

---

**POLITEKNIK NEGERI LHOKSEUMAWE**  
JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER  
TEKNOLOGI REKAYASA KOMPUTER JARINGAN

---



---

**TIK-6555 — WEBSITE DAN SISTEM MOBILE**

---

DOSEN PENGAMPU: Muhammad Davi

Gedung Terpadu Jurusan Teknologi Informasi dan Komputer

muhammad.davi@pnl.ac.id

2025-2026 Ganjil

## **Daftar Isi**

<b>1 Persiapan Lingkungan Kerja</b>	<b>2</b>
1.1 Kebutuhan Aplikasi dan Tools . . . . .	2
1.2 Instalasi . . . . .	2
1.3 Membuat Project Laravel . . . . .	2
<b>2 Membuat Restful API Login</b>	<b>2</b>
<b>3 Membuat Restful API CRUD Mahasiswa</b>	<b>2</b>
<b>4 Tugas</b>	<b>7</b>
<b>5 Kriteria Penilaian</b>	<b>7</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 sebagai lingkungan kerja.

## 1.1 Kebutuhan Aplikasi dan Tools

- ☒ **Sistem Operasi:** Windows, Linux, atau macOS
- ☒ **IDE:** JetBrains Toolbox dan PhpStorm
- ☒ **Web Browser:** Google Chrome
- ☒ **GitHub:** Web-based version control repository.
- ☒ **Markdown** untuk menulis laporan.
- ☒ **Git** sebagai version control.
- ☒ **Docker** untuk mengemas dan menjalankan aplikasi-aplikasi dalam sebuah lingkungan terisolasi (container) seperti nginx dan mysql.
- ☒ **Composer** untuk mengelola dan menginstal library serta paket eksternal yang dibutuhkan untuk proyek secara otomatis.



### Note:

Kebutuhan aplikasi dan tool dapat disesuaikan dengan speksifikasi atau kondisi di laptop masing-masing.

## 1.2 Instalasi

Beberapa kebutuhan aplikasi dan tools diatas ada yang perlu diinstall dan ada yang tersedia secara online. Aplikasi atau tools yang perlu diinstall antara lain: Git, Google Chrome, Docker, Composer dan PhpStorm.

- a) Install JetBrains Toolbox dan PhpStorm
  - | Download JetBrains Toolbox dari [link ini](#).
  - | Melalui JetBrains Toolbox silahkan install PhpStorm
- b) Install Git
  - | Download Git Installer melalui link [Git Installer](#)
  - | Pilih sesuai dengan sistem operasi yang digunakan, misla Windows.
  - | Pada bagian Standalone Installer pilih yang 64 bit.
- c) Install Google Chrome
  - | Download Google Chrome Installer melalui link [Google Chrome Installer](#).
  - | Klik tombol Download Chrome.
- d) Install Composer
  - | Download Composer dari [link ini](#).
  - | Kemudian install file composer tersebut seperti biasanya.

## 1.3 Membuat Project Laravel

Untuk membuat project dengan laravel ikuti langkah-langkah berikut ini:

- ☐ Buka aplikasi Git Bash dan pastikan sedang berada didalam direktori Project.

- ❑ Buat project laravel dengan nama **laraweb** melalui terminal Git Bash dengan perintah berikut:

Terminal

```
composer create-project --prefer-dist laravel/laravel laraweb
```



**Note:**

Project laravel yang dibuat berada didalam folder project sejajar dengan laradock.

**Bahan Bacaan:**

- [Laravel Documentation](#)
- [Laradock Documentation](#)
- [Git Handbook](#)
- [GitHub](#)

---

## 2 — MEMBUAT RESTFUL API LOGIN

---

## 2 Membuat Restful API Login

Ikuti langkah-langkah berikut ini:

- a) Install dan konfigurasi JWT (Json Web Token)

- Instalasi JWT dengan perintah:

Terminal

```
composer require tymon/jwt-auth:2.1.1
```

- Publish konfigurasi file dengan perintah:

Terminal

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

- Generate Secret Key dengan perintah:

Terminal

```
php artisan jwt:secret
```

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

[IDE]-auth.php

```
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
        'hash' => false,
    ],
],
```

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

```
[IDE]-User.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements JWTSubject
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }

    public function getJWTIdentifier()
    {
        return $this->getKey();
    }

    public function getJWTCustomClaims(): array
    {
        return [];
    }
}
```

- ❑ Membuat controller API login dengan perintah:

```
[TML]-create-AuthController.php
```

```
php artisan make:controller Api/AuthController
```

Kemudian sesuaikan kode seperti berikut ini pada file app/Http/Controllers/Api.

```
[IDE]-AuthController.php
```

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Tymon\JWTAuth\Facades\JWTAuth;

class AuthController extends Controller
{
    public function login(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => 'required|email',
            'password' => 'required',
        ]);

        if($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

        $credentials = $request->only('email', 'password');

        if(!$token = auth()->guard('api')->attempt($credentials)) {
            return response()->json([
                'success' => false,
                'message' => 'Invalid Email or Password',
            ], 401);
        }

        return response()->json([
            'success' => true,
            'token' => $token,
            'user' => auth()->guard('api')->user(),
        ], 200);
    }

    public function refresh_token(Request $request)
    {
        $refreshToken = JWTAuth::refresh(JWTAuth::getToken());

