Laravel API

Tugas Pendahuluan

1. Jelaskan dengan kata-kata kamu sendiri apa yang dimaksud dengan REST API dan sebutkan komponen utamanya dalam konteks Laravel.
2. Jelaskan apa yang dimaksud dengan JSON, serta berikan contoh format file json berdasarkan data table mahasiswa.
3. Tuliskan dan jelaskan fungsi dari file routes/api.php pada proyek Laravel. Apa perbedaan utamanya dengan file routes/web.php?
4. Sebutkan dan jelaskan empat method HTTP utama yang digunakan dalam REST API Laravel.
5. Apa perbedaan antara response biasa dan response JSON di Laravel?
   Sertakan contoh kode singkat untuk menampilkan data mahasiswa dalam format JSON.

REST API

a) REST (Representational State Transfer) adalah standar arsitektur untuk membangun API agar:
   - Data dikirim lewat HTTP request (GET, POST, PUT, DELETE)
   - Hasilnya dikirim dalam bentuk JSON (bukan HTML)
   - Tidak bergantung pada session atau tampilan view (murni data)

b) API(Application Programming Interface) adalah jembatan yang memungkinkan satu aplikasi berkomunikasi dengan aplikasi lain.

c) REST API : API yang mengikuti prinsip REST, yaitu pertukaran data menggunakan format tertentu (biasanya JSON) lewat HTTP request.

d) Contoh: misalnya kita punya aplikasi mobile yang menampilkan daftar mahasiswa dari server Laravel.
   - Android kirim request:

```
GET /api/mahasiswa
```

   - Laravel merespon:

```
[
  {
    “id”: 1,
    “nama”: “Budi Santoso”,
    “nim”: “12345678”
  },
  {
    “id”: 2,
    “nama”: “Siti Rahma”,
    “nim”: “87654321”
  }
```

```
]
```

- Android menampilkan daftar mahasiswa tersebut di aplikasi.

Jurnal (REST API di Laravel)

Untuk percobaan ini kita akan menggunakan projek yang sudah ada yang biasa digunakan untuk Latihan. Kita akan gunakan table mahasiswa yang biasa digunakan pada praktikum sebelumnya.

Step 1 – Buat Controller API

- Buat controller dengan flag `--api` agar tidak membuat method create/edit (karena itu untuk web view):

```
php artisan make:controller Api/MahasiswaController --api
```

- Controller akan dibuat di:
  app/Http/Controllers/Api/MahasiswaController.php

```php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class MahasiswaController extends Controller
{
    public function index()
    {
        //
    }

    public function store(Request $request)
    {
        //
    }

    public function show(string $id)
    {
        //
    }

    public function update(Request $request, string $id)
    {
        //
    }

    public function destroy(string $id)
    {
        //
    }
}
```

Step 2 – Isi Controller dan Model

- Controller API Mahasiswa

```php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Mahasiswa;
use Illuminate\Http\Request;

class MahasiswaController extends Controller
{
    // GET: /api/mahasiswa
    public function index()
    {
        $data = Mahasiswa::all();
        return response()->json($data);
    }

    // GET: /api/mahasiswa/{id}
    public function show($id)
    {
        $mhs = Mahasiswa::find($id);
        if (!$mhs) {
          return response()->json(['message' => 'Data tdk ditemukan'], 404);
        }
        return response()->json($mhs);
    }

    // POST: /api/mahasiswa
    public function store(Request $request)
    {
        $mhs = Mahasiswa::create($request->all());
        return response()->json(['message' => 'Data ditambahkan', 'data' => $mhs],
201);
    }

    // PUT: /api/mahasiswa/{id}
    public function update(Request $request, $id)
    {
        $mhs = Mahasiswa::find($id);
        if (!$mhs) {
          return response()->json(['message' => 'Data tdk ditemukan'], 404);
        }

        $mhs->update($request->all());
        return response()->json(['message' => 'Data diperbarui', 'data' => $mhs]);
    }

    // DELETE: /api/mahasiswa/{id}
    public function destroy($id)
    {
        $mhs = Mahasiswa::find($id);
        if (!$mhs) {
          return response()->json(['message' => 'Data tdk ditemukan'], 404);
        }
```

```
        $mhs->delete();
        return response()->json(['message' => 'Data berhasil dihapus']);
    }
}
```

Ket:

- `response()` merupakan helper global bawaan dari laravel
- `response()->json()` : mengirim data ke client dalam format JSON. Jadi Laravel akan mengubah variable yang berisi array atau object menjadi format JSON.

