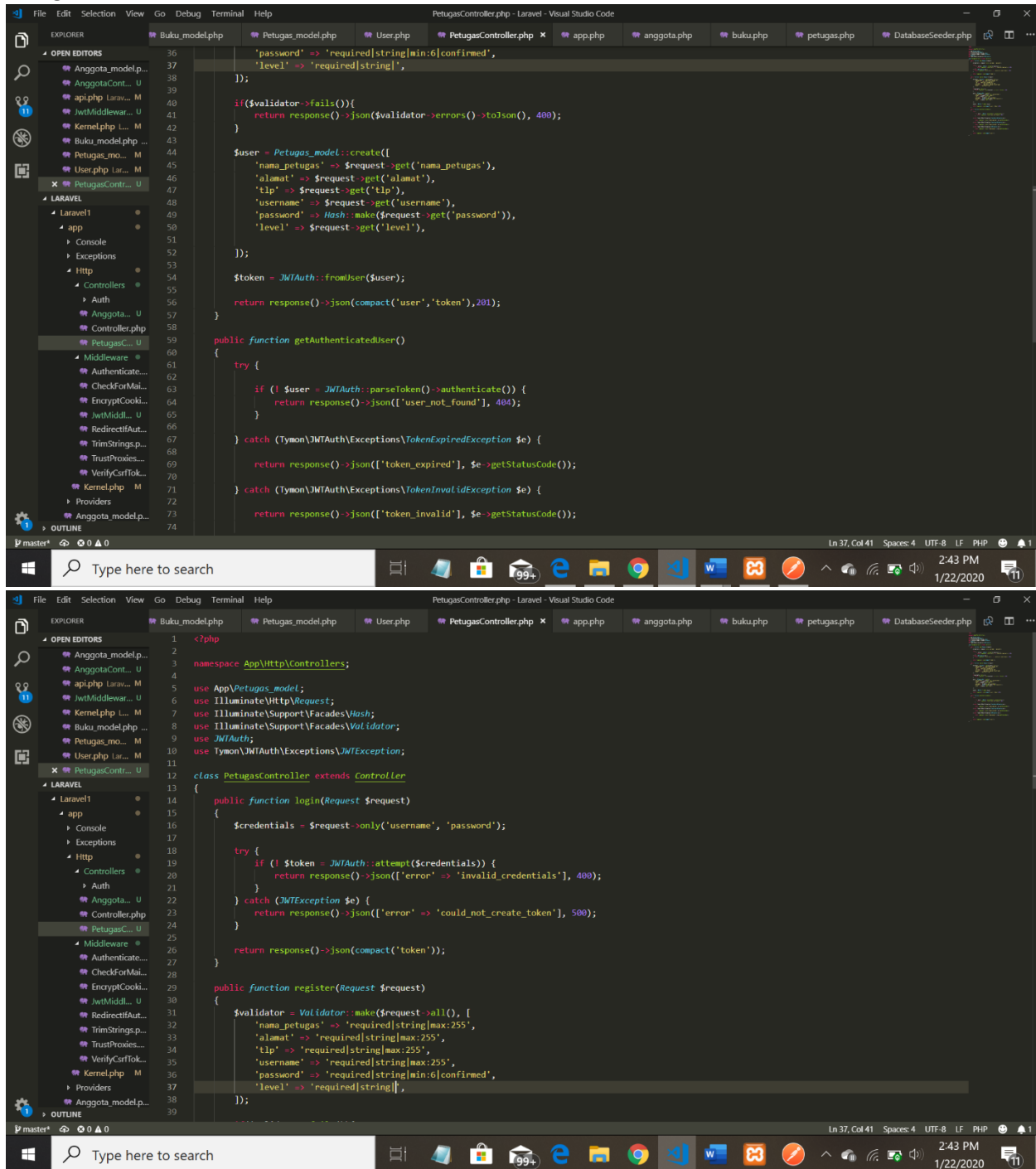


Petugas Controller



```
File Edit Selection View Go Debug Terminal Help
PetugasController.php - Laravel - Visual Studio Code

EXPLORER
Buku_model.php Petugas_model.php User.php PetugasController.php app.php anggota.php buku.php petugas.php DatabaseSeeder.php

OPEN EDITORS
Anggota_model.p... 37
AnggotaCont... U 38
api.php Larav... M 39
JwtMiddlewar... U 40
Kernel.php L... M 41
Buku_model.php ... 42
Petugas_mo... M 43
User.php Lar... M 44
PetugasCont... U 45
Laravel 46
  app 47
  Console 48
  Exceptions 49
  Http 50
    Controllers 51
      Auth 52
      Anggota... U 53
      Controller.php 54
      PetugasC... U 55
      Middleware 56
        Authenticate... 57
        CheckForMai... 58
        EncryptCooki... 59
        JwtMiddle... U 60
        RedirectAut... 61
        TrimStrings.p... 62
        TrustProxies... 63
        VerifyCsrfTok... 64
        Kernel.php M 65
        Providers 66
        Anggota_model.p... 67
        OUTLINE 68

PetugasController.php
36 'password' => 'required|string|min:6|confirmed',
37 'level' => 'required|string',
38 ];
39
40 if($validator->fails()){
41     return response()->json($validator->errors()->toJson(), 400);
42 }
43
44 $user = Petugas_model::create([
45     'nama_petugas' => $request->get('nama_petugas'),
46     'alamat' => $request->get('alamat'),
47     'telp' => $request->get('telp'),
48     'username' => $request->get('username'),
49     'password' => Hash::make($request->get('password')),
50     'level' => $request->get('level'),
51 ]);
52
53 $token = JWTAuth::fromUser($user);
54
55 return response()->json(compact('user','token'),201);
56
57 public function getAuthenticatedUser()
58 {
59     try {
60         if (! $user = JWTAuth::parseToken()->authenticate()) {