        $user = JWTAuth::setToken($refreshToken)->toUser();

        $request->headers->set('Authorization', 'Bearer ' . $refreshToken);

        return response()->json([
            'success' => true,
            'token' => $refreshToken,
            'user' => $user,
        ], 200);
    }

    public function logout()
    {
        JWTAuth::invalidate(JWTAuth::getToken());

        return response()->json([
            'success' => true,
            'message' => 'User logged out successfully',
        ], 200);
    }
}
```

- ❑ Membuat route dengan perintah:

```
[TML]-create-route-api.php
```

```
php artisan install:api
```

Kemudian edit file routes/api.php dengan kode berikut ini.

```
[IDE]-api.php
```

```
<?php

use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\MahasiswaController;
use Illuminate\Support\Facades\Route;

Route::post('/login', [AuthController::class, 'login']);

Route::group(['middleware' => 'auth:api'], function () {
    Route::get('/refresh-token', [AuthController::class, 'refresh_token']);
    Route::post('/logout', [AuthController::class, 'logout']);
});
```

- ❑ Test dengan Postman.

## ■ Bahan Bacaan:

- [Tutorial Laravel 12 JWT #7 Test API Register](#)

---

## 3 — MEMBUAT RESTFUL API CRUD MAHASISWA

---

### 3 Membuat Restful API CRUD Mahasiswa

- a) Membuat model dan migration Program Studi dengan perintah berikut:

```
[TML]-create-model-migration-program-studi
```

```
php artisan make:model ProgramStudi -m
```

- b) Membuat model dan migration Mahasiswa dengan perintah berikut:

```
[TML]-create-model-migration-mahasiswa
```

```
php artisan make:model Mahasiswa -m
```

- c) Membuat resources Mahasiswa dapat dilakukan dengan perintah:

```
[TML]-create-resource-mahasiswa
```

```
php artisan make:resource MahasiswaResource
```

- d) Membuat controller Mahasiswa dengan perintah berikut:

```
[TML]-create-resource-mahasiswa
```

```
php artisan make:controller MahasiswaController --model=Mahasiswa
```

Kemudian buka file MahasiswaController dan sesuaikan setiap fungsi dengan kode berikut ini.

- ❑ Lengkapi fungsi index dengan kode berikut:

[IDE] MahasiswaController.php

```
public function index(Request $request)
{
    $mahasiswas = Mahasiswa::when($request->q, function ($mahasiswas, $request) {
        $mahasiswas->where('nama', 'like', '%' . $request->q . '%');
    })->latest()->paginate(5);

    return new MahasiswaResource(true, 'List Data Mahasiswa', $mahasiswas);
}
```

□ Lengkapi fungsi store dengan kode berikut

[IDE] MahasiswaController.php

```
public function store(Request $request)
{
    $validator = Validator::make($request->all(), [
        'nim' => 'required|size:13|unique:mahasiswas',
        'nama' => 'required|max:100',
        'program_studi_id' => 'required|exists:program_studis,id',
        'email' => 'required|email|max:100|unique:mahasiswas',
        'nomor_hp' => 'required|max:15|unique:mahasiswas',
        'jenis_kelamin' => 'required|boolean',
        'tempat_lahir' => 'required|max:100',
        'tanggal_lahir' => 'required|date',
        'golongan_darah' => 'nullable|in:A,B,AB,O'
    ]);

    if ($validator->fails()) {
        return response()->json($validator->errors(), 422);
    }

    $mahasiswa = Mahasiswa::create($request->all());

    if ($mahasiswa) {
        return new MahasiswaResource(true, 'Data Mahasiswa Berhasil Ditambahkan!', $mahasiswa);
    } else {
        return new MahasiswaResource(false, 'Data Mahasiswa Gagal Ditambahkan!', null);
    }
}
```

□ Lengkapi fungsi show dengan kode berikut

[IDE] MahasiswaController.php

```
public function show(Mahasiswa $mahasiswa)
{
    if ($mahasiswa) {
        return new MahasiswaResource(true, 'Detail Data Mahasiswa!', $mahasiswa);
    } else {
        return new MahasiswaResource(false, 'Data Mahasiswa Tidak Ditemukan!', null);
    }
}
```

- Lengkapi fungsi update dengan kode berikut

```
[IDE] MahasiswaController.php

public function update(Request $request, Mahasiswa $mahasiswa)
{
    $validator = Validator::make($request->all(), [
        'nama' => 'required|max:100',
        'nim' => 'required|size:13|unique:mahasiswas,nim,' . $mahasiswa->id,
    ]);

    if ($validator->fails()) {
        return response()->json($validator->errors(), 422);
    }

