

Login dan Register

1. Api.php

```
19
20 Route::post('register', 'PetugasController@register');
21 Route::post('login', 'PetugasController@login');
22 Route::get('/', function(){
23     return Auth::user()->level;
24 }->middleware('jwt.verify');
25
```

2. Script User.php

```
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 User extends Authenticatable implements JWTSubject
11 {
12     use Notifiable;
13     protected $table = 'petugas';
14     /**
15      * The attributes that are mass assignable.
16      *
17      * @var array
18      */
19     protected $fillable = [
20         'nama_petugas', 'alamat', 'telp', 'username', 'password', 'level'
21     ];
22
23     /**
24      * The attributes that should be hidden for arrays.
25      *
26      * @var array
27      */
28     protected $hidden = [
29         'password', 'remember_token',
30     ];
31
32     public function getJWTIdentifier()
33     {
34         return $this->getKey();
35     }
36     public function getJWTCustomClaims()
37     {
38         return [];
39     }
40 }
```

3. Script PetugasController.php

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

```

40         if($validator->fails()){
41             return response()->json($validator->errors()->toJson(), 400);
42         }
43
44         $user = User::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
58     public function getAuthenticatedUser()
59     {
60         try {
61
62             if (! $user = JWTAuth::parseToken()->authenticate()) {
63                 return response()->json(['user_not_found'], 404);
64             }
65
66         } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
67
68             return response()->json(['token_expired'], $e->getStatusCode());
69
70         } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
71
72             return response()->json(['token_invalid'], $e->getStatusCode());
73
74         } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
75
76             return response()->json(['token_absent'], $e->getStatusCode());
77
78         }
79
80         return response()->json(compact('user'));
81     }
82 }

```

4. Postman register

[illegible]

5. Postman Login

POST http://localhost/perpustakaan/login

http://localhost/perpustakaan/public/api/login

POST http://localhost/perpustakaan/public/api/login

Send Save

Params Authorization Headers (1) **Body** Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	username	pjm			
<input checked="" type="checkbox"/>	password	123456			
	Key	Value	Description		

Body Cookies Headers (11) Test Results Status: 200 OK Time: 6923 ms Size: 708 B Download

Pretty Raw Preview JSON

```

1 {
2   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
    .eyJpc3MiOiJodHRwOiwwXC9sb2NnbGhvY3RlL3BlcnB1c3RhbnQyZmVudHViOG1jXC9hcGlCL2xvZ2luIiwiaWF0IjoxNTc1NDc3LClleHAiOjE1Nzk2ODkwNzcsIm5iZiI6Mn
    TU3OTY4NTQ3bm9ianRpIjoir256cDdhb1RjT1N1Mmh3VSIsInN1YiI6MTESinBydiI6Ijg3ZTBhZjZlZjlmZDE0OEYzMRIyZk3MTUzYTE0ZTB1MDQ3NTQ2YWEIFQ.XFDc8QYVw
    -fDhvmKSe48nC70V7kupTK0ofo81a756Y"
```

6. Postman level