61             return response()->json(['user_not_found'], 404);
62         }
63     } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
64         return response()->json(['token_expired'], $e->getStatusCode());
65     } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
66         return response()->json(['token_invalid'], $e->getStatusCode());
67     }
68 }

Ln 37, Col 41 Spaces: 4 UTF-8 LF PHP 1

Type here to search

File Edit Selection View Go Debug Terminal Help
PetugasController.php - Laravel - Visual Studio Code

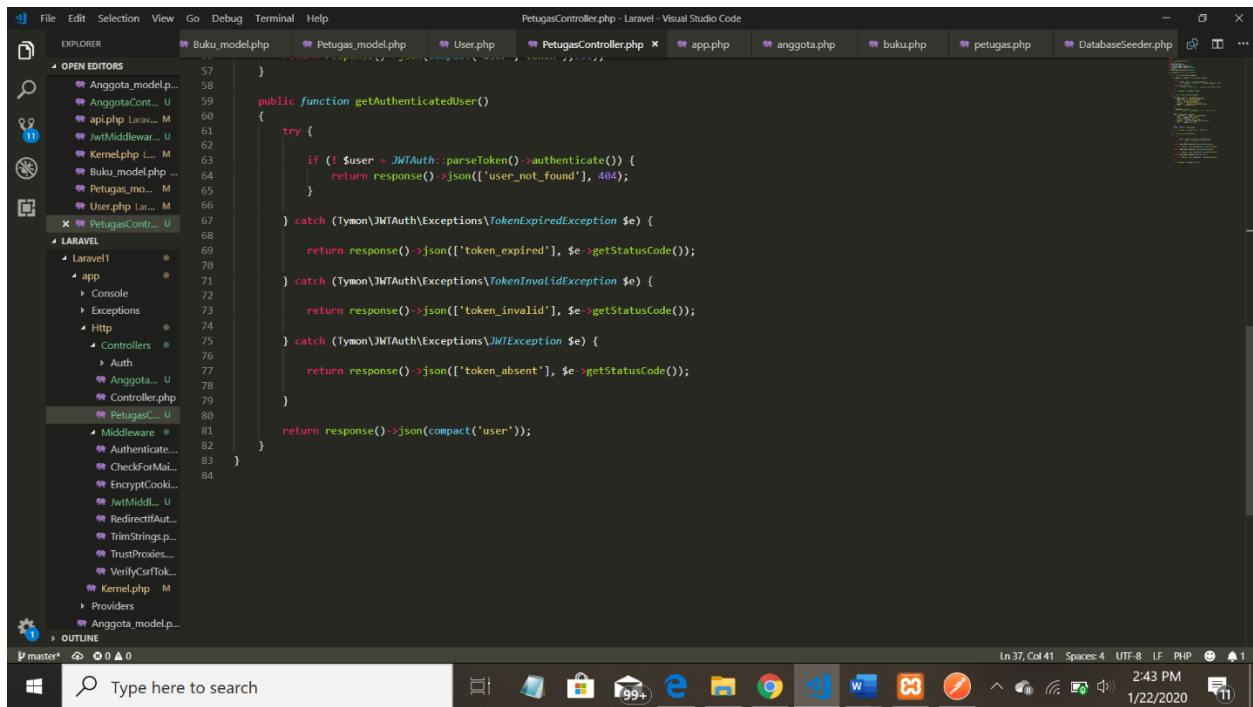
EXPLORER
Buku_model.php Petugas_model.php User.php PetugasController.php app.php anggota.php buku.php petugas.php DatabaseSeeder.php

OPEN EDITORS
Anggota_model.p... 1
AnggotaCont... U 2
api.php Larav... M 3
JwtMiddlewar... U 4
Kernel.php L... M 5
Buku_model.php ... 6
Petugas_mo... M 7
User.php Lar... M 8
PetugasCont... U 9
Laravel 10
  app 11
  Console 12
  Exceptions 13
  Http 14
    Controllers 15
      Auth 16
      Anggota... U 17
      Controller.php 18
      PetugasC... U 19
      Middleware 20
        Authenticate... 21
        CheckForMai... 22
        EncryptCooki... 23
        JwtMiddle... U 24
        RedirectAut... 25
        TrimStrings.p... 26
        TrustProxies... 27
        VerifyCsrfTok... 28
        Kernel.php M 29
        Providers 30
        Anggota_model.p... 31
        OUTLINE 32

PetugasController.php
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Petugas_model;
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Hash;
7 use Illuminate\Support\Facades\Validator;
8 use JWTAuth;
9 use Tymon\JWTAuth\Exceptions\JWTException;
10
11 class PetugasController extends Controller
12 {
13     public function login(Request $request)
