**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)**

**Fakultas Vokasi, Universitas Brawijaya**

*Pembuatan Tampilan Interface Web Dashboard IoT Menggunakan Laravel dan Chart.js*

Faiz Henri Kurniawan

Fakultas Vokasi, Universitas Brawijaya

Email: faizhenrik@student.ub.ac.id

Abstrak

Praktikum ini bertujuan untuk merancang dan mengimplementasikan sebuah tampilan antarmuka web dashboard untuk sistem Internet of Things (IoT). Dashboard ini berfungsi untuk memvisualisasikan data sensor secara real-time atau berdasarkan histori data yang tersimpan dalam database. Proses pembuatan dashboard melibatkan penggunaan framework Laravel untuk pengembangan sisi backend dan routing, pustaka Maatwebsite/Excel untuk fungsionalitas ekspor data ke format Excel, serta Chart.js untuk pembuatan grafik yang interaktif di sisi frontend. Data sensor yang digunakan berasal dari model TransaksiSensor yang mencakup nama sensor serta dua nilai sensor (nilai1 dan nilai2). Hasil dari praktikum ini adalah sebuah aplikasi web yang mampu menampilkan grafik tren data sensor, ringkasan statistik data seperti rata-rata dan korelasi antar sensor, serta menyediakan fitur untuk mengunduh data sensor. Implementasi ini menunjukkan kemampuan Laravel dan Chart.js dalam membangun solusi monitoring data IoT yang informatif dan mudah diakses.

Keywords—Internet of Things, Dashboard Web, Laravel, Chart.js, Monitoring Sensor, Visualisasi Data

1. Pendahuluan

1.1 Latar Belakang Praktikum IoT yang Dilakukan

Seiring dengan pesatnya perkembangan teknologi Internet of Things (IoT), jumlah data yang dihasilkan oleh berbagai sensor semakin meningkat. Data ini memiliki nilai yang besar jika dapat diolah dan disajikan dengan baik untuk keperluan monitoring, analisis, dan pengambilan keputusan. Tampilan antarmuka web berupa dashboard menjadi salah satu solusi efektif untuk memvisualisasikan data sensor secara intuitif dan mudah dipahami. Dashboard IoT memungkinkan pengguna untuk memantau kondisi lingkungan atau sistem secara real-time, mengidentifikasi tren, serta mendeteksi anomali dari data yang terkumpul. Praktikum ini berfokus pada pengembangan sebuah web dashboard IoT menggunakan framework Laravel, yang populer karena struktur MVC (Model-View-Controller) dan kemudahan pengembangannya, serta pustaka Chart.js untuk menyajikan data dalam bentuk grafik yang dinamis dan interaktif. Selain itu, akan diimplementasikan juga fitur ekspor data untuk analisis lebih lanjut.

1.2 Tujuan Eksperimen

Adapun tujuan dari pelaksanaan eksperimen dalam praktikum ini adalah sebagai berikut:

* Memahami konsep dan arsitektur pembuatan web dashboard untuk aplikasi IoT.
* Mempelajari implementasi pengambilan data sensor dari database (dalam konteks ini, model TransaksiSensor pada database iot\_25) menggunakan Laravel.
* Mengimplementasikan visualisasi data sensor dalam bentuk grafik garis (line chart) menggunakan pustaka Chart.js.
* Mengembangkan fitur untuk mengekspor data sensor ke dalam format file Excel menggunakan paket Maatwebsite/Excel di Laravel.
* Mampu membuat *controller*, *route*, dan *view* (*blade template*) dalam *framework* Laravel untuk membangun fungsionalitas dashboard IoT.
* Menghasilkan sebuah tampilan *interface* web dashboard yang fungsional dan informatif.

2. Metodologi

2.1 Tools & Materials (Alat dan Bahan)

Berikut adalah alat dan bahan yang digunakan dalam praktikum ini:

* Perangkat Keras:
  + Komputer/Laptop dengan sistem operasi yang mendukung pengembangan web.
  + (Dalam konteks sistem IoT yang lebih luas, praktikum sebelumnya mungkin melibatkan mikrokontroler seperti ESP8266/Arduino, dan berbagai sensor, namun untuk bab ini fokus pada pengembangan *software* dashboardnya).
* Perangkat Lunak:
  + *Text Editor*: Visual Studio Code (VSCode) atau sejenisnya.
  + *Web Browser*: Google Chrome, Mozilla Firefox, atau sejenisnya.
  + Terminal atau *Command Prompt*.
  + PHP: Versi yang kompatibel dengan Laravel.
  + Composer: Manajer dependensi untuk PHP.
  + Laravel Framework: *Framework* aplikasi web berbasis PHP.
  + Paket Laravel: maatwebsite/excel (untuk ekspor data ke Excel).
  + Pustaka JavaScript: Chart.js (untuk membuat grafik).
  + Pustaka CSS & Font:
    - Font Awesome (untuk ikon).
    - Google Fonts (Poppins).
    - Animate.css (untuk animasi dasar).
  + Sistem Manajemen Database: Seperti MySQL, PostgreSQL, atau SQLite yang dikonfigurasi untuk proyek Laravel (disebutkan database iot\_25).

