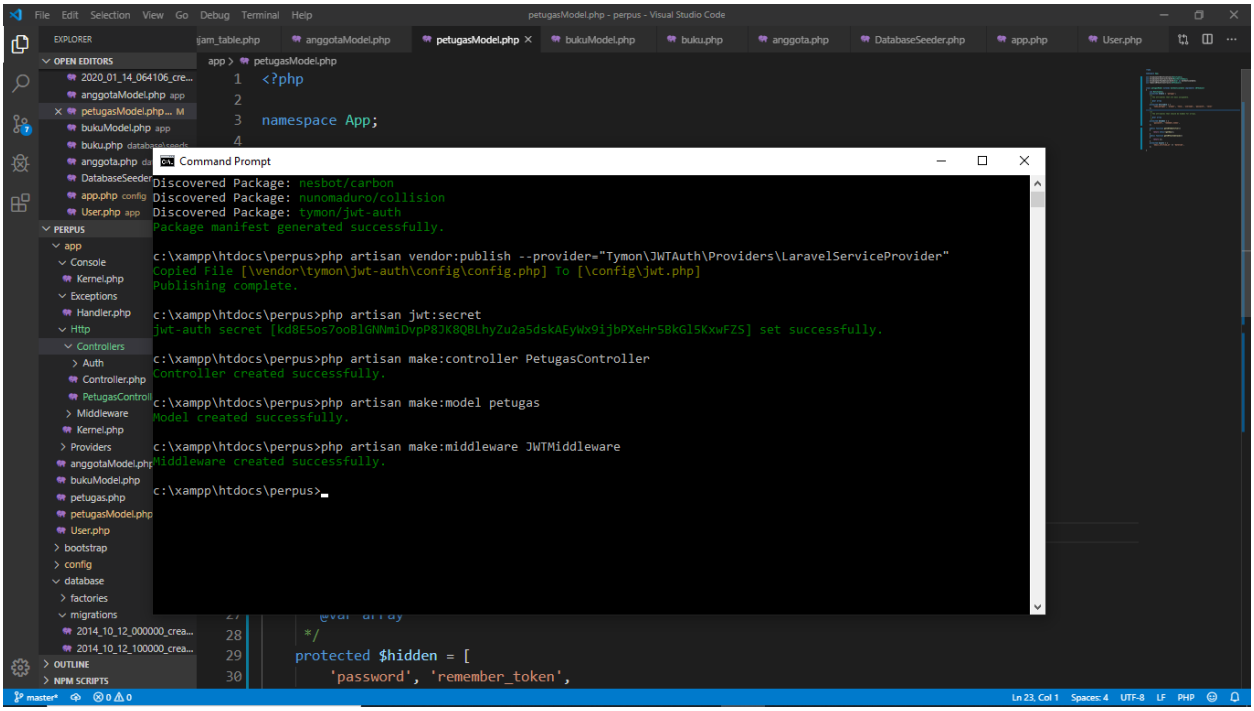
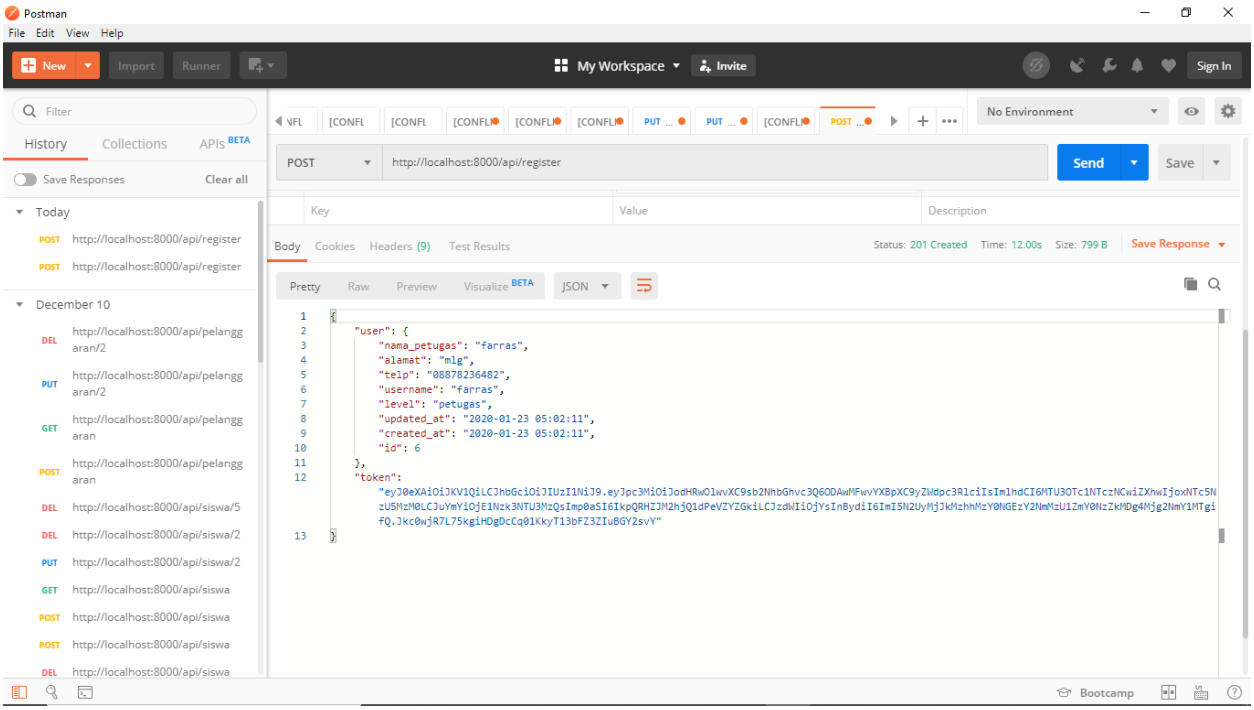
