

---

# POLITEKNIK NEGERI LHOKSEUMAWE

JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER

TEKNOLOGI REKAYASA KOMPUTER JARINGAN


---




---


## TIK-6555 — WEBSITE DAN SISTEM MOBILE

---

DOSEN PENGAMPU: Muhammad Davi, S.Kom., M.Cs. 

Gedung Terpadu Jurusan Teknologi Informasi dan Komputer 

muhammad.davi@pnl.ac.id 

2024-2025 Ganjil 

## Daftar Isi

<b>1</b>	<b>Persiapan Lingkungan Kerja</b>	<b>2</b>
1.1	Kebutuhan Aplikasi dan Tools . . . . .	2
1.2	Instalasi Laradock . . . . .	2
<b>2</b>	<b>Membuat Restful API Login</b>	<b>2</b>
<b>3</b>	<b>Membuat Restful API CRUD Mahasiswa</b>	<b>2</b>

---

## 1 — PERSIAPAN LINGKUNGAN KERJA

---

# 1 Persiapan Lingkungan Kerja












Persiapan lingkungan kerja yang dimaksud adalah persiapan aplikasi dan tools yang digunakan untuk mengembangkan aplikasi menggunakan Laravel. Pada matakuliah ini kita menggunakan Laradock atau Laragon sebagai lingkungan kerja.



## Note:

Dapat memilih salah satu antara Laradock dan Laragon. Disarankan menggunakan Laradock, namun jika komputer/laptop tidak memungkinkan maka dapat menggunakan Laragon.

## 1.1 Kebutuhan Aplikasi dan Tools

-  **Sistem Operasi:**  Windows,  Linux, **atau**  macOS
-  **IDE:** Visual Studio Code
-  **Web Browser:** Google Chrome
-  **GitHub:** Web-based version control repository.
-  **Docker:** Memanajemen aplikasi-aplikasi untuk lingkungan pengembangan aplikasi Laravel.
-  Markdown untuk menulis laporan.
-  Git untuk version control system.
-  Postman untuk uji coba (*testing*) API

## 1.2 Instalasi Laradock

---

## 2 — MEMBUAT RESTFUL API LOGIN

---

## 2 Membuat Restful API Login

Ikuti langkah-langkah berikut ini.

### a) Membuat Data Seeder User

Membuat class seeder user dengan perintah: `php artisan make:seeder UserSeeder`

Buka file `UserSeeder` dan tambahkan kode berikut.

```
1 public function run(): void
2 {
3     DB::table('users')->insert([
4         'name'      => 'Administrator',
5         'email'     => 'admin@gmail.com',
6         'password'  => Hash::make('password'),
7     ]);
8 }
```

Jalankan seeder dengan perintah: `php artisan db:seed --class=UserSeeder`

### b) Install dan konfigurasi JWT (Json Web Token)

Instalasi JWT dengan perintah: `composer require tyson/jwt-auth:2.1.1`

Publish konfigurasi file dengan perintah:

```
php artisan vendor:publish --provider="Tyson\JWTAuth\Providers
\LaravelServiceProvider"
```

Generate Secret Key dengan perintah: `php artisan jwt:secret`

Konfigurasi guard API pada file `config/auth.php` seperti kode berikut ini.

```
1 'guards' => [
2     'web' => [
3         'driver' => 'session',
4         'provider' => 'users',
5     ],
6
7     'api_admin' => [
8         'driver' => 'jwt',
9         'provider' => 'users',
10        'hash' => false,
11    ],
12 ],
```

Konfigurasi model user pada file `app/Models/User.php`

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Foundation\Auth\User as Authenticatable;
7 use Illuminate\Notifications\Notifiable;
8 use Tymon\JWTAuth\Contracts\JWTSubject;
9
10 class User extends Authenticatable implements JWTSubject
11 {
12     use HasFactory, Notifiable;
13 }
```

```

14     protected $fillable = [
15         'name',
16         'email',
17         'password',
18     ];
19
20     protected $hidden = [
21         'password',
22         'remember_token',
23     ];
24
25     protected function casts(): array
26     {
27         return [
28             'email_verified_at' => 'datetime',
29             'password' => 'hashed',
30         ];
31     }
32
33     public function getJWTIdentifier()
34     {
35         return $this->getKey();
36     }
37
38     public function getJWTCustomClaims()
39     {
40         return [];
41     }
42 }

```

- c) Membuat controller API login dengan perintah: `php artisan make:controller Api/Admin/LoginController` kemudian sesuaikan kode seperti berikut ini pada file `app/Http/Controllers/Api/Admin`.