– Isi Model Mahasiswa:

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Mahasiswa extends Model
{
    protected $table = 'mahasiswa';
    protected $fillable = ['nim', 'nama', 'email', 'prodi'];
    protected $primaryKey = 'nim';
    protected $keyType = 'string';
}
```

Step 3 – Tambahkan Route API

– Buka file **`routes/api.php`**, tambahkan: (Jika belum ada buat manual)

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

Route::apiResource('mahasiswa', MahasiswaController::class);

//atau secara manual satu persatu di buat seperti di bawah ini:

//Route::get('/mahasiswa', [MahasiswaController::class, 'index']);
//Route::get('/mahasiswa/{id}', [MahasiswaController::class, 'show']);
//Route::post('/mahasiswa', [MahasiswaController::class, 'store']);
//Route::put('/mahasiswa/{id}', [MahasiswaController::class, 'update']);
//Route::delete('/mahasiswa/{id}', [MahasiswaController::class,
'destroy']);
```

– Perintah `apiResource` otomatis membuat route CRUD lengkap:

Method	Endpoint	Fungsi	Keterangan
GET	/api/mahasiswa	index	Tampilkan semua data
GET	/api/mahasiswa/{id}	show	Tampilkan 1 data
POST	/api/mahasiswa	store	Simpan data baru
PUT	/api/mahasiswa/{id}	update	Ubah data
DELETE	/api/mahasiswa/{id}	destroy	Hapus data

- Kalau `api.php` sudah aktif, maka semua route di dalamnya otomatis punya prefix `api/`. Misalnya kamu menulis:

```
Route::get('/mahasiswa', [MahasiswaController::class, 'index']);
```

  Maka URL untuk mengaksesnya adalah: `http://127.0.0.1:8000/api/mahasiswa`
  Bukan `http://127.0.0.1:8000/mahasiswa` (karena itu milik `web.php`).

- Perintah untuk memeriksa, route apa saja yang aktif:

```
php artisan route:list
```

