

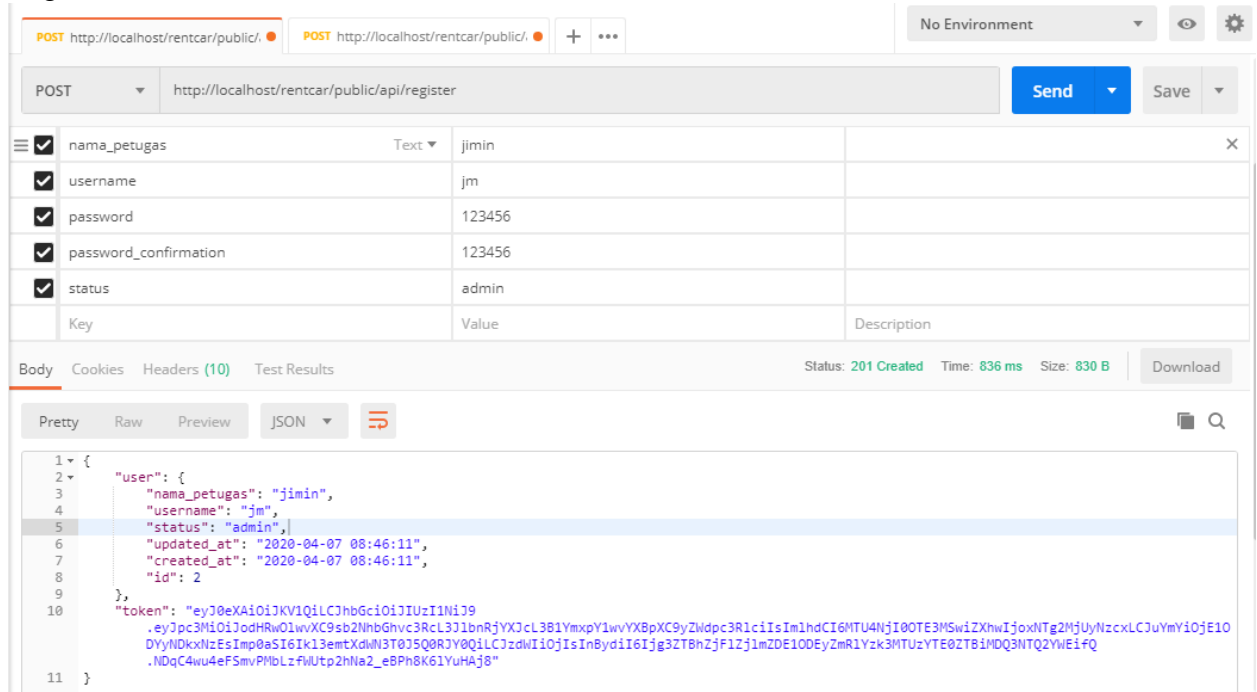
Nama : Talitha Sevrilla Duriga

No : 38

Kelas : XI Laravel

Tugas API Authentication

1. Register



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

Key	Value	Description
<input checked="" type="checkbox"/> nama_petugas	jimin	
<input checked="" type="checkbox"/> username	jim	
<input checked="" type="checkbox"/> password	123456	
<input checked="" type="checkbox"/> password_confirmation	123456	
<input checked="" type="checkbox"/> status	admin	

The response status is 201 Created, with a time of 836 ms and a size of 830 B. The response body is a JSON object:

```
1 {
2   "user": {
3     "nama_petugas": "jimin",
4     "username": "jim",
5     "status": "admin",
6     "updated_at": "2020-04-07 08:46:11",
7     "created_at": "2020-04-07 08:46:11",
8     "id": 2
9   },
10  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi1wXC9sb2NhbGhvc3Rlc3J1bnRjYXJzL3B1YmVycyY1wVYXBpXC9yZWdpd3R1ciIsIm1hdCI6MTU4NjI0OTE3MSw1ZXhwIjoxNjg2MjUyNzcxLCJyYmYiOiE1ODYyNDkxNDkxIiwiaWF0Ij0iMj02MDQ3NTQyYWEiFQ.NDqC4uu4eF5mvPMbLzFhUtp2hNa2_eBPh8K61YuHAj8"
11 }
```

2. Script register (Petugascontroller)

```
app > Http > Controllers > Petugascontroller.php
30 public function register(Request $request)
31 {
32     $validator = Validator::make($request->all(), [
33         'nama_petugas' => 'required|string|max:255',
34         'username' => 'required|string|max:255',
35         'password' => 'required|string|min:6|confirmed',
36         'status' => 'required'
37     ]);
38
39     if($validator->fails()){
40         return response()->json($validator->errors()->toJson(), 400);
41     }
42
43     $user = User::create([
44         'nama_petugas' => $request->get('nama_petugas'),
45         'username' => $request->get('username'),
46         'password' => Hash::make($request->get('password')),
47         'status' => $request->get('status')
48     ]);
49
50     $token = JWTAuth::fromUser($user);
51
52     return response()->json(compact('user','token'),201);
53 }
```

3. Login

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost/rentcar/public/api/login
- Body:** Form data with fields: username (jm), password (123456).
- Status:** 200 OK, Time: 712 ms, Size: 700 B.
- Response Body (JSON):**

```
{  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3ZWRhdGEiOiJodHRwOi1wXC9sb2NhbGhvc3RcL3J1bnRjYXJzL3B1Ym91dG8iLCJ1bm90eSI6IjE0ODYyNDkyMDMsImp0aSI6ImE2VGdXeu5hUXNGcnBTRUgiLCJzdiIjOiIsInBydiI6Ijg3ZTBhZjF1ZjJlM2DE10DEyZmRlYzE3MTUzYTE0ZTB1MDQ3NTQ2YmEiFQ.XIFV5qE85XXoRw7UfTzK-ZYRfrgfnFoxTT8b8Ad_fk"}
```

4. Script login (Petugascontroller)

```
app > Http > Controllers > Petugascontroller.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Tymon\JWTAuth\Exceptions\JWTException;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Support\Facades\Validator;
9  use JWTAuth;
10 use Auth;
11 use App\User;
12
13 class Petugascontroller extends Controller
14 {
15     public function login(Request $request)
16     {
17         $credentials = $request->only('username', 'password');
18
19         try {
20             if (! $token = JWTAuth::attempt($credentials)) {
21                 return response()->json(['error' => 'invalid_credentials'], 400);
22             }
23         } catch (JWTException $e) {
24             return response()->json(['error' => 'could_not_create_token'], 500);
25         }
26
27         return response()->json(compact('token'));
28     }
29 }
```

```

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'));
}

```

5. Api.php

```

routes > api.php
1  <?php
2
3  use Illuminate\Http\Request;
4
5  /*
6  |-----
7  | API Routes
8  |-----
9  |
10 | Here is where you can register API routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | is assigned the "api" middleware group. Enjoy building your API!
13 |
14 */
15
16 Route::middleware('auth:api')->get('/user', function (Request $request) {
17     return $request->user();
18 });
19 Route::post('register', 'Petugascontroller@register');
20 Route::post('login', 'Petugascontroller@login');|

```

6. Model (User.php)

```
app > 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', 'username', 'password', 'status'
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 }
```