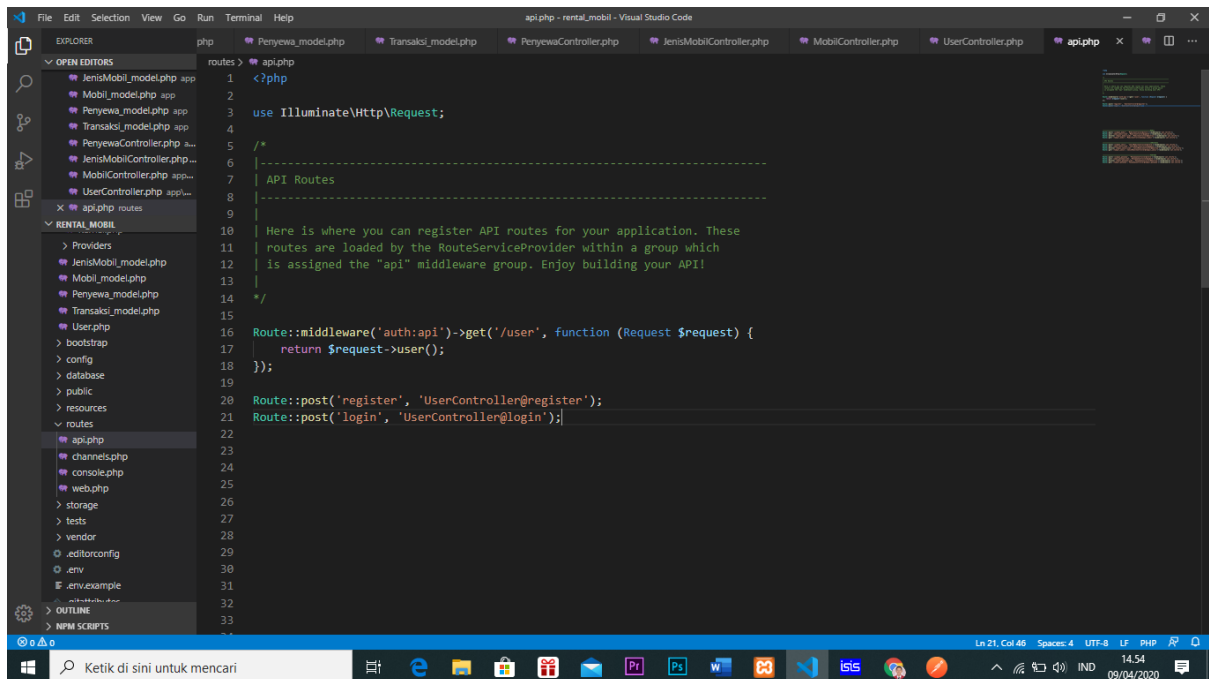


Abelino Athallah W

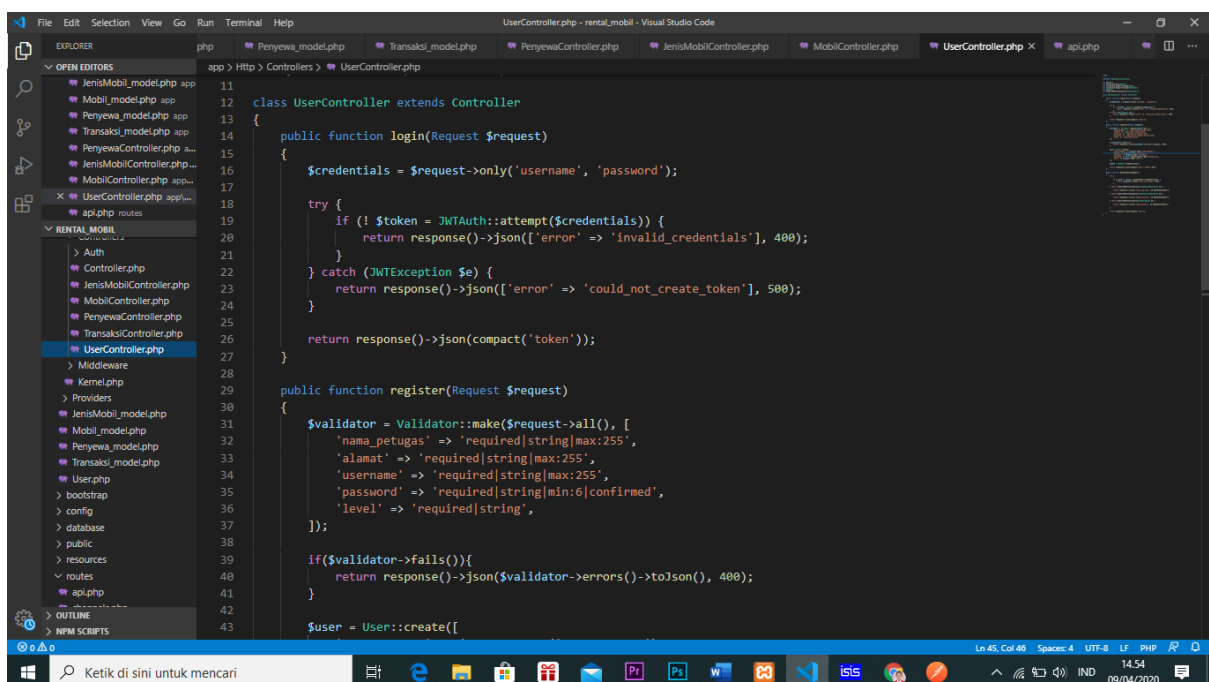
XIRPL7 / 01

API



```
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', 'UserController@register');
18 Route::post('login', 'UserController@login');
```

CONTROLLER



```
11 class UserController 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             'username' => 'required|string|max:255',
34             'password' => 'required|string|min:6|confirmed',
35             'level' => 'required|string',
36         ]);
37
38         if ($validator->fails()) {
39             return response()->json($validator->errors()->toJson(), 400);
40         }
41
42         $user = User::create([
43             'nama_petugas' => $request->nama_petugas,
44             'alamat' => $request->alamat,
45             'username' => $request->username,
46             'password' => $request->password,
47             'level' => $request->level,
48         ]);
49
50         return response()->json($user->toArray(), 201);
51     }
52 }
```

The screenshot shows the Visual Studio Code editor with the file `UserController.php` open. The `register` method is implemented as follows:

```
public function register(Request $request)
{
    $validator = Validator::make($request->all(), [
        'nama_petugas' => 'required|string|max:255',
        'alamat' => 'required|string|max:255',
        'username' => 'required|string|max:255',
        'password' => 'required|string|min:6|confirmed',
        'level' => 'required|string',
    ]);

    if($validator->fails()){
        return response()->json($validator->errors()->toJson(), 400);
    }

    $user = User::create([
        'nama_petugas' => $request->get('nama_petugas'),
        'alamat' => $request->get('alamat'),
        'username' => $request->get('username'),
        'password' => Hash::make($request->get('password')),
        'level' => $request->get('level'),
    ]);

    $token = JWTAuth::fromUser($user);

    return response()->json(compact('user','token'),201);
}

public function getAuthenticatedUser()
{
    try {
        if (! $user = JWTAuth::parseToken()->authenticate()) {
            return response()->json(['user not found'], 404);
        }
    }
}
```

MODEL

The screenshot shows the Visual Studio Code editor with the file `User.php` open. The `User` model is implemented as follows:

```
class User extends Authenticatable implements JWTSubject
{
    protected $table="petugas";
    public $timestamps=false;
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'nama_petugas', 'alamat', 'username', 'password', 'level',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function getJWTIdentifier()
    {
        return $this->getKey();
    }

    public function getJWTCustomClaims()
    {
        return [];
    }
}
```