

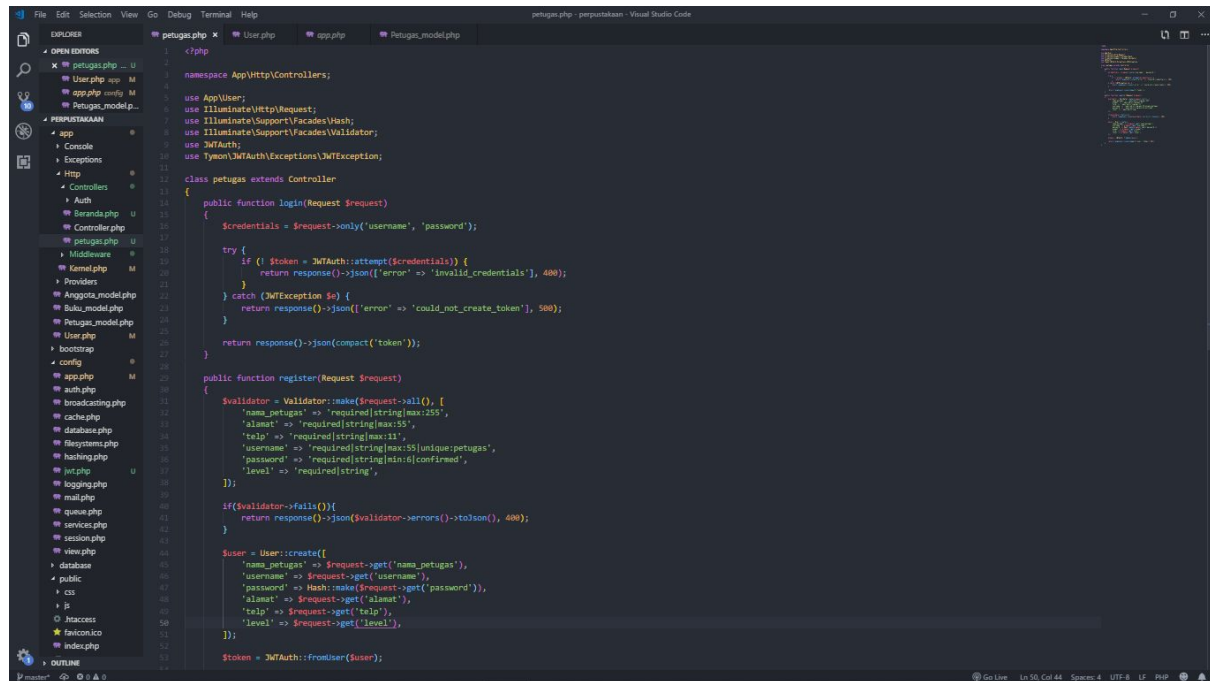
Nama : Muhammad Bagas Ramadhan

Absen :19

make controller

```
C:\xampp\htdocs\perpustakaan>php artisan make:controller Petugas
```