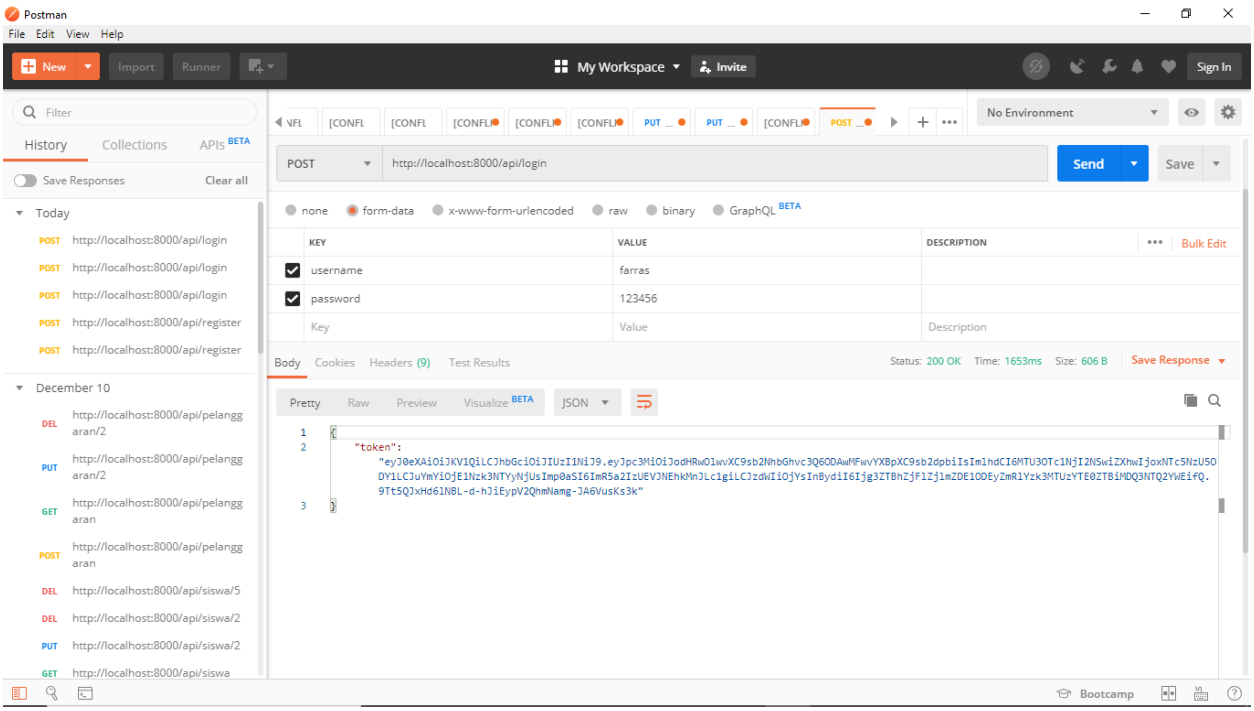
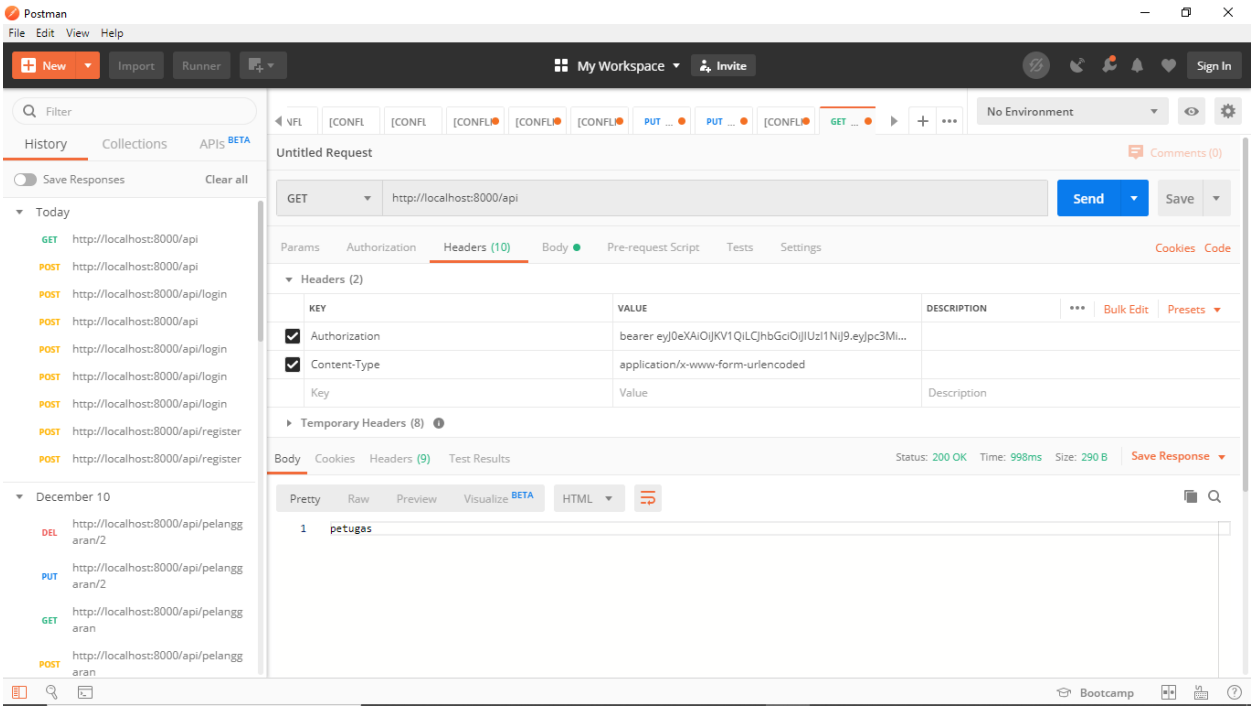

