the left shows the project structure, including folders like 'database', 'migrations', 'seeds', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', and files like '.env', '.env.example', '.gitattributes', '.gitignore', 'styleci.yml', and 'OUTLINE'. The 'routes' folder is expanded, showing 'api.php', 'channels.php', 'console.php', and 'web.php'. The 'web.php' file is selected and its content is displayed in the main editor area. The code defines two routes: a root route that returns a 'welcome' view, and a '/users' route that uses the 'UserController' class.

```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\UserController;
5
6 Route::get('/', function () {
7     return view('welcome');
8 });
9
10 Route::get('/users', [UserController::class, 'users']);
11
12
```

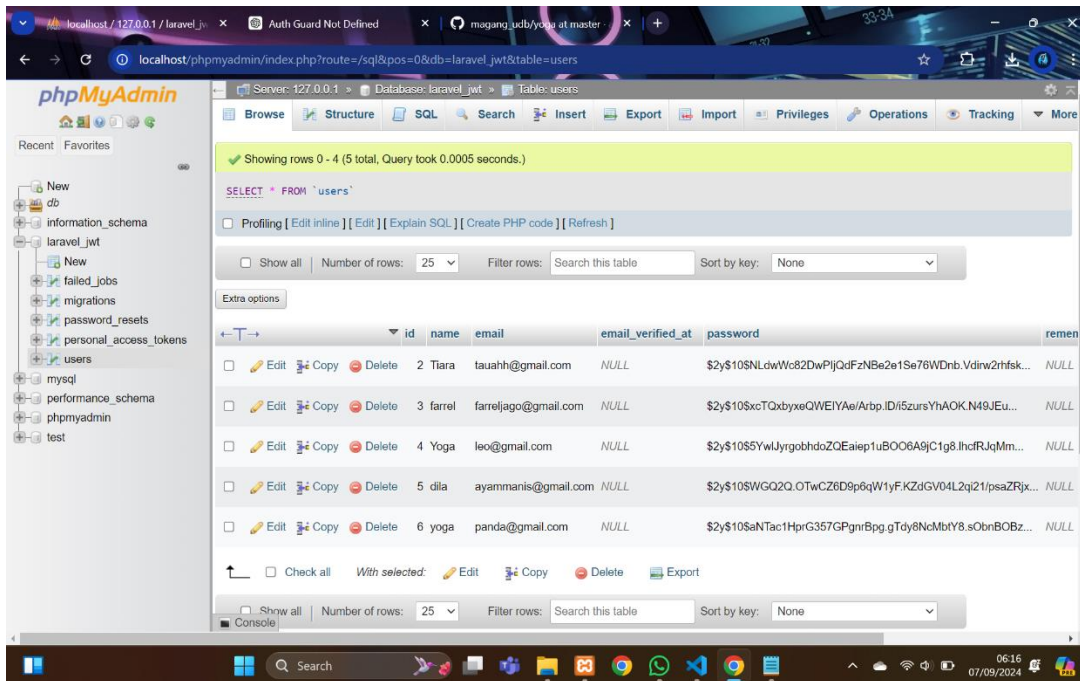
## 8. .env



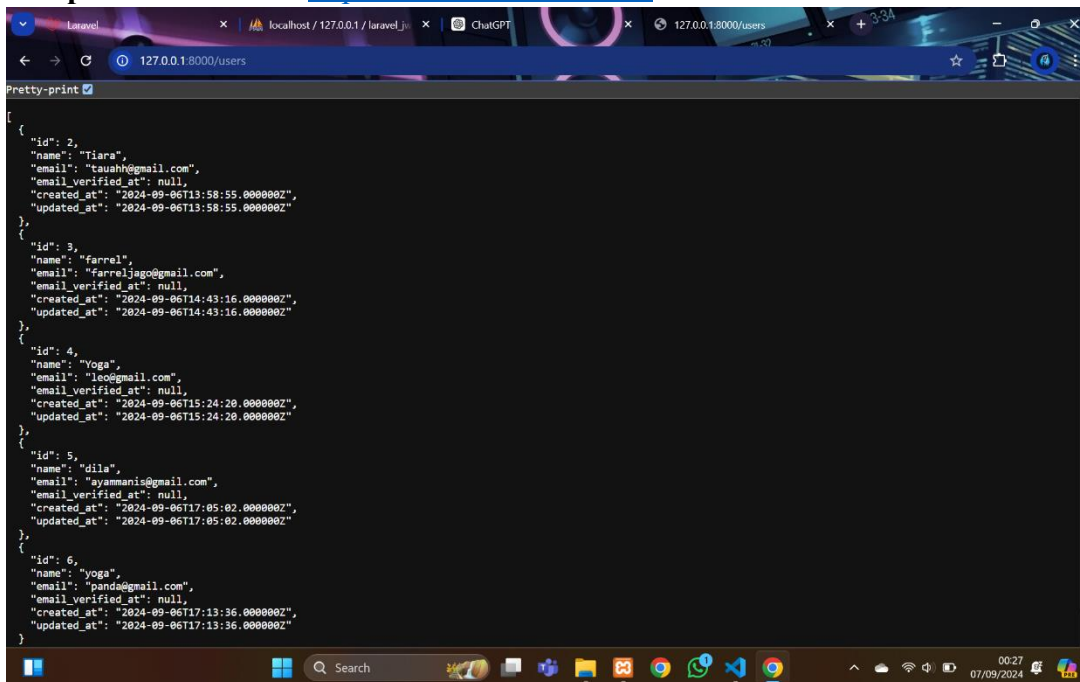
The screenshot shows the Visual Studio Code editor with the 'api-auth-jwt' project open. The Explorer sidebar on the left shows the project structure, including folders like 'database', 'migrations', 'seeds', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', and files like '.env', '.env.example', '.gitattributes', '.gitignore', 'styleci.yml', and 'OUTLINE'. The '.env' file is selected and its content is displayed in the main editor area. The file contains various configuration variables for the application, including database settings, cache settings, and mail settings.

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:V5yIv9H4ntiZG1E26w1le4GShc7pJGMRLYSkz8ixsk=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel_jwt
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DRIVER=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtp
32 MAIL_HOST=mailhog
33 MAIL_PORT=1025
34 MAIL_USERNAME=null
```

## B. Database



## C. Tampilan di localhost <http://127.0.0.1:8000/users>



## D. Pengujian di Postman

1. <http://127.0.0.1:8000/api/register>