Petugas controller



```
<?php
namespace App\Http\Controllers;

use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;

class Petugas extends Controller
{
    public function login(Request $request)
    {
        $credentials = $request->only('username', 'password');

        try {
            if (! $token = JWTAuth::attempt($credentials)) {
                return response()->json(['error' => 'invalid_credentials', 400]);
            }
        } catch (JWTException $e) {
            return response()->json(['error' => 'could_not_create_token', 500]);
        }

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

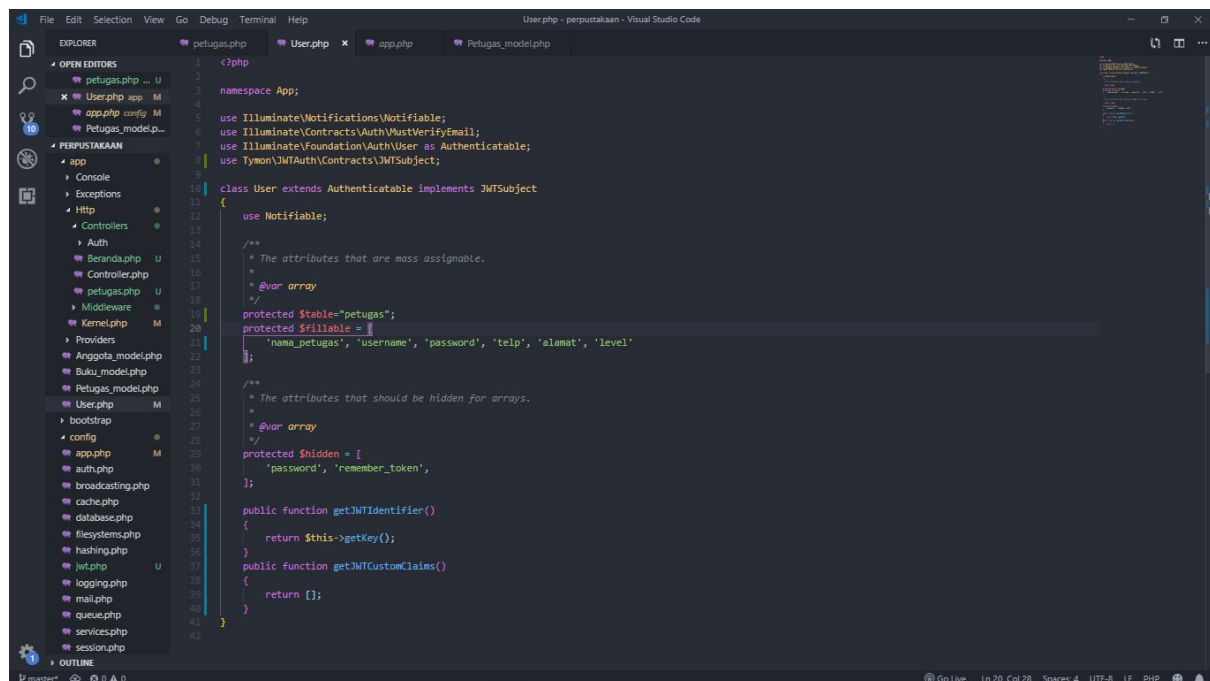
    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'nama_petugas' => 'required|string|max:255',
            'alamat' => 'required|string|max:55',
            'telp' => 'required|string|max:11',
            'username' => 'required|string|max:55|unique:petugas',
            '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'),
            'username' => $request->get('username'),
            'password' => Hash::make($request->get('password')),
            'alamat' => $request->get('alamat'),
            'telp' => $request->get('telp'),
            'level' => $request->get('level'),
        ]);

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

Model



```
<?php
namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements JWTSubject
{
    use Notifiable;