```

1 <?php
2
3 namespace App\Http\Controllers\Api\Admin;
4
5 use Illuminate\Http\Request;
6 use Tymon\JWTAuth\Facades\JWTAuth;
7 use App\Http\Controllers\Controller;
8 use Illuminate\Support\Facades\Validator;
9
10 class LoginController extends Controller
11 {
12     public function index(Request $request)
13     {
14         $validator = Validator::make($request->all(), [
15             'email' => 'required|email',
16             'password' => 'required',
17         ]);
18
19         if ($validator->fails()) {
20             return response()->json($validator->errors(), 422);
21         }
22
23         $credentials = $request->only('email', 'password');
24

```

```

25         if(!$token = auth()->guard('api_admin')->attempt($credentials)) {
26             return response()->json([
27                 'success' => false,
28                 'message' => 'Email or Password is incorrect'
29             ], 401);
30
31         }
32
33         return response()->json([
34             'success' => true,
35             'user'     => auth()->guard('api_admin')->user(),
36             'token'    => $token
37         ], 200);
38     }
39
40     public function getUser()
41     {
42         return response()->json([
43             'success' => true,
44             'user'     => auth()->guard('api_admin')->user()
45         ], 200);
46     }
47
48     public function refreshToken(Request $request)
49     {
50         $refreshToken = JWTAuth::refresh(JWTAuth::getToken());
51         $user = JWTAuth::setToken($refreshToken)->toUser();
52         $request->headers->set('Authorization', 'Bearer '.$refreshToken);
53
54         return response()->json([
55             'success' => true,
56             'user'     => $user,
57             'token'    => $refreshToken,
58         ], 200);
59     }
60
61     public function logout()
62     {
63         $removeToken = JWTAuth::invalidate(JWTAuth::getToken());
64
65         return response()->json([
66             'success' => true,
67         ], 200);
68     }
69 }
70 }

```

**d)** Membuat route dengan perintah: `php artisan install:api` kemudian edit file `routes/api.php` dengan kode berikut ini.

```

1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5
6 Route::prefix('admin')->group(function () {
7
8     Route::post('/login', [App\Http\Controllers\Api\Admin>LoginController::class]

```



```

9      , 'index', ['as' => 'admin']);
10
11      Route::group(['middleware' => 'auth:api_admin'], function() {
12          Route::get('/user', [App\Http\Controllers\Api\Admin\LoginController::↵
class, 'getUser', ['as' => 'admin']]);
13          Route::get('/refresh', [App\Http\Controllers\Api\Admin\LoginController::↵
class, 'refreshToken', ['as' => 'admin']]);
14          Route::post('/logout', [App\Http\Controllers\Api\Admin\LoginController::↵
class, 'logout', ['as' => 'admin']]);
15      });
16
17 });

```

**e)** Uji coba dengan Postman.

---

## 3 — MEMBUAT RESTFUL API CRUD

### MAHASISWA

---

### 3 Membuat Restful API CRUD Mahasiswa

**a) Membuat Migration Mahasiswa**

Untuk membuat migration mahasiswa jalankan perintah berikut: `php artisan make:migration create_mahasiswas_table`

**b) Membuat Model Mahasiswa**

Sedangkan untuk membuat model Mahasiswa jalankan perintah: `php artisan make:model Mahasiswa`

**c) Membuat Resources Mahasiswa**

Untuk membuat resource mahasiswa dapat dilakukan dengan perintah: `php artisan make:resource MahasiswaResource`

**d) Membuat Controller Mahasiswa**

Buat controller mahasiswa dengan perintah berikut: `php artisan make:controller Api/Admin/MahasiswaController`

**Lengkapi fungsi index dengan kode berikut**

```
1 public function index(Request $request)
2 {
3     $mahasiswas = Mahasiswa::when(request()->q, function($mahasiswas) {
4         $mahasiswas = $mahasiswas->where('name', 'like', '%'. request()->q . '%');
5     }->latest()->paginate(5);
6
7     return new MahasiswaResource(true, 'List Data Mahasiswa', $mahasiswas);
8 }
```

**Lengkapi fungsi store dengan kode berikut**