14     {
15         $credentials = $request->only('username', 'password');
16
17         try {
18             if (! $token = JWTAuth::attempt($credentials)) {
19                 return response()->json(['error' => 'invalid_credentials'], 400);
20             }
21         } catch (JWTException $e) {
22             return response()->json(['error' => 'could_not_create_token'], 500);
23         }
24
25         return response()->json(compact('token'));
26     }
27
28     public function register(Request $request)
29     {
30         $validator = Validator::make($request->all(), [
31             'nama_petugas' => 'required|string|max:255',
32             'alamat' => 'required|string|max:255',
33             'telp' => 'required|string|max:255',
34             'username' => 'required|string|max:255',
35             'password' => 'required|string|min:6|confirmed',
36             'level' => 'required|string',
37         ]);
38
39     }

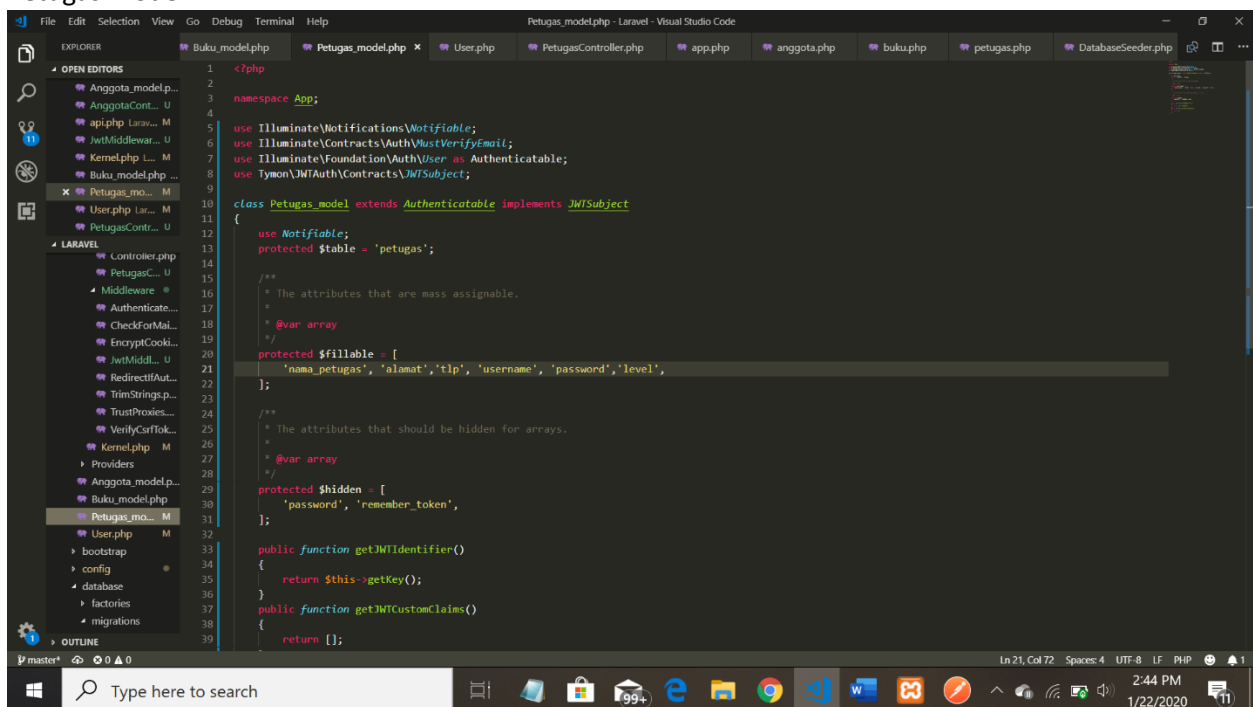
Ln 37, Col 41 Spaces: 4 UTF-8 LF PHP 1

Type here to search
```



```
57 }
58
59 public function getAuthenticatedUser()
60 {
61     try {
62         if (! $user = JWTAuth::parseToken()->authenticate()) {
63             return response()->json(['user_not_found'], 404);
64         }
65     } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
66         return response()->json(['token_expired'], $e->getStatusCode());
67     } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
68         return response()->json(['token_invalid'], $e->getStatusCode());
69     } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
70         return response()->json(['token_absent'], $e->getStatusCode());
71     }
72     return response()->json(compact('user'));
73 }
74
75 }
```