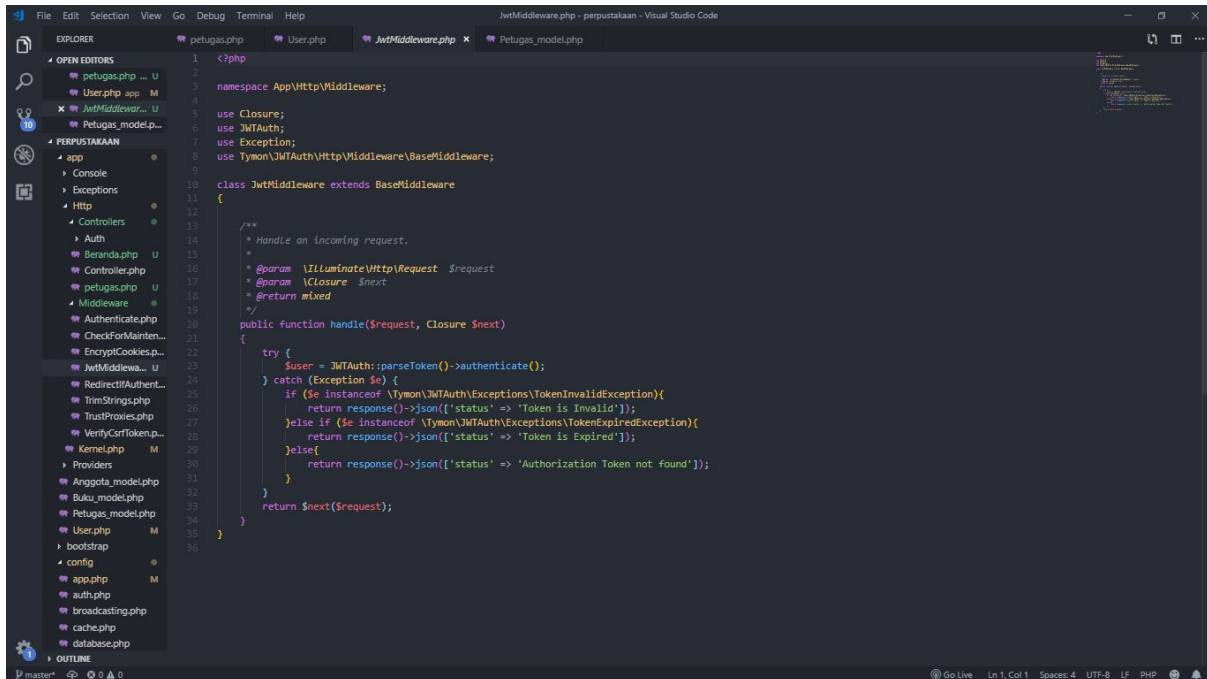
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'nama_petugas', 'username', 'password', 'telp', 'alamat', 'level'
    ];

