Login dan Register

1. Api.php

```php

Route::post('register', 'PetugasController@register');
Route::post('login', 'PetugasController@login');
Route::get('/', function(){
    return Auth::user()->level;
})->middleware('jwt.verify');

```

2. Script User.php

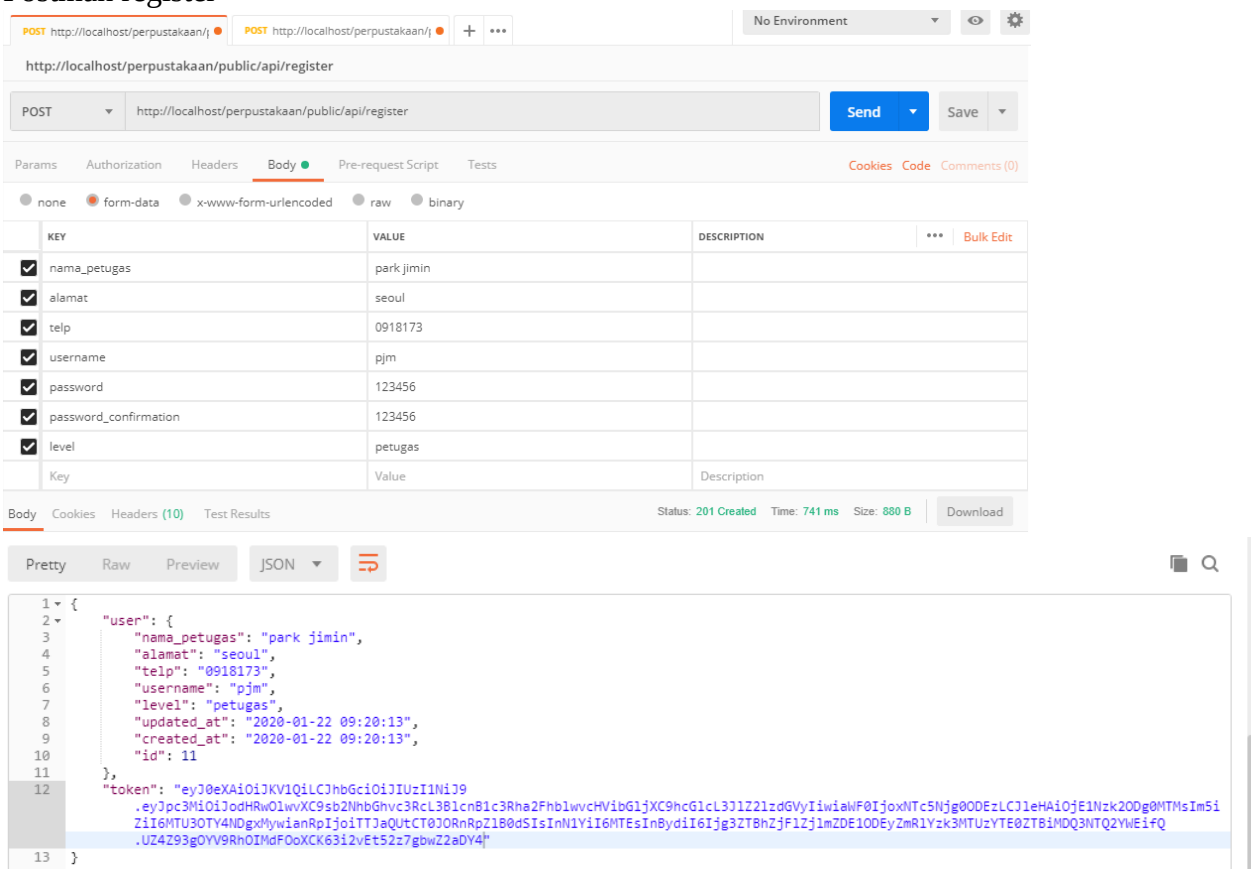
```php
<?php

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;
    protected $table = 'petugas';
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'nama_petugas','alamat', 'telp', '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 [];
    }
}
```

3. Script PetugasController.php

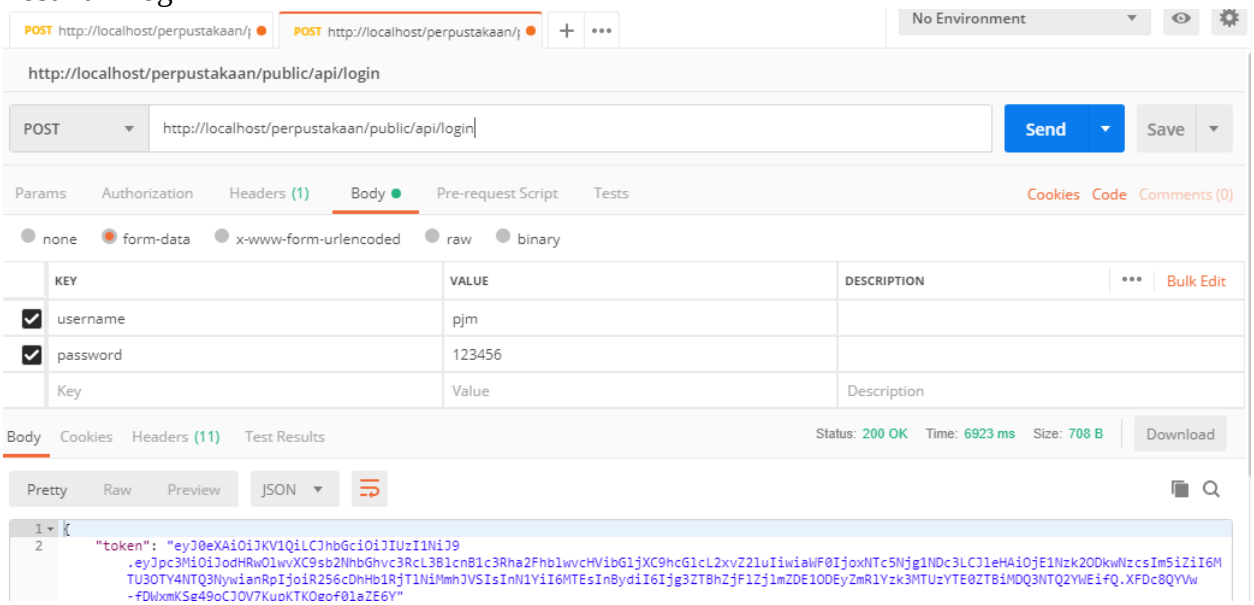
```php
<?php

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 PetugasController 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:255',
            'telp'=>'required|string|max:255',
            'username'=>'required|string|max:255',
            'password' => 'required|string|min:6|confirmed',
            'level' => 'required'
        ]);

```

```
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



5. Postman Login



6. Postman level