Luthfarras Tresanizega / XIR6 / 17 Tugas 4 JWT login-register

-membuat controller, model, dan middleware

```
D:\xampp\htdocs\projekperpus>php artisan make:controller PetugasController Controller created successfully.

D:\xampp\htdocs\projekperpus>php artisan make:model Petugass
Model created successfully.

D:\xampp\htdocs\projekperpus>php artisan make:middleware middlewarejwt
Middleware created successfully.
```

-api.php

```
Route::post('register', 'PetugasController@register');
Route::post('login', 'PetugasController@login');
Route::get('/', function(){
   return Auth::user()->level;
})->middleware('jwt.verify');
Route::get('user', 'PetugasController@getAuthenticatedUser')-
>middleware('jwt.verify');
```

-PetugasController

```
<?php
namespace App\Http\Controllers;
use App\Petugas;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;

class PetugasController extends Controller
{
   public function login(Request $request)
   {
        $credentials = $request->only('username', 'password');

        try {
            if (! $token = JWTAuth::attempt($credentials)) {
        }
}
```

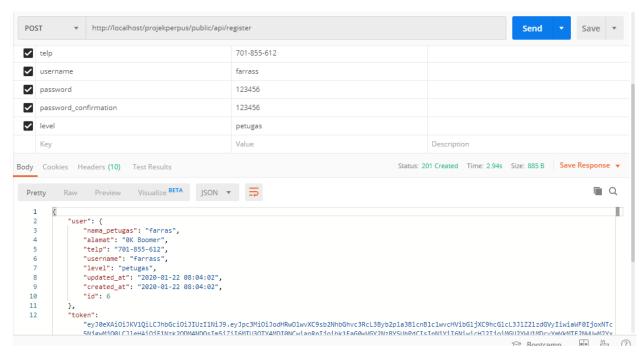
```
return response()->json(['error' => 'invalid_credentials'], 400);
        }
    } catch (JWTException $e) {
        return response()->json(['error' => 'could_not_create_token'], 500);
    }
    return response()->json(compact('token'));
}
public function register(Request $request)
    $validator = Validator::make($request->all(), [
        'nama_petugas' => 'required|string|max:255',
        'alamat' => 'required|string|max:255',
        'telp' => 'required|string|max:255',
        'username' => 'required|string|max:255|',
        'password' => 'required|string|min:6|confirmed',
        'level' => 'required|string|max:255',
    ]);
    if($validator->fails()){
        return response()->json($validator->errors()->toJson(), 400);
    }
    $user = Petugas::create([
        'nama_petugas' => $request->get('nama_petugas'),
        'alamat' => $request->get('alamat'),
        'telp' => $request->get('telp'),
        'username' => $request->get('username'),
        'password' => Hash::make($request->get('password')),
        'level' => $request->get('level'),
    ]);
    $token = JWTAuth::fromUser($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'));
}
```

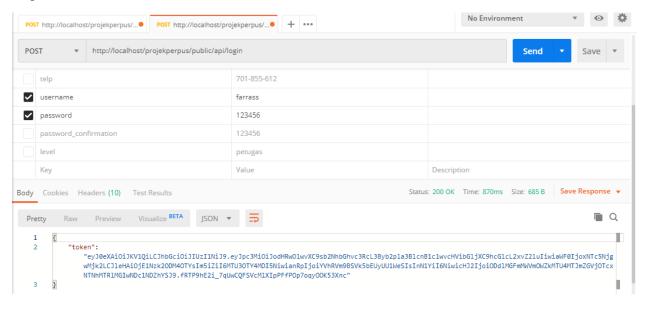
-model petugas

```
* 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 [];
 * The attributes that should be cast to native types.
 * @var array
protected $casts = [
    'email_verified_at' => 'datetime',
];
```

-register postman



-login postman



-level postman

