

## Register

The screenshot shows a REST client interface with a POST request to `http://localhost/rental_mobil/public/api/register`. The request body is a JSON object with the following fields:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama_petugas	bagas	
<input checked="" type="checkbox"/> username	bagas	
<input checked="" type="checkbox"/> password	12345678	
<input checked="" type="checkbox"/> password_confirmation	12345678	
<input checked="" type="checkbox"/> level	admin	
Key	Value	Description

The response status is 201 Created, with a time of 3.11s and a size of 836 B. The response body is a JSON object:

```
{  "user": {    "nama_petugas": "bagas",    "level": "admin",    "username": "bagas",    "updated_at": "2020-04-08 10:32:52",    "created_at": "2020-04-08 10:32:52",    "id": 1  },  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi1wXC9sb2NhOGhvc3Rlc3JlbnRhbF9tb2JpbFwvYyIiwiaWF0IjoxNTg2MzQxOTc0LCJleHAiOiJlODYzNDU1NzQsIm5iZiI6MTU4NjM0MTk3NCwianRpIjoieWQwR0VPR2ZnQ1gzRTlIz1IsInN1YiI6MSwicHJ2IjoieYU4NTc1YjJmZzIiZm10dHh0TjJMTQ0MmZmOGE3NWU5YjA5ZTU3NyJ9.mKQ0sdvjNZz0ZV0ttnvg5Uxoq1r3-VbItYv2z1bImY"}
```

## Login

The screenshot shows a REST client interface with a POST request to `http://localhost/rental_mobil/public/api/login`. The request body is a JSON object with the following fields:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	bagas	
<input checked="" type="checkbox"/> password	12345678	
<input checked="" type="checkbox"/> password_confirmation	12345678	
Key	Value	Description

The response status is 200 OK, with a time of 989ms and a size of 683 B. The response body is a JSON object:

```
{  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi1wXC9sb2NhOGhvc3Rlc3JlbnRhbF9tb2JpbFwvYyIiwiaWF0IjoxNTg2MzQxOTc0LCJleHAiOiJlODYzNDU1NzQsIm5iZiI6MTU4NjM0MTk3NCwianRpIjoieWQwR0VPR2ZnQ1gzRTlIz1IsInN1YiI6MSwicHJ2IjoieYU4NTc1YjJmZzIiZm10dHh0TjJMTQ0MmZmOGE3NWU5YjA5ZTU3NyJ9.SX0yZ4WydFKgTaH-wXg7qtT1kc0pQjjQ2fd1RX2JtIA"}
```

## Script API

```
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::post('register', 'Petugas@register');
Route::post('login', 'Petugas@login');

Route::get("/", "Petugas@tampil")->middleware('jwt.verify');

Route::get('user', 'Petugas@getAuthenticatedUser')->middleware('jwt.verify');
```

## 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 PetugasModel extends Authenticatable implements JWTSubject
{
    use Notifiable;
    protected $table = 'petugas';

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

## Controller petugas

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\PetugasModel;
use App\User;
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',
            'level' => 'required|string',
            'username' => 'required|string|max:255',
            'password' => 'required|string|min:6|confirmed',
        ]);

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

        $user = PetugasModel::create([
            'nama_petugas' => $request->get('nama_petugas'),
            'level' => $request->get('level'),
            'username' => $request->get('username'),
            'password' => Hash::make($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'));
}

public function tampil()
{
    $data=['status'=>Auth::user()->nama_petugas];
    return response()->json($data);
}
}
```