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

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

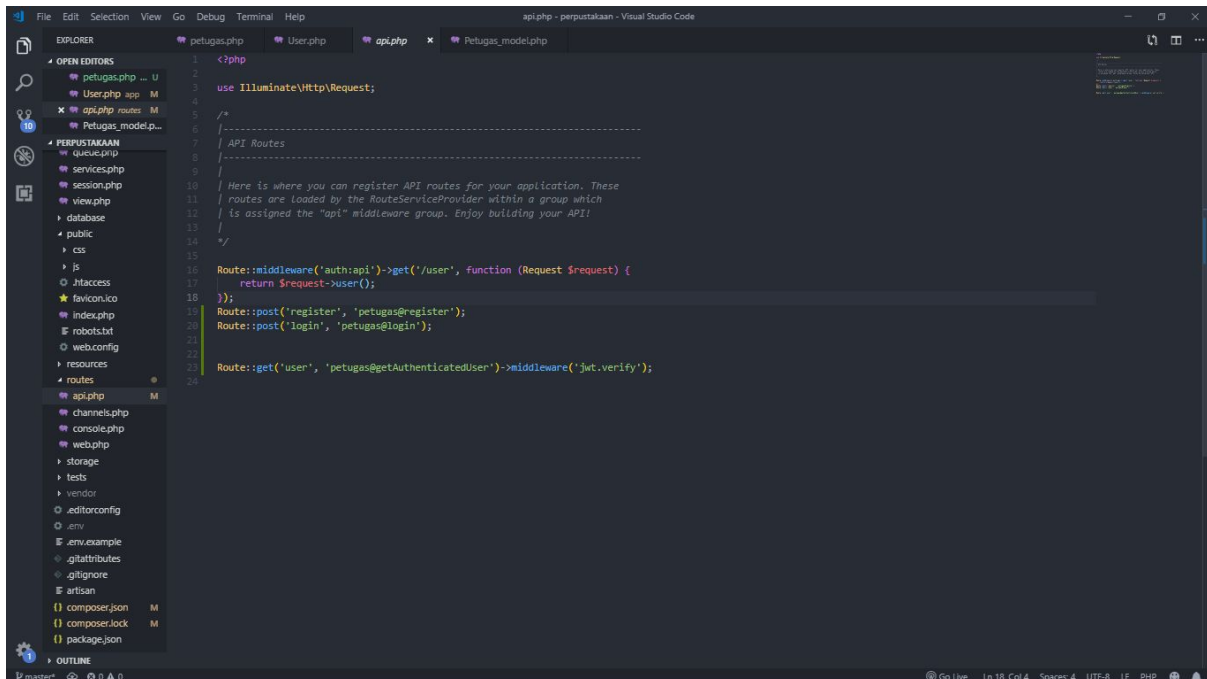
    public function getJWTCustomClaims()
    {
        return [];
    }
}
```

middlewarere



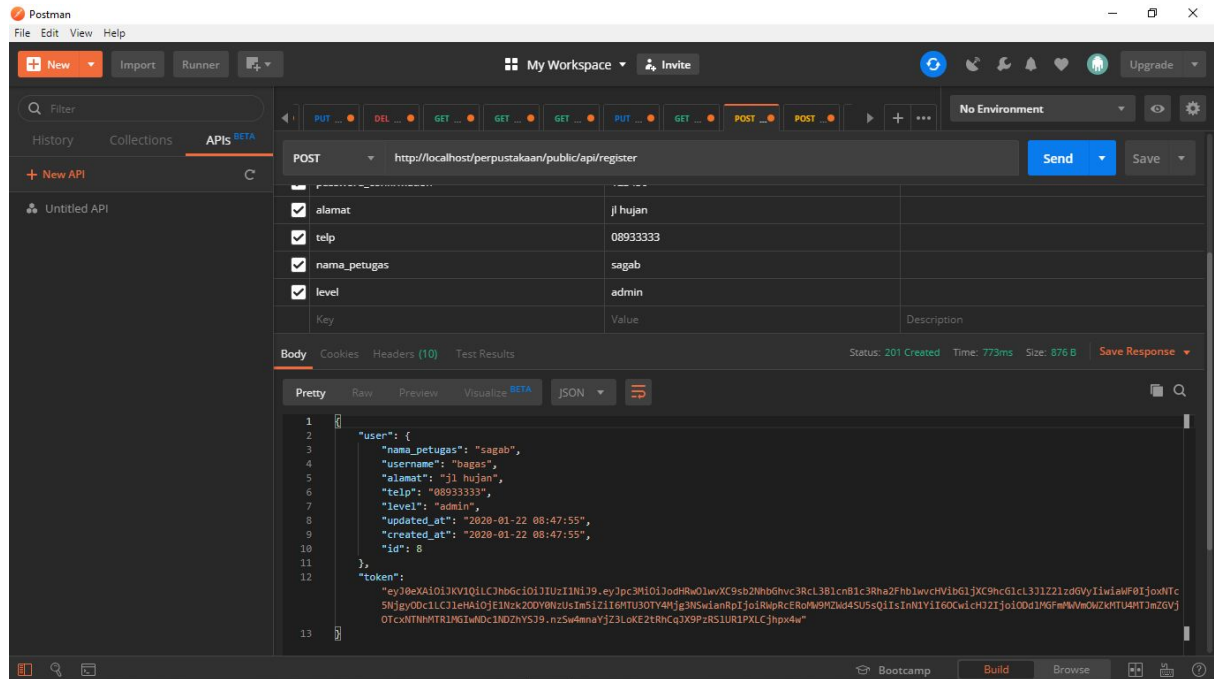
```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use JWTAuth;
7 use Exception;
8 use Tymon\JWTAuth\Http\Middleware\BaseMiddleware;
9
10 class JwMiddleware extends BaseMiddleware
11 {
12
13     /**
14      * Handle an incoming request.
15      *
16      * @param \Illuminate\Http\Request $request
17      * @param \Closure $next
18      * @return mixed
19      */
20     public function handle($request, Closure $next)
21     {
22         try {
23             $user = JWTAuth::parseToken()->authenticate();
24         } catch (Exception $e) {
25             if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenInvalidException){
26                 return response()->json(['status' => 'Token is Invalid']);
27             }else if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenExpiredException){
28                 return response()->json(['status' => 'Token is Expired']);
29             }else{
30                 return response()->json(['status' => 'Authorization Token not found']);
31             }
32         }
33         return $next($request);
34     }
35 }
36
```

api.php



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

Hasil register



Hasil Login

