PRAKTIKUM PEMROGRAMAN WEB

Laporan Praktikum Jobsheet 4

Dosen Pengampu: Dr. Ika Parma Dewi, M.Pd.T.

Oleh :

Rian Septiawan 23076052

PRODI PENDIDIKAN TEKNIK INFORMATIKA

DEPARTEMEN ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

FT UNP Padang	**Lembaran : JOBSHEET 4**
Jurusan : Pendidikan Teknik Informatika	**Matakuliah: Pratikum Pemrograman Web**
Waktu : 3 x 50 Menit	**Topik : Halaman Data Dokter**
Kode : TIK1.61.4318	**Judul : Halaman Data Dokter**

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.4318	2 x 170 Menit

A. TUJUAN PRAKTIKUM

Setelah mengikuti perkuliahan ini diharapkan mahasiswa mampu :

1. Dapat menampilkan tabel dari database
2. Dapat mengelola database
3. Dapat membuat projek dengan Laravel.

B. ALAT DAN BAHAN

1. PC/Laptop
2. Browser Internet (Internet Explorer/Mozilla Firefox/Google Chrome)
3. XAMPP/Laragon
4. Text Editor (Visual Studio Code)

C. TEORI SINGKAT

Fungsi Controller dan Model di Laravel

1. Controller

Controller di Laravel berfungsi sebagai penghubung antara model (database) dan view (tampilan). Controller menangani logika aplikasi dan permintaan dari pengguna.

Fungsi utama Controller:

1) Mengelola request dan response.
2) Memproses logika bisnis sebelum data dikirim ke view.
3) Menggunakan model untuk mengambil, menyimpan, atau memperbarui data di database.

Contoh Controller:

```
php
CopyEdit
class PendaftaranController extends Controller
{
    public function index()
    {
        $pendaftaran = Pendaftaran::all(); // Mengambil semua data
pendaftaran
        return view('pendaftaran.index', compact('pendaftaran'));
    }
}
```

2. Model

Model di Laravel berfungsi untuk berinteraksi dengan database menggunakan ORM (Eloquent). Model merepresentasikan tabel di database dan memungkinkan manipulasi data tanpa menulis query SQL langsung.

Fungsi utama Model:

a. Representasi tabel database dalam bentuk class PHP.
b. Menangani operasi CRUD (Create, Read, Update, Delete).
c. Menggunakan Eloquent ORM untuk query database dengan cara yang lebih sederhana.

Contoh Model:

```
php
CopyEdit
class Pendaftaran extends Model
{
    protected $table = 'pendaftaran'; // Nama tabel di database
    protected $fillable = ['nama_pasien', 'penjamin',
'jadwalpoliklinik_id']; // Kolom yang bisa diisi
}
```

Kesimpulan:

1) **Controller** → Menangani request dan logika aplikasi.
2) **Model** → Berinteraksi dengan database menggunakan Eloquent ORM.

D. LANGKAH KERJA

4.1. Membuat Tabel Dokter

1. Buat tabel dengan nama “dokter” pada database, dengan cara menjalankan **php artisan make:model nama_tabel -m** pada terminal, kemudian buat seperti ini

```
PS C:\Users\ryans\Documents\PROJECT DEVELOP\A Github> cd "C:\laragon\www\rumah-sakit"
PS C:\laragon\www\rumah-sakit> php artisan make:model dokter -m
<!-- web.php -->

  INFO  Model [C:\laragon\www\rumah-sakit\app/Models/dokter.php] created successfully.

  INFO  Migration [C:\laragon\www\rumah-sakit\database\migrations/2025_02_25_180627_create_dokters_table.php] created successfully.
```

```php
public function up()
{
    Schema::create('dokter', function (Blueprint $table) {
        $table->id();
        $table->string('nama_dokter');
        $table->foreignId('poliklinik_id')->constrained('poliklinik')->onDelete('cascade');
        $table->string('foto_dokter')->nullable();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('dokter');
}
```

