LAPORAN PRAKTIUM INTERNET OF THINGS (IoT)

Fakultas Vokasi , Universitas Brawijaya

Praktik Pembuatan Tampilan Interface Web Dashboard IoT

Adhini Aulia Tiva

Fakultas Vokasi, Universitas Brawijaya

adiniaulia69@gmail.com

Abstract

Praktikum ini bertujuan untuk memandu pembuatan tampilan interface web dashboard untuk aplikasi Internet of Things (IoT) menggunakan framework Laravel. Fokus utama adalah pada pengembangan fitur visualisasi data sensor dalam bentuk grafik dan fungsionalitas untuk mengekspor data sensor ke dalam format Excel. Pengembangan meliputi instalasi package tambahan seperti maatwebsite/excel untuk ekspor data, pembuatan controller untuk menangani logika penampilan data, pembuatan class export, konfigurasi routing, serta perancangan view menggunakan Blade templating engine dengan integrasi library Chart.js untuk visualisasi grafik. Hasil yang diharapkan dari praktikum ini adalah sebuah halaman web dashboard fungsional yang mampu menampilkan data sensor secara dinamis dan menyediakan opsi ekspor data bagi pengguna.

Keywords: Laravel, Web Dashboard, IoT, Chart.js, Maatwebsite/Excel, PHP, Blade Template, Visualisasi Data, Ekspor Data.

---

1. Introduction (Pendahuluan)

Dalam pengembangan sistem Internet of Things (IoT), visualisasi data merupakan aspek krusial untuk memonitor dan menganalisis informasi yang dikumpulkan dari berbagai sensor. Sebuah web dashboard yang interaktif dan informatif memungkinkan pengguna untuk memahami tren data, membuat keputusan berdasarkan informasi terkini, serta mengelola perangkat IoT secara lebih efektif. Selain visualisasi, kemampuan untuk mengekspor data mentah untuk analisis lebih lanjut atau pelaporan juga seringkali menjadi kebutuhan penting.

Praktikum ini akan membahas langkah-langkah detail untuk membangun sebuah tampilan interface web dashboard menggunakan framework PHP Laravel. Pembahasan akan mencakup dari persiapan awal, instalasi package yang diperlukan, pembuatan komponen backend seperti controller dan export class, hingga perancangan frontend menggunakan Blade template yang diintegrasikan dengan library JavaScript Chart.js untuk membuat grafik data sensor. Fitur ekspor data ke format Excel juga akan diimplementasikan menggunakan package maatwebsite/excel.

Tujuan dari praktikum ini adalah agar praktikan mampu merancang dan mengimplementasikan sebuah web dashboard IoT yang fungsional, menampilkan data sensor dalam bentuk grafik, dan menyediakan fitur ekspor data.

2. Methodology (Metodologi / Langkah-Langkah Pengembangan)

Metodologi yang digunakan dalam pembuatan tampilan interface web dashboard IoT ini mengikuti langkah-langkah terstruktur sebagai berikut, dengan asumsi proyek Laravel dasar telah disiapkan (sesuai Praktik 12). Seluruh pengembangan dilakukan menggunakan Visual Studio Code (VSCode) sebagai editor teks dan terminal terintegrasi.

2.1. Persiapan Awal dan Instalasi Package Langkah pertama adalah membuka folder proyek Laravel yang telah ada di VSCode. Kemudian, dilakukan instalasi package maatwebsite/excel yang berfungsi untuk menangani proses ekspor data ke format Excel. Perintah berikut dijalankan melalui terminal VSCode:

Bash

composer require maatwebsite/excel

Selanjutnya, sebuah controller baru dengan nama GraphController dibuat untuk menangani logika terkait penampilan grafik dan data.

Bash

php artisan make:controller GraphController

2.2. Modifikasi GraphController File app/Http/Controllers/GraphController.php yang telah dibuat kemudian dimodifikasi untuk menambahkan logika pengambilan data sensor dari model dan mengirimkannya ke view. Controller ini juga akan menangani permintaan ekspor data.