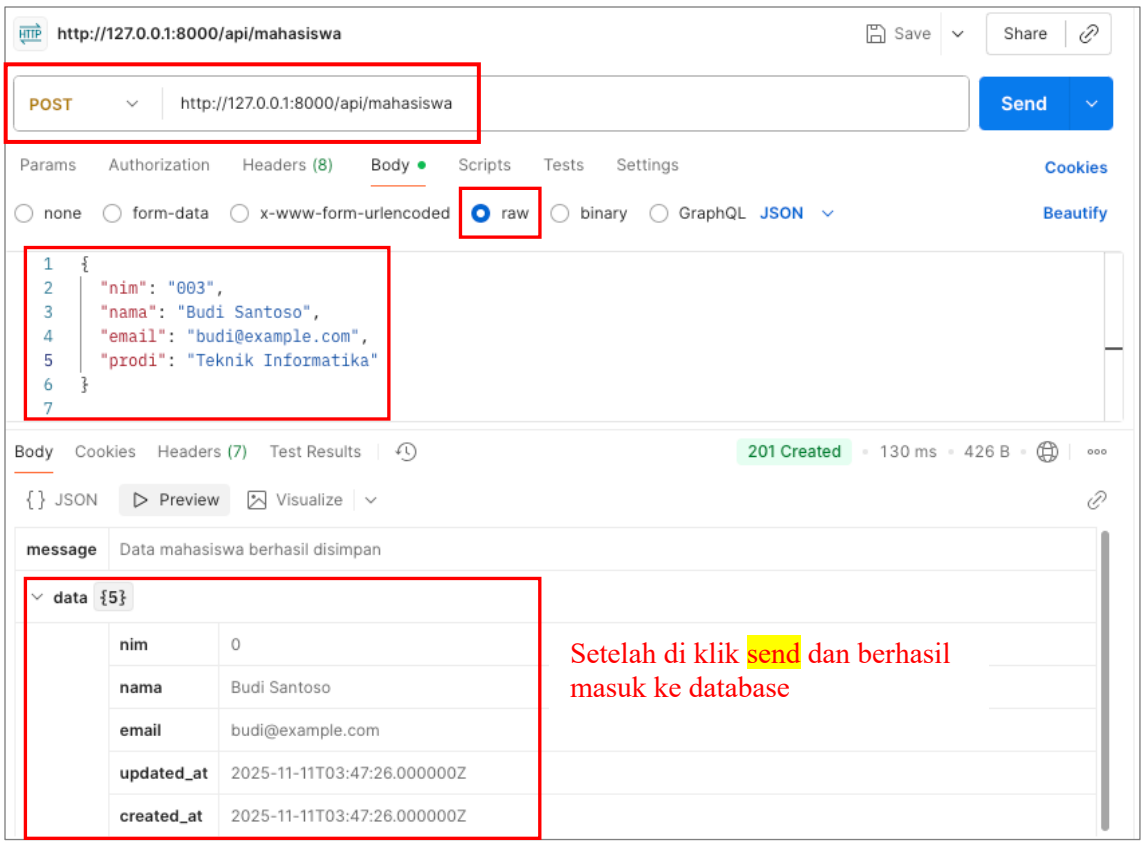
- Troubleshoot: Misal Ketika url API tidak ada di list dan ketika di akses melalui browser outputnya not found maka edit file `bootstrap/app.php` menjadi:

```
->withRouting(
    web: __DIR__.'/../routes/web.php',
    api: __DIR__.'/../routes/api.php', // pastikan baris ini ada
    commands: __DIR__.'/../routes/console.php',
)
```

Step 4 – Tes API Menggunakan Browser atau Postman

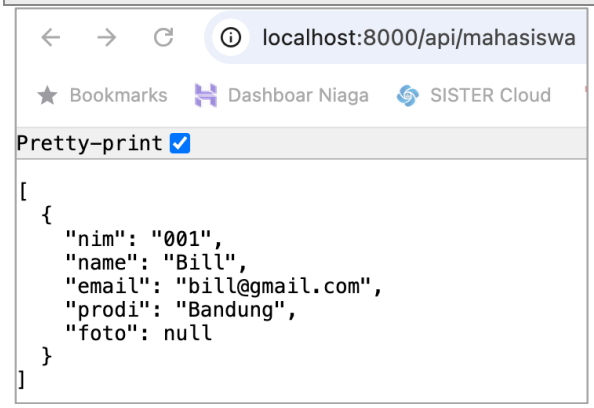
- Create / Insert (POST)
  Gunakan aplikasi postman dan masuan parameter berikut:
  URL : `http://127.0.0.1:8000/api/mahasiswa`
  Method : POST
  Body (form-data / JSON):

```
{
  "nim": "123456",
  "nama": "Budi Santoso",
  "email": "budi@example.com",
  "prodi": "Teknik Informatika"
}
```