```
1 public function store(Request $request)
2 {
3     $validator = Validator::make($request->all(), [
4         'nim' => 'required|unique:mahasiswas',
5     ]);
6
7     if ($validator->fails()) {
8         return response()->json($validator->errors(), 422);
9     }
10
11     $mahasiswa = Mahasiswa::create([
12         'nama' => $request->name,
13         'nim' => $request->nim,
14     ]);
15
16     if ($mahasiswa) {
17         return new MahasiswaResource(true, 'Data Mahasiswa Berhasil Disimpan!', $mahasiswa);
18     }
19
20     return new MahasiswaResource(false, 'Data Mahasiswa Gagal Disimpan!', null);
21 }
```

**Lengkapi fungsi show dengan kode berikut**

```

1 public function show(Mahasiswa $mahasiswa)
2 {
3     if($mahasiswa) {
4         return new MahasiswaResource(true, 'Detail Data Mahasiswa!', ↵
5         $mahasiswa);
6     }
7     return new MahasiswaResource(false, 'Detail Data Mahasiswa Tidak ↵
8     Ditemukan!', null);
9 }

```

### Lengkapi fungsi update dengan kode berikut

```

1 public function update(Request $request, Mahasiswa $mahasiswa)
2 {
3     $validator = Validator::make($request->all(), [
4         'name' => 'required|unique:categories,name,',$category->id,
5     ]);
6
7     if ($validator->fails()) {
8         return response()->json($validator->errors(), 422);
9     }
10
11     $mahasiswa->update([
12         'name' => $request->name,
13     ]);
14
15     if($mahasiswa) {
16         return new MahasiswaResource(true, 'Data Mahasiswa Berhasil ↵
17         Diupdate!', $mahasiswa);
18     }
19     return new MahasiswaResource(false, 'Data Mahasiswa Gagal Diupdate↵
20     !', null);
21 }

```

### Lengkapi fungsi destroy dengan kode berikut

```

1 public function destroy(Mahasiswa $mahasiswa)
2 {
3     if($mahasiswa->delete()) {
4         return new MahasiswaResource(true, 'Data Mahasiswa Berhasil ↵
5         Dihapus!', null);
6     }
7     return new MahasiswaResource(false, 'Data Mahasiswa Gagal Dihapus↵
8     !', null);
9 }

```

**e)** Tambahkan route pada file routes/api.php dibawah group middleware auth:api\_admin.

```

1 Route::apiResource('/mahasiswa', App\Http\Controllers\Api\Admin\↵
    MahasiswaController::class, ['except' => ['create', 'edit'], 'as' => 'admin↵
    ']);

```

**f)** Uji coba dengan Postman.

- Uji coba list mahasiswa (function index)  
URL: domain.test/api/admin/mahasiswas  
Method: GET  
Header:
  - Accept: application/json
  - Content-Type: application/json
  - Authorization: Bearet Token
- Uji coba simpan data mahasiswa (function store)  
URL: domain.test/api/admin/mahasiswas  
Method: POST  
Header:
  - Accept: application/json
  - Content-Type: application/json
  - Authorization: Bearet Token  
Body: form-data
  - nama: Faiza Zulaikha
  - nim: 2020
  - prodi\_id: 2
- Uji coba detail mahasiswa (function show)  
URL: domain.test/api/admin/mahasiswas/1  
Method: GET  
Header:
  - Accept: application/json
  - Content-Type: application/json
  - Authorization: Bearet Token
- Uji coba update mahasiswa (function update)  
URL: domain.test/api/admin/mahasiswas/1  
Method: POST  
Header:
  - Accept: application/json
  - Content-Type: application/json
  - Authorization: Bearet Token  
Body: form-data
  - nama: Faiza Zulaikha
  - nim: 2020
  - prodi\_id: 2
  - \_method: PATCH
- Uji coba hapus mahasiswa (function destroy)  
URL: domain.test/api/admin/mahasiswas/1  
Method: DELETE  
Header:
  - Accept: application/json
  - Content-Type: application/json
  - Authorization: Bearet Token