

Tampil sukses

Register

The screenshot shows a Postman interface with a POST request to `rental/api/register`. The request body is a JSON object with the following fields:

Key	Value	Description
level	bawahan	
alamat	jl.gunung	
username	bakpao	
password	baksoo11	
password_confirmation	baksoo11	

The response is a JSON object with the following fields:

```
1 {
2   "user": {
3     "nama_petugas": "saimin",
4     "username": "bakpao",
5     "level": "bawahan",
6     "alamat": "jl.gunung",
7     "updated_at": "2020-04-10 16:17:19",
8     "created_at": "2020-04-10 16:17:19",
9     "id": 1
10  },
11  "token":
12  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOiJvXC9yZW50YmxcL2FwaVwvcmVnaXN0ZSIiLCJpYXQiOiJlODY1MzU0NDAsImV4cCI6MTU4NjUzOTA0MCwibmJmIjoxNTg2NTM1NDQwLCJqdGkiOiJTMQ4ZldPUHdJRjVEOW5xIiwic3ViIjoxLCJwcnYiOiI4N2UwYVYxZWY5ZmQxNTgxNmZkZWp5NzE1M2ExNGUwYjA0NzU0NmFhIn0.55mRduYR-LxTwjXNdlqD_8cqmwF-ZsLQi867pAdZRiU"
```

Login

The screenshot shows a Postman interface with a POST request to `rental/api/login?username=bakpao&password=baksoo11`. The request body is a JSON object with the following fields:

KEY	VALUE	DESCRIPTION
username	bakpao	
password	baksoo11	

The response is a JSON object with the following fields:

```
1 {
2   "token":
3   "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOiJvXC9yZW50YmxcL2FwaVwvcmVnaXN0ZSIiLCJpYXQiOiJlODY1MzU0NDAsImV4cCI6MTU4NjUzOTA0MCwibmJmIjoxNTg2NTM1NDQwLCJqdGkiOiJTMQ4ZldPUHdJRjVEOW5xIiwic3ViIjoxLCJwcnYiOiI4N2UwYVYxZWY5ZmQxNTgxNmZkZWp5NzE1M2ExNGUwYjA0NzU0NmFhIn0.dYctk3IOX915oMc8IfNGR7eV1Ue1IXCfUsmPLkd18Io"
```

Script

User model

```
<?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="petugas";
    protected $fillable=[ 'nama_petugas','username','password','level','alamat'];

    /**
     * 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 petugas

```
<?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 Controllerpetugas 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',
                'username' => 'required|string|max:200',
                'password' => 'required|string|min:6|confirmed',
            ]
        );

        if ($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

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

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

        return response()->json(compact('token'));
    }
}
```

```

        'level'=>'required|string|max:200',
        'alamat'=>'required'
    ]);

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

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

    $token = JWTAuth::fromUser($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

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();
});

//petugas
Route::post('login', 'Controllerpetugas@login');
Route::post('register', 'Controllerpetugas@register');
```