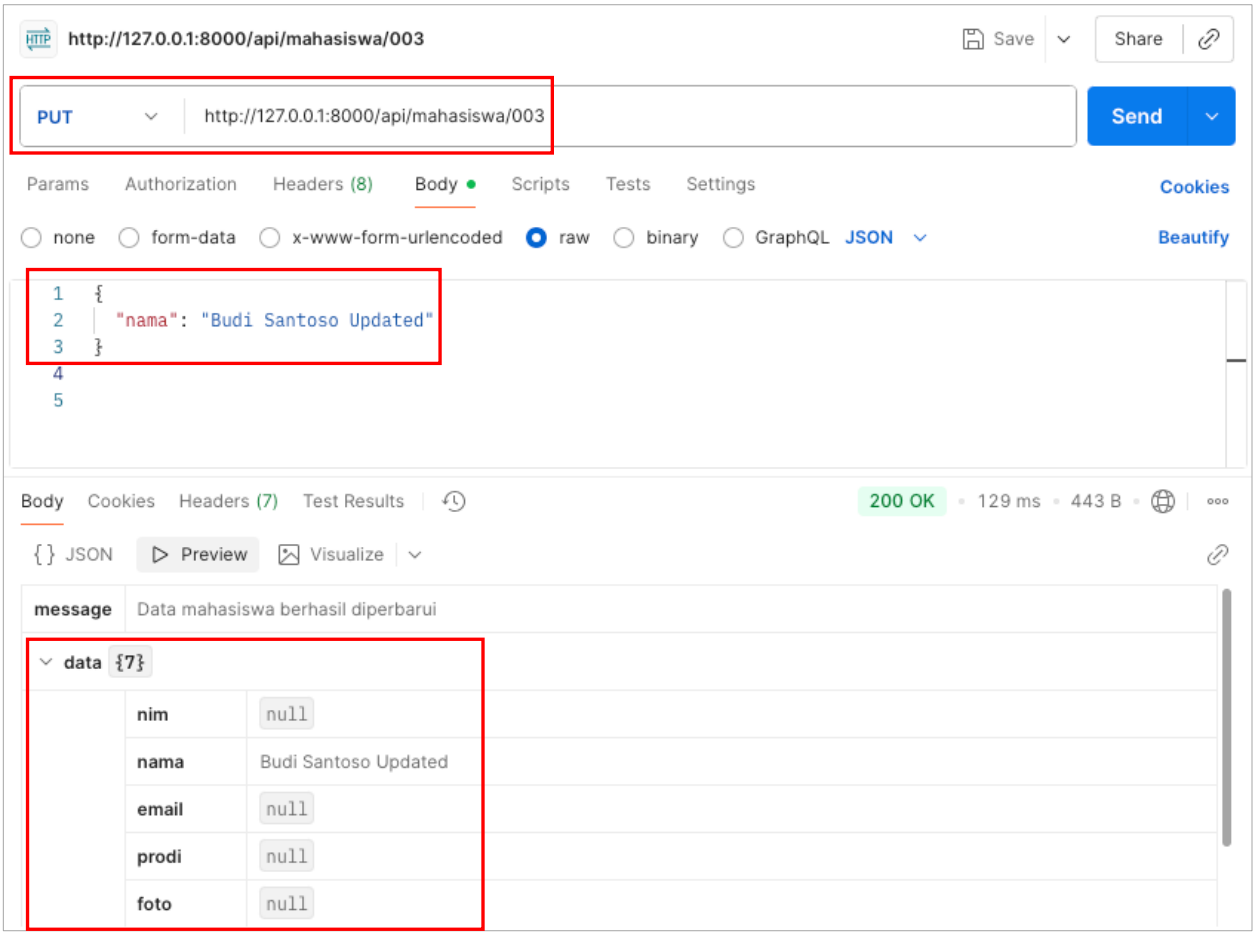
- Read (GET)
  Buka di browser atau postman menggunakan method get

```
http://127.0.0.1:8000/api/mahasiswa
```



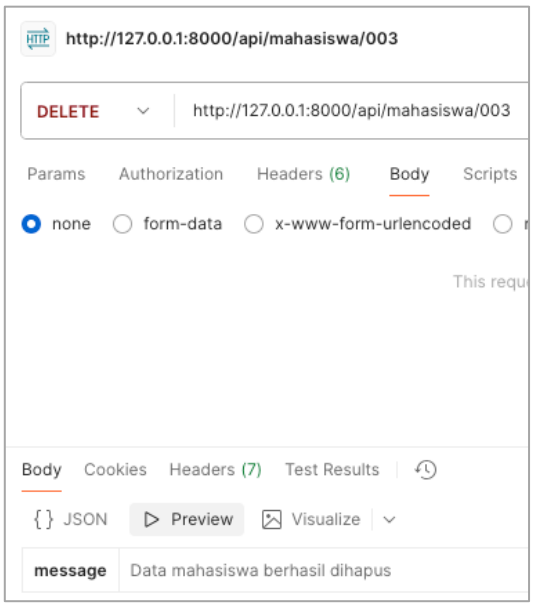
- Update/Edit (PUT)
  Gunakan aplikasi postman dan masuan parameter berikut:
  URL : `http://127.0.0.1:8000/api/mahasiswa/1`
  Method : PUT
  Body :

```
{
  "nama": "Budi Santoso Updated"
}
```