PHP

```
<?php


namespace App\Http\Controllers;


use App\Exports\TransaksiSensorExport;

use Maatwebsite\Excel\Facades\Excel;

use App\Models\TransaksiSensor; // Pastikan model TransaksiSensor sudah ada


class GraphController extends Controller

{

/**

* Menampilkan grafik transaksi sensor.

* @return \Illuminate\View\View

*/
```

```
public function index()
{
$transaksiSensors = TransaksiSensor::latest()->take(10)->get();
$labels = $transaksiSensors->pluck('nama_sensor'); // Atau kolom waktu
$dataNilai1 = $transaksiSensors->pluck('nilai1');
$dataNilai2 = $transaksiSensors->pluck('nilai2');

return view('graph', compact('labels', 'dataNilai1', 'dataNilai2'));
}

/**
* Mengunduh data transaksi sensor dalam format Excel
* @return \Symfony\Component\HttpFoundation\BinaryFileResponse
*/
public function exportToExcel()
{
return Excel::download(new TransaksiSensorExport, 'transaksi_sensor.xlsx');
}
}
```

Catatan: Model App\Models\TransaksiSensor diasumsikan sudah ada dan memiliki kolom seperti nama_sensor, nilai1, dan nilai2.

2.3. Pembuatan Class Export Data Untuk menangani logika ekspor data ke Excel, sebuah class export bernama TransaksiSensorExport dibuat menggunakan perintah Artisan. Class ini dihubungkan langsung dengan model TransaksiSensor.

Bash

```
php artisan make:export TransaksiSensorExport --model=TransaksiSensor
```

2.4. Modifikasi TransaksiSensorExport File app/Exports/TransaksiSensorExport.php kemudian diisi dengan logika untuk mengambil semua data dari model TransaksiSensor.

PHP

```
<?php

namespace App\Exports;
```

```
use App\Models\TransaksiSensor; // Pastikan model TransaksiSensor sudah ada
use Maatwebsite\Excel\Concerns\FromCollection;
use Maatwebsite\Excel\Concerns\WithHeadings; // Opsional

class TransaksiSensorExport implements FromCollection, WithHeadings
{
/**
* @return \Illuminate\Support\Collection
*/
public function collection()
{
return TransaksiSensor::all();
}

// Opsional: untuk header kolom di Excel
public function headings(): array
{
return [
'ID', 'Nama Sensor', 'Nilai 1', 'Nilai 2', 'Waktu Dibuat', 'Waktu Diupdate'
// Sesuaikan dengan kolom pada tabel TransaksiSensor
];
}
}
```

2.5. Konfigurasi Rute (Routes) File routes/web.php diedit untuk mendefinisikan rute yang akan mengarahkan permintaan HTTP ke metode yang sesuai di GraphController.

PHP

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\GraphController;
```

Route::get('/', [GraphController::class, 'index'])->name('graph.index');

Route::get('/graph/export', [GraphController::class, 'exportToExcel'])->name('graph.export');

2.6. Pembuatan Tampilan Blade untuk Grafik (graph.blade.php) Sebuah file view baru dibuat dengan nama graph.blade.php di dalam direktori resources/views/. File ini berisi struktur HTML untuk halaman dashboard dan kode JavaScript untuk merender grafik menggunakan Chart.js.

HTML

<!DOCTYPE html>

<html lang="id">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard Sensor IoT</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<style>

body { font-family: Arial, sans-serif; margin: 20px; background-color: #f4f7f6; }

.container { max-width: 800px; margin: auto; background-color: #fff; padding: 20px; border-radius: 8px; box-shadow: 0 0 10px rgba(0,0,0,0.1); }

h1 { color: #333; text-align: center; }

.chart-container { position: relative; height:40vh; width:80vw; margin: auto; margin-top: 30px; }

.export-button { display: block; width: 150px; margin: 20px auto; padding: 10px 15px; background-color: #4CAF50; color: white; text-align: center; text-decoration: none; border-radius: 5px; }

.export-button:hover { background-color: #45a049; }

</style>

</head>

<body>

<h1>Grafik Perbandingan Sensor</h1>

<canvas id="sensorChart"></canvas>