Command Prompt
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: tymon/jwt-auth
Package manifest generated successfully.
c:\xampp\htdocs\perpus>php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
Copied File [\vendor\tymon\jwt-auth\config\config.php] To [\config\jwt.php]
Publishing complete.
c:\xampp\htdocs\perpus>php artisan jwt:secret
jwt-auth secret [kd8E5o57ooBlGNNmiDvpP8JK8QBLhyZu2a5dskAEyWx9ijbPXeHr5BkGl5KxwFZS] set successfully.
c:\xampp\htdocs\perpus>php artisan make:controller PetugasController
Controller created successfully.
c:\xampp\htdocs\perpus>php artisan make:model petugas
Model created successfully.
c:\xampp\htdocs\perpus>php artisan make:middleware JWTMiddleware
Middleware created successfully.
c:\xampp\htdocs\perpus>


Postman
New
Import
Runner
My Workspace
Invite
Sign In
History
Collections
APIs BETA
Save Responses
Clear all
Today
POST http://localhost:8000/api/register
POST http://localhost:8000/api/register
December 10
DEL http://localhost:8000/api/pelanggaran/2
PUT http://localhost:8000/api/pelanggaran/2
GET http://localhost:8000/api/pelanggaran
POST http://localhost:8000/api/pelanggaran
DEL http://localhost:8000/api/siswa/5
DEL http://localhost:8000/api/siswa/2
PUT http://localhost:8000/api/siswa/2
GET http://localhost:8000/api/siswa
POST http://localhost:8000/api/siswa
POST http://localhost:8000/api/siswa
DEL http://localhost:8000/api/siswa
No Environment
POST
http://localhost:8000/api/register
Send
Save
Key
Value
Description
Body
Cookies
Headers (9)
Test Results
Status: 201 Created
Time: 12.00s
Size: 799 B
Save Response
Pretty
Raw
Preview
Visualize BETA
JSON
{
"user": {
"nama_petugas": "farras",
"alamat": "mlg",
"telp": "08878236482",
"username": "farras",
"level": "petugas",
"updated_at": "2020-01-23 05:02:11",
"created_at": "2020-01-23 05:02:11",
"id": 6
},
"token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOlwvXC9sb2NhbGhvc3Q6ODAwMFwvYXBpXC9yZWdpc3RlciIsImlhdCI6MTU3OTc1NTczNCwiZXhwIjoxNTc5NzU5MzM0LCJuYmYiOjE1Nzk3NTU3MzQsImp0aSI6IkpQRHZJM2hjQ1dPeVZYZGkiLCJzdWIiOjYsInBydiI6ImI5N2UyMjJkMzhhMzY0NGEzY2NmMzU1ZmY0NzZkMDg4Mjg2NmY1MTgifQ.Jkc0wjR7L75kgiHDgDcCq01KkyT13bFZ3ZIuBGY2svY"
}
Bootcamp





