Login dan Register

1. Api

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

2. Controller

```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 UserController 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'));
    }
```

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

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

        $user = User::create([
            'nama_petugas' => $request->get('nama_petugas'),
            'alamat' => $request->get('alamat'),
            'no_telp' => $request->get('no_telp'),
            '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);
    }
```

```php

    public function getAuthenticatedUser()
    {
        try {

            if (! $user = JWTAuth::parseToken()->authenticate()) {
                return response()->json(['user_not_found'], 404);
            }

        } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {

            return response()->json(['token_expired'], $e->getStatusCode());

        } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {

            return response()->json(['token_invalid'], $e->getStatusCode());

        } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {

            return response()->json(['token_absent'], $e->getStatusCode());

        }

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

```

3. Model

```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', 'no_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 [];
    }
}
```

4. Hasil


```
{
    "user": {
        "nama_petugas": "Nanon",
        "alamat": "jl. blaem blaem",
        "no_telp": "088888888888",
        "username": "nanonthai",
        "level": "petugas",
        "updated_at": "2020-01-22 11:04:33",
        "created_at": "2020-01-22 11:04:33",
        "id": 8
    },
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
        .eyJpc3MiOiJodHRwOlwvXC9sb2NhbGhvc3RcL3BlcnB1czFcL3B1YmxpY1wvYXBpXC9yZWdpc3RlciIsImlhdCI6MTU3OTY5MTA3MywiZXhwI
        joxNTc5Njk0NjczLCJuYmYiOjE1Nzk2OTEwNzMsImp0aSI6IjZ1UTZSRlVYZ2FLeWRFazkiLCJzdWIiOjgsInBydiI6Ijg3ZTBhZjFlZjlmZDE
        1ODEyZmRlYzk3MTUzYTE0ZTBiMDQ3NTQ2YWEifQ.cvjmH5IjP15bTqQ6yT9BVMPEZK_Kk33d7AiMNxy7LeM"
}
```

```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
        .eyJpc3MiOiJodHRwOlwvXC9sb2NhbGhvc3RcL3BlcnB1czFcL3B1YmxpY1wvYXBpXC9sb2dpbiIsImlhdCI6MTU3OTY5MTQ1MywiZXhwIjoxNTc5Njk1MDUzLCJuYmY
        iOjE1Nzk2OTE0NTMsImp0aSI6IkZMODRFclpJR2pBS1JQaGMiLCJzdWIiOjEwLCJwcnYiOiI4N2UwYWYxZWY5ZmQxNTgxMmZkZWM5NzE1M2ExNGUwYjA0NzU0NmFhIn0
        .JaWAS2RGlJojoznOJQC8ZXSFY8K3lMw31Sq-zTL0PyM"
}
```


```
1  petugas
```