

Nama : Mukhamad Rico Ramada

Kelas : XIRPL6

1. Register

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

- Method:** POST
- URL:** `http://localhost/rental_mobil/public/api/register`
- Status:** 201 Created
- Time:** 1683ms
- Size:** 823 B

The response body is displayed in JSON format:

```
1 {
2   "user": {
3     "nama_admin": "rafly",
4     "username": "apasaj12@gmail.com",
5     "role": "admin",
6     "alamat": "jl durian no37",
7     "id": 1
8   },
9   "token":
10  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi1wXC9sb2NhbnRhbF9tb2JpbFwvcHVibG1jXC9hcG1cL3JlZ21zdGVyIiwiaWF0IjoxNTg2NTE0ODVzLj1eHA10jE1ODY1MTg0NjQsIm51ZiI6MTU4NjUxNDg2NCwianRpIjo1QmJ0dEVoYkZPYzgyM2dFeCI6InN1YiI6MSwicHJ2Ijo1ODd1MGFmNmVmOmZkMTU4MTJmZGVjOTcxNTNhMTR1MG1wNDc1NDZhYSJ9.FduLvs_DJ-7HqYnmnQnc2sCpd6j3zIMSvyHEHifPXog"
```

2. Login

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

- Method:** POST
- URL:** `http://localhost/rental_mobil/public/api/login`
- Status:** 200 OK
- Time:** 1646ms
- Size:** 704 B

The response body is displayed in JSON format:

```
1 {
2   "token":
3   "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi1wXC9sb2NhbnRhbF9tb2JpbFwvcHVibG1jXC9hcG1cL2xvZ21uIiwiaWF0IjoxNTg2NTE0ODVzLj1eHA10jE1ODY1MTg1MjQsIm51ZiI6MTU4NjUxNDkyNCwianRpIjo1Y3hqQVZ4MM1ZW10cFF5UyIsInN1YiI6MSwicHJ2Ijo1ODd1MGFmNmVmOmZkMTU4MTJmZGVjOTcxNTNhMTR1MG1wNDc1NDZhYSJ9.yhUbhvDaY6MoS1R82062LLJZ31wSTZuFbLiOecqfUTw"
```

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\User;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Support\Facades\Validator;
9  use JWTAuth;
10 use Tymon\JWTAuth\Exceptions\JWTException;
11
12 class UserController extends Controller
13 {
14     public function login(Request $request)
15     {
16         $credentials = $request->only('username', 'password');
17
18         try {
19             if (!$token = JWTAuth::attempt($credentials)) {
20                 return response()->json(['error' => 'invalid_credentials'], 400);
21             }
22         } catch (JWTException $e) {
23             return response()->json(['error' => 'could_not_create_token'], 500);
24         }
25
26         return response()->json(compact('token'));
27     }
28
29     //-----
30     public function register(Request $request)
31     {
32         $validator = Validator::make($request->all(), [
33             'nama_admin' => 'required|string|max:255',
34             'role' => 'required|string|max:255',
35             'username' => 'required|string|email|max:255|unique:admin',
36             'password' => 'required|string|min:8|confirmed',
37             'alamat' => 'required|string|max:255',
38         ]);
39
40         if($validator->fails()){
41             return response()->json($validator->errors()->toJson(), 400);
42         }
43
44         $user = User::create([
45             'nama_admin' => $request->get('nama_admin'),
46             'username' => $request->get('username'),
47             'password' => Hash::make($request->get('password')),
48             'role' => $request->get('role'),
49             'alamat' => $request->get('alamat'),
50         ]);
51
52         $token = JWTAuth::fromUser($user);
53
54         return response()->json(compact('user','token'),201);
55     }
56
57     //-----
58     public function getAuthenticatedUser()
59     {
60         try {
61             if (!$user = JWTAuth::parseToken()->authenticate()) {
62                 return response()->json(['user_not_found'], 404);
63             }
64
65         } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
66             return response()->json(['token_expired'], $e->getStatusCode());
67
68         } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {

```

```

75
76     }
77
78     return response()->json(compact('user'));
79 }
80 }
81

```

API

```

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', 'UserController@register');
20  Route::post('login', 'UserController@login');
21  Route::get('book', 'BookController@book');
22
23  Route::get('bookall', 'BookController@bookAuth')->middleware('jwt.verify');
24  Route::get('user', 'UserController@getAuthenticatedUser')->middleware('jwt.verify');
25

```

GET

http://localhost/rental_mobil/public/api/user

Send

Save

Params

Authorization

Headers (11)

Body

Pre-request Script

Tests

Settings

Cookies

Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (11)

Test Results

Status: 200 OK

Time: 685ms

Size: 462 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

```
"user": {
  "id": 1,
  "nama_admin": "rafly",
  "role": "admin",
  "username": "apasaj12@gmail.com",
  "alamat": "jl durian no37"
}
```

```

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 = 'admin';
14     protected $primaryKey = 'id';
15     public $timestamps = false;
16     /**
17      * The attributes that are mass assignable.
18      *
19      * @var array
20      */
21     protected $fillable = [
22         'nama_admin', 'role', 'username', 'password', 'alamat',
23     ];
24
25     /**
26      * The attributes that should be hidden for arrays.
27      *
28      * @var array
29      */
30     protected $hidden = [
31         'password', 'remember_token',
32     ];
33
34     /**
35      * The attributes that should be cast to native types.
36      *
37      * @var array
38      */
39     protected $casts = [
40         'email_verified_at' => 'datetime',
41     ];
42     public function getJWTIdentifier()
43     {
44         return $this->getKey();
45     }
46     public function getJWTCustomClaims()
47     {
48         return [];
49     }
50 }
51

```