2. Kemudian buka **Dokter.php** pada folder **Models,** dan tambahkan source code dibawah ini

app > Models > dokter.php > PHP Intelephense > dokter

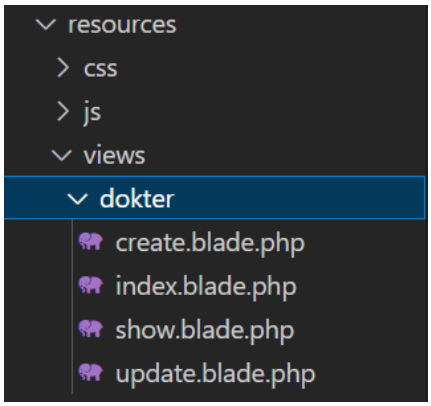
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class dokter extends Model
{
    use HasFactory;

    // Beritahu Laravel untuk menggunakan tabel 'dokter'
    protected $table = 'dokter';

    protected $fillable = ['nama_dokter', 'poliklinik_id', 'foto_dokter'];

    public function poliklinik()
    {
        return $this->belongsTo(Poliklinik::class, 'poliklinik_id');
    }
}
```

3. Setelah itu, jalankan **php artisan migrate** agar masuk kedalam database

```
PS C:\laragon\www\rumah-sakit> php artisan migrate
<!-- web.php -->

  INFO  Running migrations.

  2025_02_25_180627_create_dokters_table ..... 138ms DONE

PS C:\laragon\www\rumah-sakit>
```

4.2. Buat Halaman Data Dokter

1. Buat folder **dokter** pada views dengan 4 file seperti dibawah ini



2. Isi halaman **index.blade.php**

```
<!-- indext.blade.php -->
@extends('layout.admin')

@section('title', 'dokter')

@section('content')
<!-- Page Heading -->
<h1 class="h3 mb-2 text-gray-800">Data Dokter</h1>

<!-- DataTales Example -->
<div class="card shadow mb-4">
    <div class="card-header py-3">
        <a href="{{ route('dokter.create') }}" class="btn btn-primary btn-
sm"><i class="fas fa-plus"></i> Tambah</a>
    </div>
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-bordered" id="dataTable" width="100%"
cellspacing="0">
                <thead>
                    <tr>
                        <th>No</th>
                        <th>Nama Dokter</th>
                        <th>Poli</th>
                        <th>Profil</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
```

```
                    @php $no = 1; @endphp
                    @foreach ($dokter as $item)
                    <tr>
                        <td>{{ $no++ }}</td>
                        <td>{{ $item->nama_dokter }}</td>
                        <td>{{ $item->poliklinik->nama_poliklinik }}</td>
                        <td>
                            <img src="{{ asset('storage/foto_dokter/' .
$item->foto_dokter) }}" alt="Foto Dokter" width="50" height="50">
                        </td>
                        <td>
                            <a href="{{ route('dokter.show', $item->id) }}"
class="btn btn-info btn-sm"><i class="fas fa-eye"></i></a>
                            <a href="{{ route('dokter.edit', $item->id) }}"
class="btn btn-warning btn-sm"><i class="fas fa-edit"></i></a>
                            <form action="{{ route('dokter.destroy', $item-
>id) }}" method="POST" class="d-inline delete-form">
                                @csrf
                                @method('DELETE')
                                <button type="submit" class="btn btn-danger
btn-sm btn-delete">Hapus</button>
                            </form>
                        </td>
                    </tr>
                    @endforeach
                    @include('sweetalert::alert')
                </tbody>
            </table>
        </div>
    </div>
</div>

<!-- Sertakan SweetAlert2 -->
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@10"></script>
<!-- Sertakan file JavaScript khusus -->
<script src="{{ asset('js/sweetalert.js') }}"></script>
@endsection
```

3. Isi halaman **create.blade.php**