Petugas Model



```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Notifications\Notifiable;
6 use Illuminate\Contracts\Auth\MustVerifyEmail;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Tymon\JWTAuth\Contracts\JWTSubject;
9
10 class Petugas_model extends Authenticatable implements JWTSubject
11 {
12     use Notifiable;
13     protected $table = 'petugas';
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array
19      */
20     protected $fillable = [
21         'nama_petugas', 'alamat', 'tipe', 'username', 'password', 'level',
22     ];
23
24     /**
25      * The attributes that should be hidden for arrays.
26      *
27      * @var array
28      */
29     protected $hidden = [
30         'password', 'remember_token',
31     ];
32
33     public function getJWTIdentifier()
34     {
35         return $this->getKey();
36     }
37     public function getJWTCustomClaims()
38     {
39         return [];
40     }
41 }
```

```

1  <?php
2
3  use Illuminate\Http\Request;
4
5  /*
6   * API Routes
7   *
8   * Here is where you can register API routes for your application. These
9   * routes are loaded by the RouteServiceProvider within a group which
10  * is assigned the "api" middleware group. Enjoy building your API!
11  */
12
13  Route::middleware('auth:api')->get('/user', function (Request $request) {
14      return $request->user();
15  });
16
17  Route::post('register', 'PetugasController@register');
18  Route::post('login', 'PetugasController@login');
19
20  Route::get('/', function(){
21      return Auth::user() ?>alamat;
22  })->middleware('jwt.verify');
23

```

The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with 'New', 'Import', 'Runner', and a dropdown. The main workspace area has 'My Workspace' and 'Invite' buttons. The left sidebar shows a list of requests under 'History' and 'Collections'. The right sidebar shows 'No Environment' and 'Send'/'Save' buttons. The main panel shows a 'POST' request to 'http://127.0.0.1:8000/api/register' with a JSON body containing user details and a token.

Login

The screenshot shows the Postman application window. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and workspace controls (My Workspace, Invite). The left sidebar shows a list of requests under 'Today', with the first one selected. The main panel displays a POST request to `http://127.0.0.1:8000/api/login`. The response is a JSON object with a 'token' field, displayed in the 'Pretty' view. The status is 200 OK, time is 553ms, and size is 608 B.

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

Body Cookies Headers (9) Test Results Status: 200 OK Time: 553ms Size: 608 B Save Response

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "token":
3     "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOlwvXC8xMjcucMC4wLjE6ODAwMFwvYX8pXC9sb2dpciIsImh
dCI6MTU3OTY3ODg4NCwiZXhwIjozNTc5NjgyNDg0LCJyYmYiOiJlNzk2Nzg4ODQsImp0aSI6InpjbWNLV0dLM3JlNm5PSUUiLCJzdw
IiOjksInBydiI6Ijg3ZTBhZjFlZjlmZDE0DEYzMRIYzk3MTUzYTE0ZTBjMDQ3NTQ2YWEifQ.
KnRSKw8pZOWsuZF1qKufpU2OQ4a-rRxhdY1iAw8D2Q8"
```

Get Alamat

The screenshot shows the Postman application window. The top bar is the same as the previous image. The left sidebar shows a list of requests under 'Today', with the first one selected. The main panel displays a GET request to `http://127.0.0.1:8000/api`. The response is a JSON object with a 'Malang' field, displayed in the 'Pretty' view. The status is 200 OK, time is 651ms, and size is 291 B.

GET `http://127.0.0.1:8000/api` Send Save

☒ Content-Type `application/x-www-form-urlencoded`

Key Value Description

Temporary Headers (8)

Body Cookies Headers (9) Test Results Status: 200 OK Time: 651ms Size: 291 B Save Response

Pretty Raw Preview Visualize BETA HTML

```
1 Malang
```