– Delete (DELETE)
Buka di postman menggunakan method get

```
http://127.0.0.1:8000/api/mahasiswa/1
```



Step 5 – Menampilkan data dari API di Laravel view(simulasi)

- Pertama-tama kita ganti port server untuk menjalakan API dari 8000 menjadi 8001, agar server yang menjalankan API bisa di akses dari controller.

```
php artisan serve --port=8001
```

Jika port projek Laravel kita sama dengan port Laravel API, maka akan terjadi error akibat tidak bisa mengakses data APInya. Jadi nantinya aka nada 2 server artisan yang berjalan:

  - API → `http://127.0.0.1:8001/api/mahasiswa`
  - View → `http://127.0.0.1:8000/mahasiswa`

- Pastikan API sudah bisa diakses dengan cara buka di browser / Postman:

```
GET http://127.0.0.1:8001/api/mahasiswa
```

Harus muncul data JSON mahasiswa.

- Gunakan `Http` Facade untuk mengambil data

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http; // <- penting

class MahasiswaViewController extends Controller
{
    public function index()
    {
        // Panggil API menggunakan Http::get()
        $a = Http::get('http://127.0.0.1:8001/api/mahasiswa');
```

```
        // mengubah isi respon API menjadi array
        $mahasiswa = $a->json();

        // Kirim ke view
        return view('mahasiswa-api, compact('mahasiswa'));
    }
}
```

- Tambahkan route web (Edit `routes/web.php`)

```
Route::get('/mahasiswa-view', [MahasiswaViewController::class, 'index']);
```

- Buat view `mahasiswa-api.blade.php`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Data Mahasiswa (Dari API)</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">
</head>
<body class="p-4">
    <div class="container">
        <h3 class="mb-4 text-center">Daftar Mahasiswa</h3>

        <table class="table table-bordered table-striped">
            <thead>
                <tr class="table-dark text-center">
                    <th>ID</th>
                    <th>NIM</th>
                    <th>Nama</th>
                    <th>Email</th>
                    <th>Prodi</th>
                    <th>Foto</th>
                </tr>
            </thead>
            <tbody>
                @forelse ($mahasiswa as $m)
                    <tr>
                        <td>{{ $m['id'] }}</td>
                        <td>{{ $m['nim'] }}</td>
                        <td>{{ $m['nama'] }}</td>
                        <td>{{ $m['email'] }}</td>
                        <td>{{ $m['prodi'] }}</td>
                        <td>
                            @if(isset($m['foto']))
                                <img src="{{ asset($m['foto']) }}">
                            @else
                                <em>Tidak ada</em>
                            @endif
                        </td>
                    </tr>
```

```
                @empty
                    <tr>
                        <td colspan="6">Tidak ada data</td>
                    </tr>
                @endforelse
            </tbody>
        </table>
    </div>
</body>
</html>
```

- Coba di browser
  Akses: `http://127.0.0.1:8000/mahasiswa-view`
  Maka data dari `api/mahasiswa` akan muncul dalam bentuk tabel di view Laravel.

Latihan

1. Tambahkan fitur untuk mengupload foto mahasiswa melalui API. Foto harus tersimpan di folder `public/uploads/mahasiswa/`.
   - Tambahkan route di `routes/api.php`:

```
Route::post('/mahasiswa/{id}/upload-foto', [MahasiswaController::class, 'uploadFoto']);
```