```
@extends('layout.admin')
@section('title', 'Tambah Dokter')
@section('content')
<div class="card shadow mb-4">
    <div class="card-header py-3">
        <h6 class="m-0 font-weight-bold text-primary">Tambah Dokter
Baru</h6>
    </div>
    <div class="card-body">
        <form action="{{ route('dokter.store') }}" method="POST"
enctype="multipart/form-data">
            @csrf
            <div class="form-group">
                <label for="nama_dokter">Nama Dokter</label>
                <input type="text" name="nama_dokter" id="nama_dokter"
class="form-control" required>
            </div>
            <div class="form-group">
                <label for="poliklinik_id">Nama Poliklinik</label>
                <select name="poliklinik_id" id="poliklinik_id"
class="form-control" required>
                    @foreach($poliklinik as $item)
                        <option value="{{ $item->id }}">{{ $item-
>nama_poliklinik }}</option>
                    @endforeach
                </select>
            </div>
            <div class="form-group">
                <label for="foto_dokter" class="form-label">Foto
Profil</label>
                <input class="form-control" type="file" name="foto_dokter"
id="foto_dokter">
            </div>
            <div class="text-center">
                <button type="submit" class="btn btn-
primary">Simpan</button>
                <a href="{{ route('dokter.index') }}" class="btn btn-
secondary">Batal</a>
            </div>
        </form>
    </div>
</div>
@endsection
```

4. Isi halaman **update.blade.php**

```
<!-- update.blade.php -->
@extends('layout.admin')

@section('title', 'Edit Dokter')

@section('content')
<div class="card shadow mb-4">
    <div class="card-header py-3">
        <h6 class="m-0 font-weight-bold text-primary">Edit Dokter</h6>
    </div>
    <div class="card-body">
        <form action="{{ route('dokter.update', $dokter->id) }}" method="POST" enctype="multipart/form-data">
            @csrf
            @method('PUT')
            <div class="form-group">
                <label for="nama_dokter">Nama Dokter</label>
                <input type="text" name="nama_dokter" id="nama_dokter" class="form-control" required value="{{ $dokter->nama_dokter }}">
            </div>
            <div class="form-group">
                <label for="poliklinik_id">Nama Poliklinik</label>
                <select name="poliklinik_id" id="poliklinik_id" class="form-control" required>
                    @foreach($poliklinik as $item)
                        <option value="{{ $item->id }}" {{ $dokter->poliklinik_id == $item->id ? 'selected' : '' }}>
                            {{ $item->nama_poliklinik }}
                        </option>
                    @endforeach
                </select>
            </div>
            <div class="form-group">
                <label for="foto_dokter" class="form-label">Foto Profil</label>
                <input class="form-control" type="file" name="foto_dokter" id="foto_dokter">
            </div>
            <div class="text-center">
                <button type="submit" class="btn btn-primary">Update</button>
                <a href="{{ route('dokter.index') }}" class="btn btn-secondary">Batal</a>
```

```
            </div>
        </form>
    </div>
</div>
@endsection
```

5. Isi halaman **show.blade.php**

```
@extends('layout.admin')

@section('title', 'Detail Dokter')

@section('content')
<!-- Page Heading -->
<h1 class="h3 mb-2 text-gray-800 text-center">Detail Dokter</h1>

<div class="card shadow mb-4">
    <div class="card-header py-3">
        <a href="{{ route('dokter.index') }}" class="btn btn-primary btn-sm"><i class="fas fa-arrow-left"></i> Kembali</a>
    </div>
    <div class="card-body text-center">
        @if($dokter)
            <div class="form-group">
                <img src="{{ asset('storage/foto_dokter/' . $dokter->foto_dokter) }}" alt="Foto Dokter" class="img-fluid rounded mb-4">
            </div>
            <div class="form-group">
                <h5>Nama Dokter:</h5>
                <p>{{ $dokter->nama_dokter }}</p>
            </div>
            <div class="form-group">
                <h5>Poli:</h5>
                <p>{{ $dokter->poliklinik->nama_poliklinik }}</p>
            </div>
        @else
            <p>Dokter tidak ditemukan.</p>
        @endif
    </div>
</div>
@endsection
```

4.3 Controller dan web.php

1. Buat **DokterController** melalui terminal dengan **php artisan make:controller NamaController –resource**

```
PS C:\laragon\www\rumah-sakit> php artisan make:controller DokterController --resource
<!-- web.php -->

  INFO  Controller [C:\laragon\www\rumah-sakit\app/Http/Controllers/DokterController.php] created successfully.