```
</div>
<a href="{{ route('graph.export') }}" class="export-button">Export Data ke Excel</a>
</div>
<script>
const ctx = document.getElementById('sensorChart').getContext('2d');
const sensorChart = new Chart(ctx, {
type: 'line',
data: {
labels: @json($labels),
datasets: [
{
label: 'Sensor 1 (Nilai 1)',
data: @json($dataNilai1),
borderColor: 'rgb(75, 192, 192)',
tension: 0.1,
fill: false
},
{
label: 'Sensor 2 (Nilai 2)',
data: @json($dataNilai2),
borderColor: 'rgb(255, 99, 132)',
tension: 0.1,
fill: false
}
]
},
options: {
responsive: true,
maintainAspectRatio: false,
scales: { y: { beginAtZero: true } },
```

plugins: { legend: { position: 'top' }, title: { display: true, text: 'Data Sensor Terbaru' } }

}

});

</script>

</body>

</html>

3. Results and Discussion (Hasil yang Diharapkan dan Pembahasan)

3.1. Expected Results (Hasil yang Diharapkan) Setelah mengikuti seluruh langkah metodologi di atas, hasil yang diharapkan adalah sebagai berikut:

1. Sebuah halaman web dashboard dapat diakses melalui browser (misalnya, http://localhost:8000).
2. Halaman dashboard menampilkan grafik garis (line chart) yang merepresentasikan data dari nilai1 dan nilai2 dari 10 entri terbaru model TransaksiSensor. Sumbu X grafik akan menampilkan label yang diambil dari kolom nama_sensor (atau kolom waktu jika disesuaikan).
3. Terdapat tombol "Export Data ke Excel" pada halaman dashboard. Ketika tombol ini diklik, browser akan mengunduh sebuah file transaksi_sensor.xlsx yang berisi seluruh data dari tabel TransaksiSensor.
4. Struktur proyek Laravel akan memiliki GraphController, TransaksiSensorExport, dan view graph.blade.php yang telah dikonfigurasi.

(Pembahasan) Pemilihan Chart.js sebagai library untuk visualisasi data didasarkan pada kemudahannya dalam penggunaan, sifatnya yang open-source, serta dokumentasi yang lengkap. Library ini mampu menghasilkan berbagai jenis grafik yang responsif dan interaktif dengan konfigurasi yang relatif sederhana.

Penggunaan maatwebsite/excel untuk fungsionalitas ekspor data merupakan pilihan populer di ekosistem Laravel karena menyediakan abstraksi yang kuat untuk membaca dan menulis berbagai format spreadsheet, termasuk Excel. Hal ini memudahkan pengembang untuk mengimplementasikan fitur ekspor tanpa perlu menangani detail format file secara manual.

Pemisahan logika ke dalam Controller, Export Class, dan View (Blade) mengikuti prinsip Model-View-Controller (MVC) yang dianut oleh Laravel, menghasilkan kode yang lebih terstruktur, mudah dipelihara, dan skalabel. Pengambilan data dibatasi pada 10 entri terbaru (latest()->take(10)) untuk menjaga performa tampilan awal dashboard, sementara fitur ekspor akan mengambil keseluruhan data (TransaksiSensor::all()). Kustomisasi lebih lanjut pada query data, baik untuk tampilan maupun ekspor, dapat dilakukan sesuai kebutuhan spesifik aplikasi.

Beberapa faktor yang perlu diperhatikan saat implementasi meliputi:

- Ketersediaan Model: Pastikan model TransaksiSensor telah dibuat melalui migration dan sesuai dengan struktur tabel di database.

- Nama Kolom: Konsistensi nama kolom pada model, controller, dan view sangat penting.
- Koneksi Internet: Untuk penggunaan Chart.js melalui CDN, koneksi internet diperlukan saat memuat halaman. Alternatifnya, Chart.js dapat di-host secara lokal.

4. Kesimpulan

Praktikum ini telah berhasil memaparkan langkah-langkah untuk membuat tampilan interface web dashboard IoT menggunakan Laravel. Dengan mengikuti metodologi yang dijelaskan, sebuah dashboard fungsional yang mampu menampilkan data sensor dalam bentuk grafik dan menyediakan fitur ekspor data ke Excel telah berhasil dirancang. Penggunaan Laravel bersama dengan package seperti maatwebsite/excel dan library JavaScript seperti Chart.js terbukti efektif dalam membangun aplikasi web yang interaktif dan kaya fitur untuk keperluan monitoring sistem IoT. Pemahaman terhadap konsep Controller, Model, View, Routing, serta integrasi dengan library eksternal menjadi kunci keberhasilan dalam pengembangan ini.

5. Dokumentasi