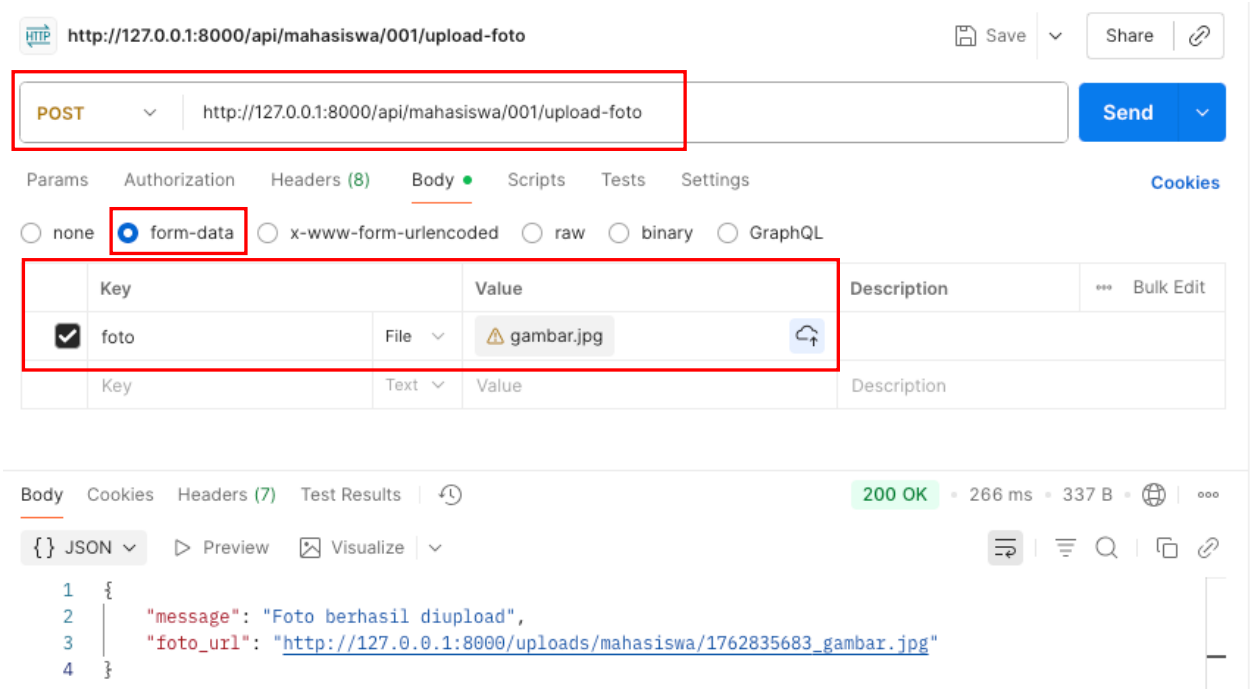
   - Tambahkan method di `MahasiswaController`:

```
public function uploadFoto(Request $request, $id)
{
    $mahasiswa = Mahasiswa::findOrFail($id);

    if ($request->hasFile('foto')) {
        $file = $request->file('foto');
        $filename = time().'_'.$file->getClientOriginalName();
        $file->move(public_path('uploads/mahasiswa'), $filename);

        $mahasiswa->foto = 'uploads/mahasiswa/'.$filename;
        $mahasiswa->save();

        return response()->json([
            'message' => 'Foto berhasil diupload',
            'foto_url' => asset($mahasiswa->foto),
        ], 200);
    }

    return response()->json(['message' => 'Tidak ada file foto diunggah'], 400);
}
```

   - Uji di Postman:
     Method : `POST`
     URL : `http://127.0.0.1:8000/api/mahasiswa/1/upload-foto`

Body → form-data :

- Key : `foto`,
- Type : `File`,
- Value : pilih file gambar yang ada di laptop



2. Buat feature untuk mencari mahasiswa berdasarkan **nama** atau **NIM**.
   - Tambahkan route untuk pencarian

```
Route::get('/mahasiswa/search/{keyword}', [MahasiswaController::class, 'search']);
```

   - Tambahkan method untuk pencarian

```
public function search($keyword)
{
    $result = Mahasiswa::where('nama', 'like', "%{$keyword}%")
                ->orWhere('nim', 'like', "%{$keyword}%")
                ->get();

    if ($result->isEmpty()) {
        return response()->json(['message' => 'Data tidak ditemukan'], 404);
    }

    return response()->json($result);
}
```

   - Coba hasilnya di postman:

```
GET http://127.0.0.1:8000/api/mahasiswa/search/budi
```