    $mahasiswa->update($request->all());

    if ($mahasiswa) {
        return new MahasiswaResource(true, 'Data Mahasiswa Berhasil Diubah!', $mahasiswa);
    } else {
        return new MahasiswaResource(false, 'Data Mahasiswa Gagal Diubah!', null);
    }
}
```

- Lengkapi fungsi destroy dengan kode berikut

```
[IDE] MahasiswaController.php

public function destroy(Mahasiswa $mahasiswa)
{
    $mahasiswa->delete();

    if ($mahasiswa) {
        return new MahasiswaResource(true, 'Data Mahasiswa Berhasil Dihapus!', null);
    } else {
        return new MahasiswaResource(false, 'Data Mahasiswa Gagal Dihapus!', null);
    }
}
```

- e) Buka file MahasiswaResouce.php yang berada pada direktori app/Http/Resources dan tambahkan kode seperti pada gambar berikut.

[IDE] MahasiswaResource.php

```
<?php

namespace App\Http\Resources;

use Illuminate\Http\Request;
use Illuminate\Http\Resources\Json\JsonResource;

class MahasiswaResource extends JsonResource
{
    public $status;
    public $message;

    public function __construct($status, $message, $resource)
    {
        parent::__construct($resource);
        $this->status = $status;
        $this->message = $message;
    }

    public function toArray(Request $request): array
    {
        return [
            'status' => $this->status,
            'message' => $this->message,
            'data' => $this->resource
        ];
    }
}
```

- f) Buka file migration dari Program Studi di dalam folder database/migrations dan tambahkan kode berikut ini.

[IDE] program\_studis\_table.php

```
Schema::create('program_studis', function (Blueprint $table) {
    $table->id();
    $table->string('program_studi', 100);
    $table->timestamps();
});
```

- g) Buka file migration dari Mahasiswa didalam folder yang sama dengan Progam Studi dan tambahkan kode berikut ini.

[IDE] mahasiswas\_table.php

```
Schema::create('mahasiswas', function (Blueprint $table) {
    $table->id();
    $table->char('nim', 13)->unique();
    $table->string('nama', 100);
    $table->foreignId('program_studi_id')->constrained('program_studis');
    $table->string('email', 100)->unique();
    $table->string('nomor_hp', 15)->unique();
    $table->boolean('jenis_kelamin')->default(true); // 1 = Laki-laki, 0 = Perempuan
    $table->string('tempat_lahir', 100);
    $table->date('tanggal_lahir');
    $table->char('golongan_darah', 2)->nullable();
    $table->timestamps();
});
```

- h) Tambahkan route mahasiswa pada file routes/api.php sehingga menjadi seperti berikut.

[IDE]-api.php

```
use App\Http\Controllers\Api\MahasiswaController;

Route::group(['middleware' => 'auth:api'], function () {
    Route::get('/refresh-token', [AuthController::class, 'refresh_token']);
    Route::post('/logout', [AuthController::class, 'logout']);

    Route::apiResource('/mahasiswa', MahasiswaController::class, ['except' => ['create',
    'edit']]);
});
```

- i) Test dengan Postman.

## Bahan Bacaan:

- [JWT \(JSON Web Token\)](#)

## 4 Tugas

Buatlah sebuah Restful API CRUD untuk Dosen dengan data yang disimpan seperti pada gambar berikut ini.

```
[IDE]-dosens-table.php

public function up(): void
{
    Schema::create('dosens', function (Blueprint $table) {
        $table->id();
        $table->string('nidn', 10)->unique()->nullable();
        $table->string('nip', 18)->unique()->nullable();
        $table->date('tmt')->nullable();
        $table->string('nama_lengkap');
        $table->foreignId('pangkat_id')->nullable()->constrained('pangkats');
        $table->boolean('jenis_kelamin')->default(1);
        $table->string('foto')->default('https://www.gravatar.com/avatar/?d=identicon');
        $table->foreignId('kelompok_bidang_keahlian_id')->nullable()-
>constrained('kelompok_bidang_keahlians');
        $table->text('bidang_keilmuan')->nullable();
        $table->enum('jabatan_fungsional', ['tenaga pengajar', 'asisten ahli', 'lektor',
'lektor kepala', 'guru besar'])->default('tenaga pengajar');
        $table->enum('status', ['aktif', 'cuti', 'ijin belajar', 'tugas di instansi lain',
'tugas belajar'])->default('aktif');
        $table->foreignId('user_id')->nullable()->constrained('users')-
>onUpdate('cascade')->onDelete('cascade');
        $table->foreignId('program_studi_id')->nullable()->constrained('program_studis')-
>onUpdate('cascade')->onDelete('cascade');
        $table->timestamps();
        $table->softDeletes();
    });
}
```

Ketentuan mengumpulkan tugas:

- 📌 Tugas dikumpulkan melalui repository GitHub dengan pesan komit "Restful API Dosen"
- 📌 Tugas yang dikumpulkan telah diuji dengan postman.

## 5 Kriteria Penilaian

Kriteria	Poin
Responsi	15%
Laporan	15%
Hasil Kerja	30%
Seminar	40%
<b>Total</b>	<b>100%</b>