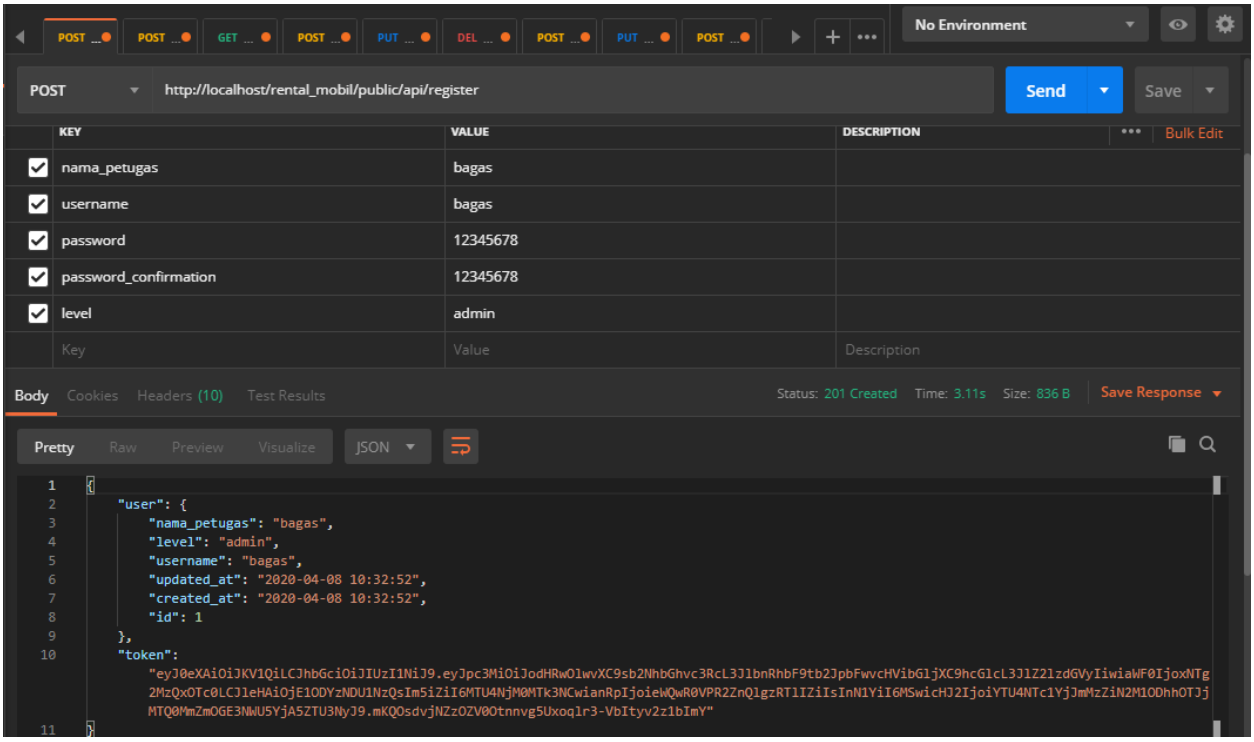
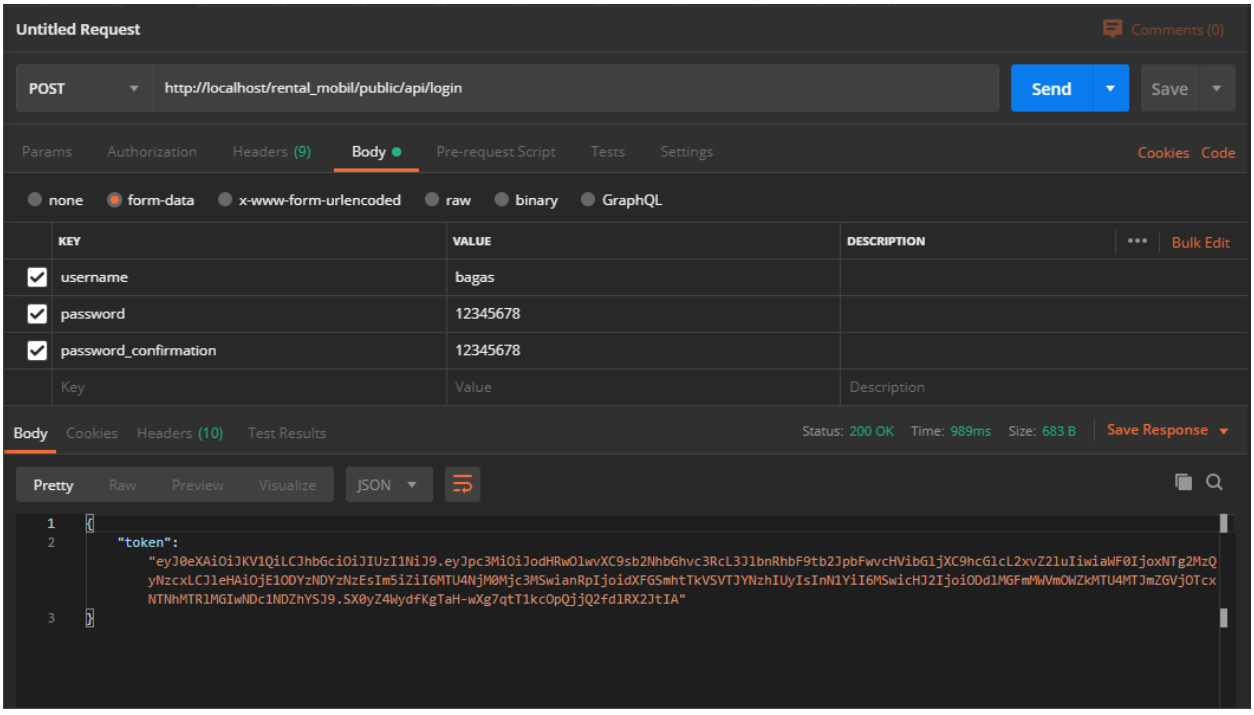
Muhammad Bagas Ramadhan / 19

Manual Book

Register



Login



Script API

```php
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
<?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
<?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);
    }
```

```php
    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);
    }
}
```