3. Buat feature filtering untuk menampilkan mahasiswa berdasarkan prodi tertentu, misal: `Informatika`.
   - Tambahkan route:
     ```
     Route::get('/mahasiswa/filter/prodi/{prodi}', [MahasiswaController::class, 'filterByProdi']);
     ```
   - Tambahkan method:
     ```
     public function filterByProdi($prodi)
     {
         $data = Mahasiswa::where('prodi', $prodi)->get();

         return response()->json([
             'count' => $data->count(),
             'data' => $data
         ]);
     }
     ```
   - Coba hasilnya di postman:
     ```
     GET http://127.0.0.1:8000/api/mahasiswa/filter/prodi/Informatika
     ```

4. Buat feature untuk pagination tampilan data mahasiswa dengan sistem pagination (5 data per halaman).
   - Tambahkan route:
     ```
     Route::get('/mahasiswa/page', [MahasiswaController::class, 'paginate']);
     ```
   - Tambahkan method:
     ```
     public function paginate()
     {
         $data = Mahasiswa::paginate(5);

         return response()->json([
             'current_page' => $data->currentPage(),
             'total' => $data->total(),
             'per_page' => $data->perPage(),
             'data' => $data->items(),
         ]);
     }
     ```
   - Coba hasilnya di postman:
     ```
     GET http://127.0.0.1:8000/api/mahasiswa/page?page=2
     ```

5. Buat feature validation saat menambah mahasiswa(insert), pastikan kolom `nim`, `nama`, `email`, dan `prodi` wajib diisi.
   ```
   public function store(Request $request)
   {
       $validated = $request->validate([
   ```

```
        'nim' => 'required|unique:mahasiswas,nim',
        'nama' => 'required|string|max:100',
        'email' => 'required|email',
        'prodi' => 'required|string'
    ]);

    $mahasiswa = Mahasiswa::create($validated);

    return response()->json([
        'message' => 'Mahasiswa berhasil ditambahkan',
        'data' => $mahasiswa
    ], 201);
}
```

- Coba hasilnya di postman:

  Method : `POST`
  URL : `http://127.0.0.1:8000/api/mahasiswa`
  Body → raw → JSON:

```
{
  "nim": "23001",
  "nama": "Ayu Lestari",
  "email": "ayu@pei.ac.id",
  "prodi": "Informatika"
}
```

  Jika ada field kosong:

```
{
  "message": "The nim field is required.",
  "errors": {
    "nim": ["The nim field is required."]
  }
}
```

Tugas

Buat CRUD seperti pada materi model, hanya saja kali ini proses CRUDnya tidak langsung mengakses model tapi menggunakan API. Gunakan table lain selain table mahasiswa.

Contoh CRUD menggunakan API pada table mahasiswa:

Step 1 – Pastikan Controller API dan Route API sudah lengkap seperti pada Latihan di atas.

Method	URL	Keterangan
GET	`/api/mahasiswa`	Mengambil semua data
POST	`/api/mahasiswa`	Menambah data
GET	`/api/mahasiswa/{id}`	Detail data
PUT	`/api/mahasiswa/{id}`	Edit data
DELETE	`/api/mahasiswa/{id}`	Hapus data