PS C:\laragon\www\rumah-sakit>
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Dokter;
use App\Models\Poliklinik;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Log;

class DokterController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $dokter = Dokter::latest()->get();
        return view('dokter.index', [
            'dokter' => $dokter
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        Log::info('Metode create dipanggil');
        $poliklinik = Poliklinik::all();
        return view('dokter.create', compact('poliklinik'));
    }

```

```
38.     /**
39.      * Store a newly created resource in storage.
40.      *
41.      * @param  \Illuminate\Http\Request  $request
42.      * @return \Illuminate\Http\Response
43.      */
44.     public function store(Request $request)
45.     {
46.         // Validasi data
47.         $validatedData = $request->validate([
48.             'nama_dokter' => 'required|max:255',
49.             'poliklinik_id' => 'required',
50.             'foto_dokter' => 'image|nullable|max:1999'
51.         ]);
52.
53.         // Proses upload file foto
54.         if ($request->hasFile('foto_dokter')) {
55.             $filenameWithExt = $request->file('foto_dokter')-
    >getClientOriginalName();
56.             $filename = pathinfo($filenameWithExt, PATHINFO_FILENAME);
57.             $extension = $request->file('foto_dokter')-
    >getClientOriginalExtension();
58.             $filenameToStore = $filename . '_' . time() . '.' .
    $extension;
59.             $path = $request->file('foto_dokter')-
    >storeAs('public/foto_dokter', $filenameToStore);
60.         } else {
61.             $filenameToStore = 'noimage.jpg';
62.         }
63.
64.         // Simpan data dokter baru
65.         $dokter = new Dokter;
66.         $dokter->nama_dokter = $validatedData['nama_dokter'];
67.         $dokter->poliklinik_id = $validatedData['poliklinik_id'];
68.         $dokter->foto_dokter = $filenameToStore;
69.         $dokter->save();
70.
71.         return redirect()->route('dokter.index')->with('success',
    'Berhasil menyimpan data');
72.         return redirect()->back()->withInput()->withErrors(['error' =>
    'Gagal menyimpan data. Silakan coba lagi.']);
73.     }
74.
75.     /**
76.      * Display the specified resource.
```

```
77.      *
78.      * @param  int  $id
79.      * @return \Illuminate\Http\Response
80.      */
81.     public function show($id)
82.     {
83.         $dokter = Dokter::find($id);
84.         return view('dokter.show', compact('dokter'));
85.     }
86.
87.     /**
88.      * Show the form for editing the specified resource.
89.      *
90.      * @param  int  $id
91.      * @return \Illuminate\Http\Response
92.      */
93.     public function edit($id)
94.     {
95.         $dokter = Dokter::find($id);
96.         $poliklinik = Poliklinik::all(); // Asumsikan Anda memiliki
    model Poliklinik untuk mengambil semua data poliklinik
97.         return view('dokter.update', compact('dokter', 'poliklinik'));
98.     }
99.
100.      /**
101.       * Update the specified resource in storage.
102.       *
103.       * @param  \Illuminate\Http\Request  $request
104.       * @param  int  $id
105.       * @return \Illuminate\Http\Response
106.       */
107.      public function update(Request $request, $id)
108.      {
109.          // Validation remains the same
110.          $validatedData = $request->validate([
111.              'nama_dokter' => 'required|max:255',
112.              'poliklinik_id' => 'required',
113.              'foto_dokter' => 'image|nullable|max:1999'
114.          ]);
115.
116.          // Find the doctor record
117.          $dokter = Dokter::findOrFail($id);
118.
119.          // Only handle photo upload if a new file is provided
120.          if ($request->hasFile('foto_dokter')) {
```

