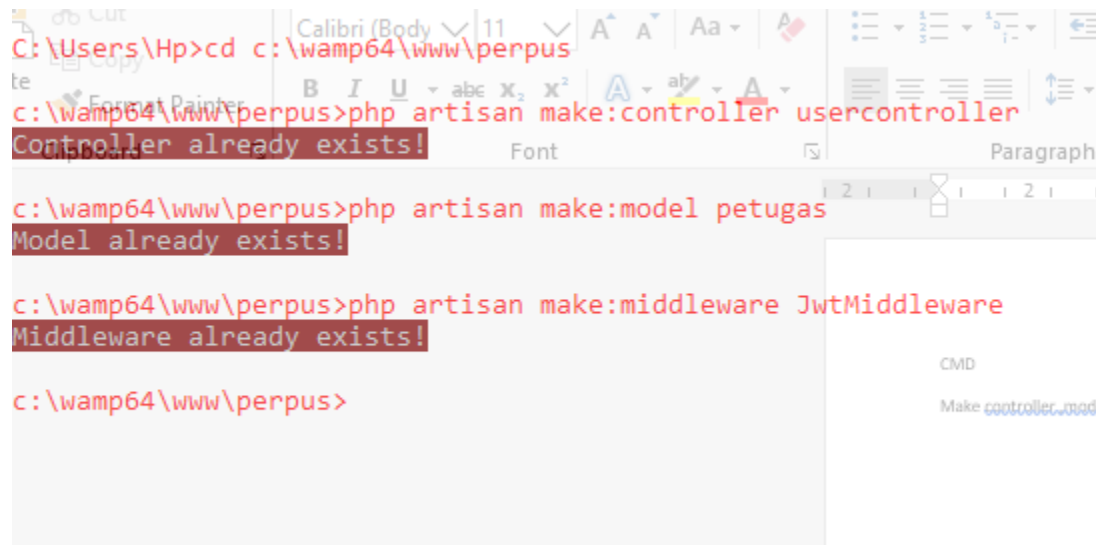


CMD

Make controller ,model ,dan middleware



```
C:\Users\Hp>cd c:\wamp64\www\perpus
c:\wamp64\www\perpus>php artisan make:controller usercontroller
Controller already exists!
c:\wamp64\www\perpus>php artisan make:model petugas
Model already exists!
c:\wamp64\www\perpus>php artisan make:middleware JwtMiddleware
Middleware already exists!
c:\wamp64\www\perpus>
```

Code

Model user

```
<?php

namespace App;

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

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $table="table_petugas";
    protected $fillable=['nama_petugas','alamat','no_telp','username','password',
'levels','created_at'];

    /**
     * The attributes that should be hidden for arrays.
     */
}
```

```

    *
    * @var array
    */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    public function getJWTIdentifier()
    {
        return $this->getKey();
    }
    public function getJWTCustomClaims()
    {
        return [];
    }
}

```

## Controller

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\User;
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'));
    }

    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'nama_petugas' => 'required|string|max:255',
            'no_telp'      => 'required|max:25',
            'alamat' => 'required|string|max:255',
            'username' => 'required|string|max:200',
            'password' => 'required|string|min:6|confirmed',
            'levels' => 'required|string|max:200',
        ]);

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

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

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

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

    public function getAuthenticatedUser()
    {
        try {

```

```

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

    } 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'));
}
}

```

Api.php

```

<?php

use Illuminate\Http\Request;

/*
|-----
| API Routes
|-----
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
Route::post('register', 'UserController@register');
Route::post('login', 'UserController@login');

```

login



KEY	VALUE	DESCRIPTION	*** Bulk Edit
<input checked="" type="checkbox"/> nama_petugas	hat		
<input checked="" type="checkbox"/> username	hat		
<input checked="" type="checkbox"/> password	aku123		
<input checked="" type="checkbox"/> password_confirmation	aku123		
<input checked="" type="checkbox"/> levels	kasir		
<input checked="" type="checkbox"/> no_telp	08888999		
<input checked="" type="checkbox"/> alamat	jl.goa		
Key	Value	Description	

Body
Cookies
Headers (10)
Test Results
Status: 201 Created
Time: 383ms
Size: 828 B
Save Response ▼

Pretty
Raw
Preview
Visualize BETA
JSON ▼

```

1  {
2    "user": {
3      "nama_petugas": "hat",
4      "no_telp": "08888999",
5      "alamat": "jl.goa",
6      "username": "hat"

```

```
7      "levels": "kasir",
8      "updated_at": "2020-01-23 13:26:59",
9      "created_at": "2020-01-23 13:26:59",
10     "id": 11
11   },
12   "token":
13     "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOiVvXC9wZXJwdXNcL2FwaVvvcmVnaXN0ZXIiLCJpYXQiOiJlNzk3ODYwMTksImV4cCI6MTU3OTc4OTYxOSwibm7mIjoxNTc5Nzg2MDE5LCJqdGkiOiJ2Y0tzME5Db29IRkZlIHhfZ2Iiwic3ViIjoxd45wicHJ2IjoiodDd1MGFmflVmQWZkMTU4MTJmZGVjOTcxNTNhMTRlMGItwNDc1NDZhYSJ9.FXIRmGvC83OpEuyPnk_6_5aUm-nQb-L6wIikIr05wL0"
```