Petugascontroller.php

```php
<?php

namespace App\Http\Controllers;
```

```php
use App\petugasModel;
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|confirm
ed',
            'level' => 'required|string|max:255',
        ]);

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

        $user = petugasModel::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\TokenExpiredEx
ception $e) {

            return response()-
>json(['token_expired'], $e->getStatusCode());

        } catch (Tymon\JWTAuth\Exceptions\TokenInvalidEx
ception $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'));
    }
}
```

JWTMiddleware.php

```php
<?php

namespace App\Http\Middleware;

use Closure;
use JWTAuth;
use Exception;
use Tymon\JWTAuth\Http\Middleware\BaseMiddleware;


class JWTMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        try {
            $user = JWTAuth::parseToken()->authenticate();
        } catch (Exception $e) {
            if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenInvalidException){
                return response()->json(['status' => 'Token is Invalid']);
            }else if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenExpiredException){
```

```
                return response()-
>json(['status' => 'Token is Expired']);
            }else{
                return response()-
>json(['status' => 'Authorization Token not found']);
            }
        }

        return $next($request);
    }
}
```

Petugasmodel.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 JW
TSubject
{
    use Notifiable;
    protected $table = 'petugas';
    /**
     * The attributes that are mass assignable.
     *
```

```
     * @var array
     */
    protected $fillable = [
        'nama_petugas', 'alamat', 'telp', 'username', 'p
assword', '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 [];
    }
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

}
```