```
121.             $filenameWithExt = $request->file('foto_dokter')->getClientOriginalName();
122.             $filename = pathinfo($filenameWithExt, PATHINFO_FILENAME);
123.             $extension = $request->file('foto_dokter')->getClientOriginalExtension();
124.             $filenameToStore = $filename . '_' . time() . '.' . $extension;
125.             $path = $request->file('foto_dokter')->storeAs('public/foto_dokter', $filenameToStore);
126.
127.             // Delete old photo if it's not the default
128.             if ($dokter->foto_dokter != 'noimage.jpg') {
129.                 Storage::delete('public/foto_dokter/' . $dokter->foto_dokter);
130.             }
131.
132.             // Set the new filename only if a file was uploaded
133.             $dokter->foto_dokter = $filenameToStore;
134.         }
135.
136.         // Update the doctor data
137.         $dokter->nama_dokter = $validatedData['nama_dokter'];
138.         $dokter->poliklinik_id = $validatedData['poliklinik_id'];
139.
140.         if ($dokter->save()) {
141.             return redirect()->route('dokter.index')->with('success', 'Data berhasil diperbarui');
142.         } else {
143.             return redirect()->back()->withInput()->withErrors(['error' => 'Gagal memperbarui data. Silakan coba lagi.']);
144.         }
145.     }
146.
147.     /**
148.      * Remove the specified resource from storage.
149.      *
150.      * @param  int  $id
151.      * @return \Illuminate\Http\Response
152.      */
153.     public function destroy($id)
154.     {
155.         $dokter = Dokter::findOrFail($id);
156.
157.         // Hapus file foto jika bukan 'noimage.jpg'
```

