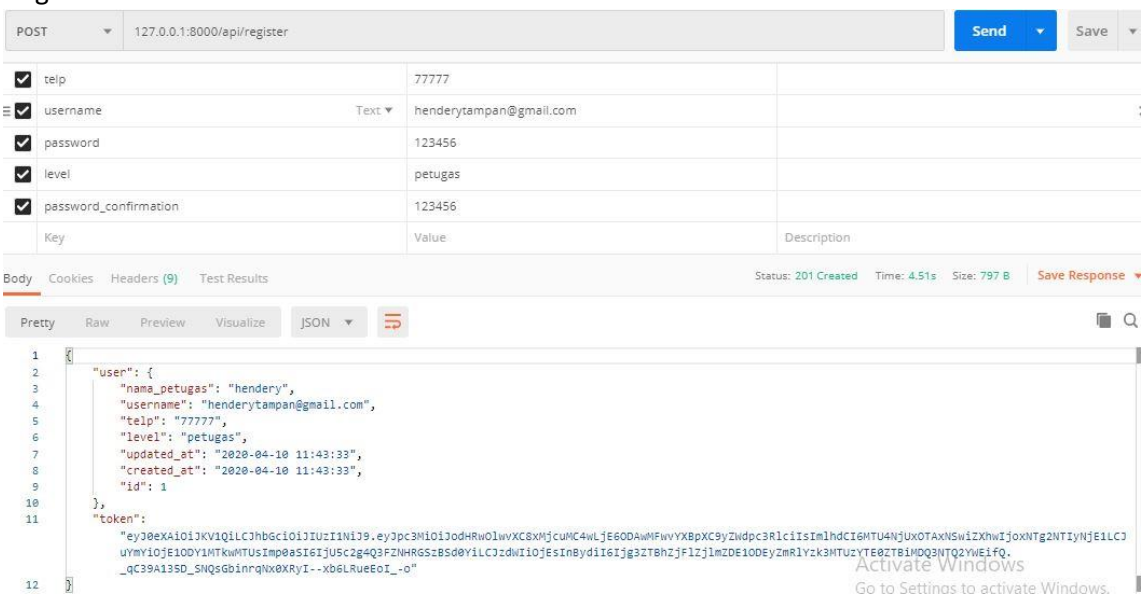
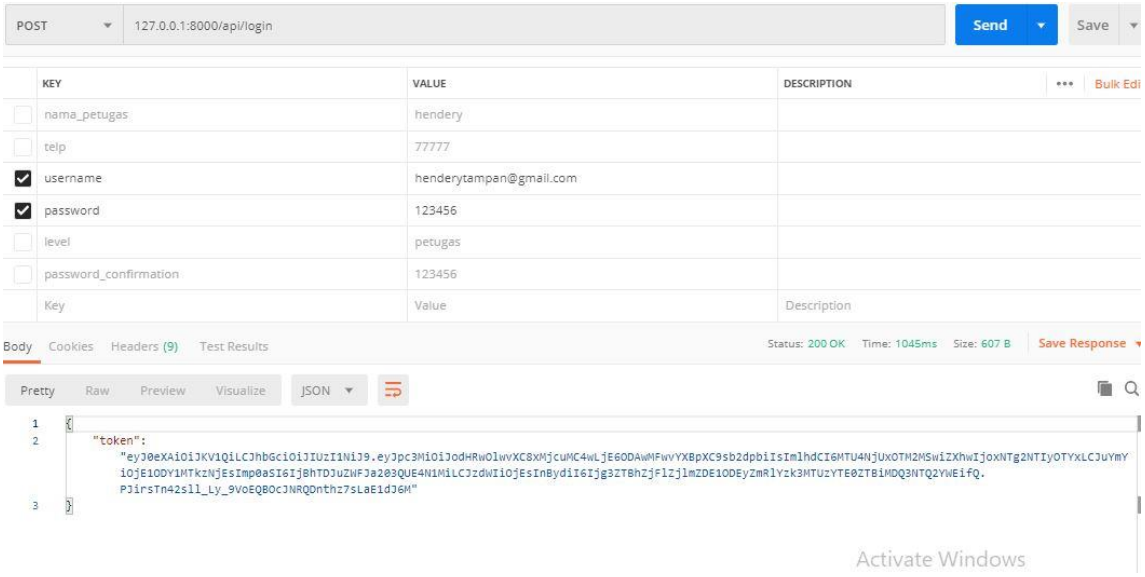
1. Register



Penjelasan : Di halaman register ini kita akan mendapat token untuk login

2. Login



Penjelasan : Di halaman login kita juga mendapat token yaitu token untuk mengakses halaman selanjutnya

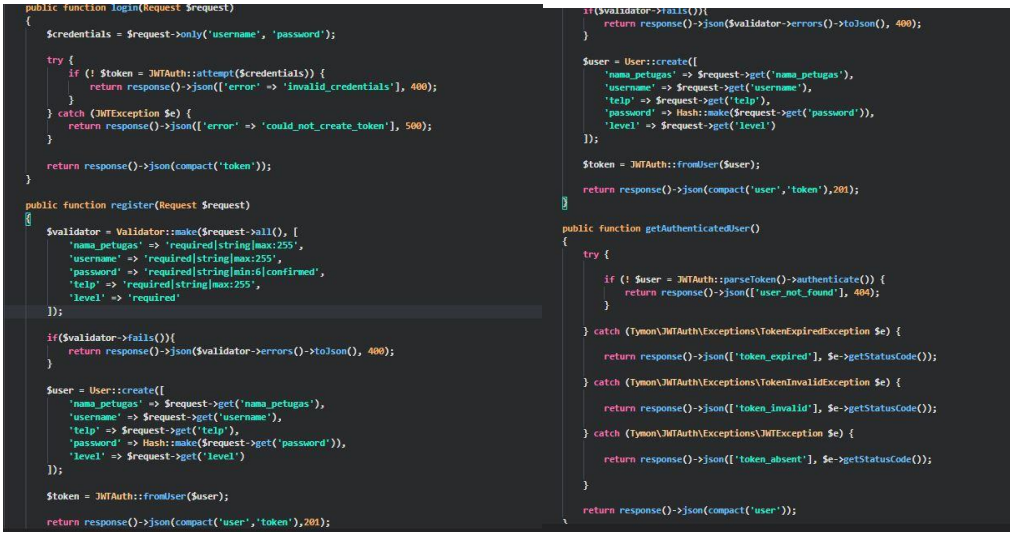
3. Api

```php
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
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');
```

Penjelasan : Disini ada untuk memproses request yang dibuat

4. Controller



```php
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:255',
        'password' => 'required|string|min:6|confirmed',
        'telp' => 'required|string|max:255',
        'level' => 'required'
    ]);

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

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

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

    return response()->json(compact('user','token'),201);
}
```

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

    $user = User::create([
        'nama_petugas' => $request->get('nama_petugas'),
        'username' => $request->get('username'),
        'telp' => $request->get('telp'),
        'password' => Hash::make($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'));
}
```

Penjelasan : di controller ini terdapat script untuk mengatur output di postman

5. Model

```php
class User extends Authenticatable implements JWTSubject
{
    use Notifiable;
    protected $table="table_petugas";
    //protected $primaryKey="id";
    //public $timestamps=false;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'nama_petugas', 'telp', '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 [];
    }
}
```

Penjelasan : model ini berisi untuk mengatur pada input dan memberi sumber untuk inputan (table database)