Step 2 – Buat route untuk semua proses CRUD pada web.php

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

Route::get('/mahasiswa', [MahasiswaController::class, 'index']);;
Route::get('/mahasiswaInput', [MahasiswaController::class, 'create']);
Route::post('/mahasiswaSimpan', [MahasiswaController::class, 'store']);
Route::get('/mahasiswaEdit/{id}', [MahasiswaController::class, 'edit']);
Route::post('/mahasiswaupdate/{id}', [MahasiswaController::class, 'update']);
Route::get('/mahasiswaHaous/{id}', [MahasiswaController::class, 'destroy']);
```

Step 3 – Buat Controller Mahasiswa yang didalamnya terdapat semua method untuk proses CRUD menggunakan API.

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;//untuk menggunakan http request

class MahasiswaController extends Controller
{
    private $apiUrl = 'http://localhost:8001/api/mahasiswa';

    // Tampilkan semua data
    public function index()
    {
        $a = Http::get($this->apiUrl);
        $mahasiswa = $a->json();

        return view('mahasiswa.index', compact('mahasiswa'));
    }

    // Form tambah data
    public function create()
    {
        return view('mahasiswa.create');
    }

    // Simpan data baru
    public function store(Request $x)
    {
        Http::post($this->apiUrl, [
            'nim' => $x->nim,
```

```
            'nama' => $x->nama,
            'email' => $x->email,
        ]);

        return redirect('/mahasiswa');
    }

    // Form edit
    public function edit($id)
    {
        $a = Http::get("{$this->apiUrl}/{$id}");
        $mhs = $a->json();

        return view('mahasiswa.edit', compact('mhs'));
    }

    // Update data
    public function update(Request $x, $id)
    {
        Http::put("{$this->apiUrl}/{$id}", [
            'nim' => $x->nim,
            'nama' => $x->nama,
            'email' => $x->email,
        ]);

        return redirect('/mahasiswa');
    }

    // Hapus data
    public function destroy($id)
    {
        Http::delete("{$this->apiUrl}/{$id}");
        return redirect('/mahasiswa');
    }
}
```

Step 4 – Buat halaman view untuk semua proses CRUD.

- Halaman table: `resources/views/mahasiswa/index.blade.php`

```
<!DOCTYPE html>
<html>
<head>
    <title>Data Mahasiswa (API)</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="p-5">

    <h2>Data Mahasiswa</h2>
    <a href="/mahasiswaInput" class="btn btn-primary mb-3">Tambah</a>

    <table class="table table-bordered">
        <tr>
            <th>No</th>
            <th>NIM</th>
```

```
            <th>Nama</th>
            <th>Email</th>
            <th>Aksi</th>
        </tr>
        @foreach($mahasiswa as $key => $m)
        <tr>
            <td>{{ $key + 1 }}</td>
            <td>{{ $m['nim'] }}</td>
            <td>{{ $m['nama'] }}</td>
            <td>{{ $m['email'] }}</td>
            <td>
                <a href="/mahasiswaEdit/{{$m['nim']) }}">Edit</a>
                <a href="/mahasiswaHapus/{{$m['nim']) }}">Hapus</a>
            </td>
        </tr>
        @endforeach
    </table>
</body>
</html>
```

- Halaman insert: `resources/views/mahasiswa/create.blade.php`

```
<!DOCTYPE html>
<html>
<head>
    <title>Tambah Mahasiswa (API)</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="p-5">
    <h2>Tambah Mahasiswa</h2>

    <form action="/mahasiswaSimpan" method="POST">
        @csrf
        <div class="mb-3">
            <label>NIM</label>
            <input type="text" name="nim" class="form-control">
        </div>
        <div class="mb-3">
            <label>Nama</label>
            <input type="text" name="nama" class="form-control">
        </div>
        <div class="mb-3">
            <label>Email</label>
            <input type="email" name="email" class="form-control">
        </div>
        <button class="btn btn-primary">Simpan</button>
    </form>
</body>
</html>
```

- Halaman Edit: `resources/views/mahasiswa/edit.blade.php`

```
<!DOCTYPE html>
<html>
<head>
    <title>Edit Mahasiswa (API)</title>
```

```
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="p-5">
    <h2>Edit Mahasiswa</h2>

    <form action="/mahasiswaUpdate/{{$mhs['nim']}}" method="POST">
        @csrf
        <div class="mb-3">
            <label>NIM</label>
            <input type="text" name="nim" class="form-control" value="{{ $mhs['nim']
}}">
        </div>
        <div class="mb-3">
            <label>Nama</label>
            <input type="text" name="nama" class="form-control" value="{{ $mhs['nama']
}}">
        </div>
        <div class="mb-3">
            <label>Email</label>
            <input type="email" name="email" class="form-control" value="{{
$mhs['email'] }}">
        </div>
        <button class="btn btn-primary">Update</button>
        <a href="{{ route('mahasiswa.index') }}" class="btn btn-secondary">Kembali</a>
    </form>
</body>
</html>
```