```
158.          if ($dokter->foto_dokter != 'noimage.jpg') {
159.              Storage::delete('public/foto_dokter/' . $dokter-
   >foto_dokter);
160.          }
161.
162.          // Hapus data dokter dari database
163.          $dokter->delete();
164.
165.          return redirect()->route('dokter.index')->with('success',
   'Data berhasil dihapus');
166.      }
167.  }
```

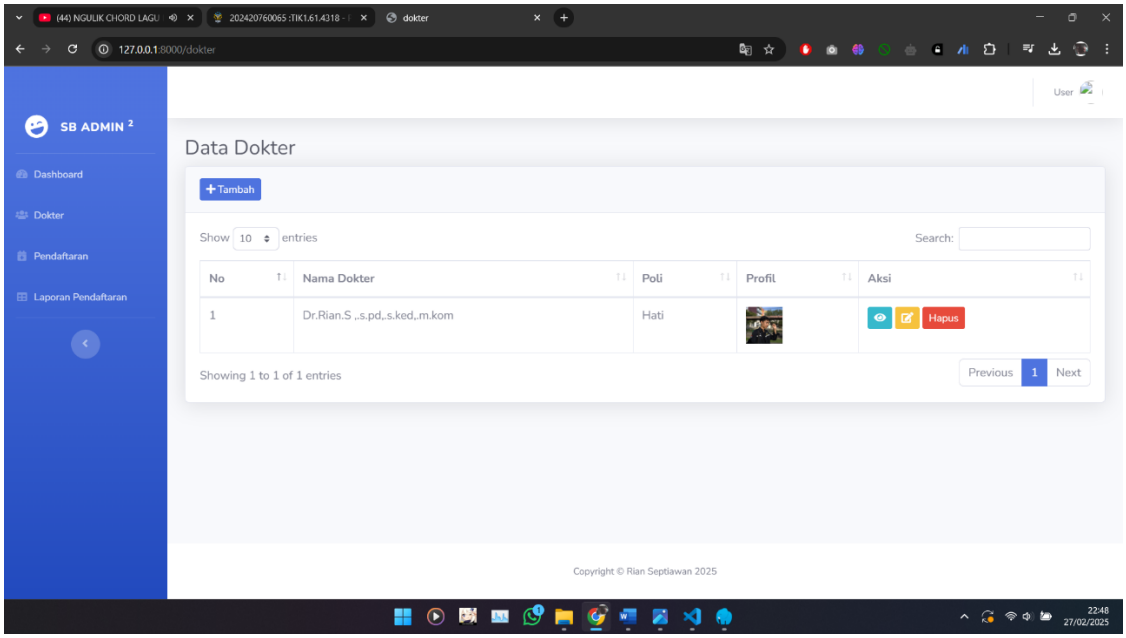
2. Tambahkan ini di bagian **layout** yang bagian menu **Data Dokter**

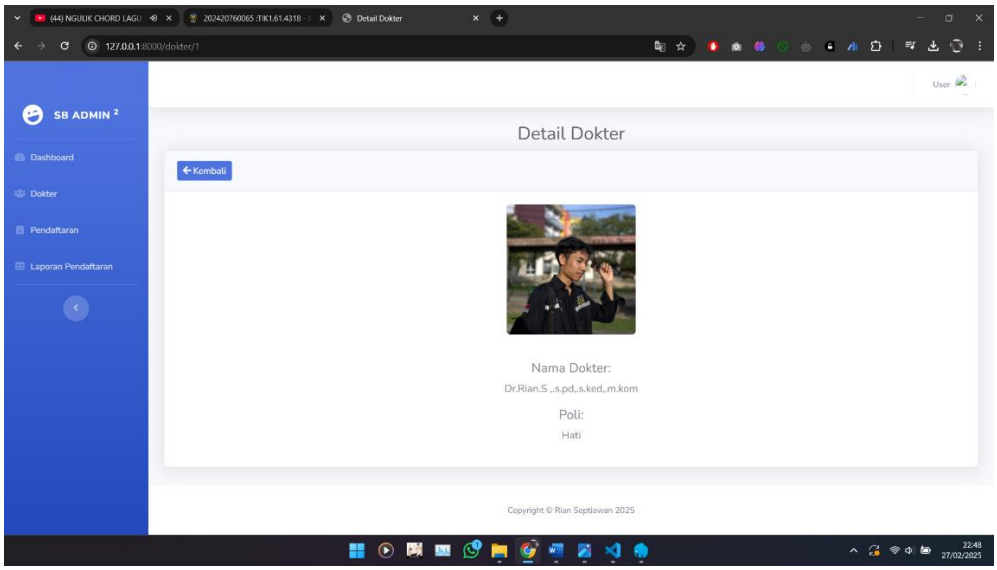
```
<a class="collapse-item" href="{{ route('dokter.index') }}">Data Dokter</a>
```

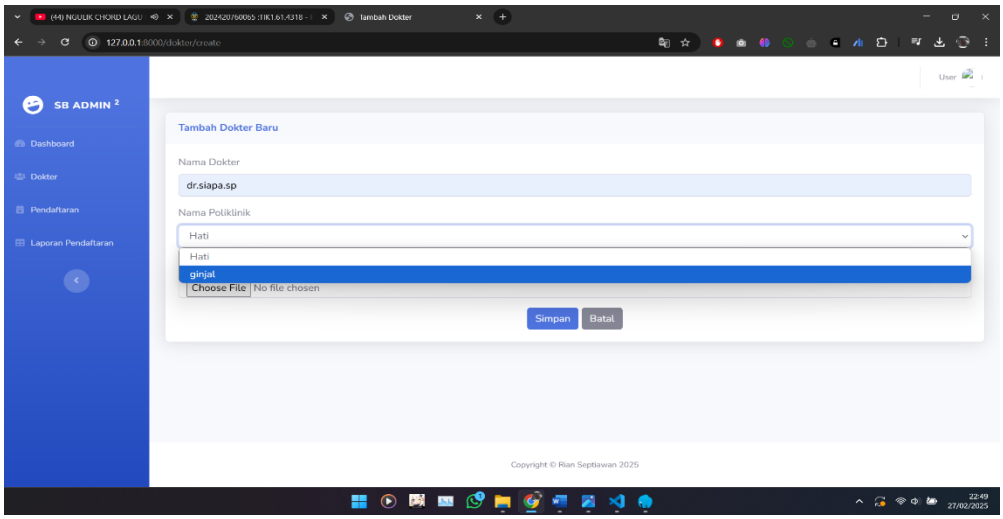
3. Supaya foto dokter bisa tampil, jalankan **php artisan storage:link** di terminal

```
PS C:\laragon\www\rumah-sakit> php artisan storage:link
<!-- web.php -->

  INFO  The [C:\laragon\www\rumah-sakit\public\storage] link has been connected to [C:\laragon\www\rumah-sakit\storage\app/public].

PS C:\laragon\www\rumah-sakit>
```

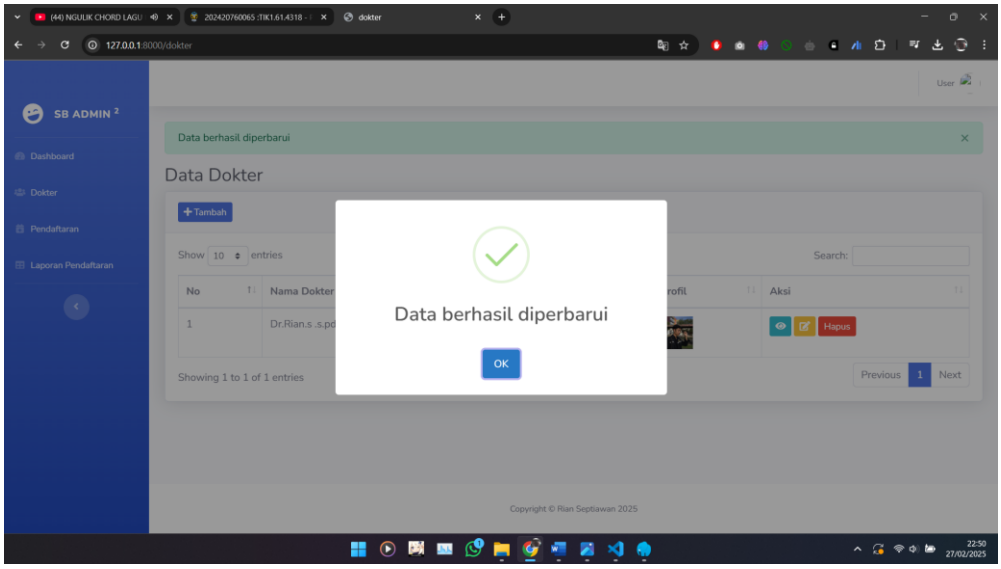
4. Buka link http://127.0.0.1:8000/dokter setelah menjalankan perintah **php artisan serve**







5. Periksa apakah membuat, mengubah, dan menghapus data berfungsi dengan baik!



E. LEMBAR KERJA

1. Lakukan praktikum sesuai langkah kerja yang diberikan
2. Buat laporannya

F. DAFTAR PUSTAKA

[1] Deitel. 2001 Internet & WWW How to Programing, Prentice Hall

[2]. Budi Raharjo. 2012. Modul Prmograman WEB. Modula: Bandung.

[3]. Bunafit Nugroho. 2004. PHP & MySQL Dengan Editor Dreamweaver MX. Andi Yogyakarta

[4] Sklar, David, and Adam Trachtenberg. PHP Cookbook. O'Reilly, 2008.

[5] Welling, Luke. PHP and MySQL Web Development. Developers Library, 2012

[6] Yank, Kevin, and Cameron Adams. Simply Javascript. Sitepoint, 2010

[7] Manual PHP. 2014

[8] Niederst, Jennifer. Learning Web Design (4th edition). O'Reilly. 2013