2.2 Implementation Steps (Langkah Implementasi)

Langkah-langkah implementasi pembuatan web dashboard IoT adalah sebagai berikut:

1. Persiapan Proyek Laravel: Buka folder proyek Laravel yang telah dibuat pada Praktik 12 menggunakan VSCode.
2. Instalasi Paket Excel: Buka terminal pada VSCode dan jalankan perintah composer require maatwebsite/excel untuk menginstal paket yang akan digunakan untuk fungsionalitas ekspor data.
3. Pembuatan Controller: Buat *controller* baru dengan nama GraphController menggunakan perintah Artisan: php artisan make:controller GraphController.
4. Implementasi Logika pada GraphController:
   * Tambahkan *use statement* untuk App\Exports\TransaksiSensorExport, Maatwebsite\Excel\Facades\Excel, dan App\Models\TransaksiSensor.
   * Buat metode index():
     + Mengambil 10 data transaksi sensor terbaru (TransaksiSensor::latest()->take(10)->get()).
     + Mengekstrak nama\_sensor sebagai label untuk grafik.
     + Mengekstrak nilai1 dan nilai2 sebagai data untuk grafik.
     + Mengirimkan data tersebut ke *view* graph menggunakan compact().
   * Buat metode exportToExcel():
     + Menggunakan Excel::download() untuk mengunduh data dengan memanfaatkan kelas TransaksiSensorExport.
5. Pembuatan Kelas Export: Buat kelas *export* dengan nama TransaksiSensorExport yang terhubung dengan model TransaksiSensor menggunakan perintah Artisan: php artisan make:export TransaksiSensorExport --model=TransaksiSensor.
6. Implementasi Logika pada TransaksiSensorExport:
   * Tambahkan *use statement* untuk App\Models\TransaksiSensor dan Maatwebsite\Excel\Concerns\FromCollection.
   * Implementasikan *interface* FromCollection.
   * Dalam metode collection(), kembalikan semua data dari model TransaksiSensor (TransaksiSensor::all()).
7. Konfigurasi Rute (Routes): Edit file routes/web.php:
   * Tambahkan *use statement* untuk App\Http\Controllers\GraphController.
   * Definisikan rute GET untuk / yang mengarah ke metode index pada GraphController dengan nama graph.
   * Definisikan rute GET untuk /graph/export yang mengarah ke metode exportToExcel pada GraphController dengan nama graph.export.
8. Pembuatan View (Blade Template): Buat file baru bernama graph.blade.php di dalam folder resources/views/.
9. Desain Tampilan Interface (HTML, CSS, JavaScript): Isi file graph.blade.php dengan kode berikut:
   * Struktur HTML: Membuat layout dasar dashboard, termasuk *header*, area untuk grafik, dan kartu ringkasan data.
   * Styling CSS: Menggunakan CSS *custom* dan pustaka eksternal (Font Awesome, Google Fonts, Animate.css) untuk tampilan yang menarik dan responsif. Mendefinisikan variabel warna, *styling* untuk *card*, tombol, *chart container*, dan *summary card*.
   * Logika JavaScript:
     + Mengambil data (labels, dataNilai1, dataNilai2) yang dikirim dari GraphController menggunakan direktif Blade @json().
     + Fungsi calculateStats() untuk menghitung rata-rata dari data sensor.
     + Fungsi calculateCorrelation() untuk menghitung koefisien korelasi Pearson antara dataNilai1 dan dataNilai2.
     + Menampilkan hasil kalkulasi (rata-rata, korelasi) pada elemen HTML yang sesuai. (Catatan: Persentase perubahan pada contoh kode bersifat acak dan dapat dikembangkan lebih lanjut untuk perbandingan data aktual).
     + Menginisialisasi *line chart* menggunakan Chart.js dengan data sensor, label, dan berbagai opsi konfigurasi (responsif, interaksi, legenda, *tooltip*, anotasi garis rata-rata).
     + Menambahkan fungsionalitas dasar untuk tombol pemilih rentang waktu (pada contoh kode, ini hanya memperbarui grafik dengan data acak sebagai demonstrasi).
10. Menjalankan Aplikasi: Jalankan *server* pengembangan Laravel menggunakan perintah php artisan serve di terminal.
11. Pengujian: Buka *web browser* dan akses alamat yang diberikan oleh php artisan serve (biasanya http://127.0.0.1:8000) untuk melihat dashboard, grafik data sensor, dan menguji fungsionalitas ekspor ke Excel.

3. Results and Discussion (Hasil dan Pembahasan)

3.1 Experimental Results (Hasil Eksperimen)

Setelah mengikuti langkah-langkah implementasi, diperoleh hasil sebagai berikut:

* Sebuah *interface web dashboard* berhasil dibuat dan dapat diakses melalui *web browser* pada alamat yang disediakan oleh *server* pengembangan Laravel.
* Dashboard menampilkan judul "Dashboard Monitoring Sensor" dengan ikon yang relevan.
* Terdapat tombol pemilih rentang waktu (24 Jam, 7 Hari, 30 Hari, Custom), meskipun fungsionalitas lanjutannya pada contoh kode masih bersifat demonstratif (memperbarui grafik dengan data acak).
* Sebuah kartu utama (*card*) menampilkan "Grafik Perbandingan Sensor" yang berisi:
  + *Line chart* (grafik garis) yang dibuat menggunakan Chart.js. Grafik ini memvisualisasikan dua set data sensor (Sensor 1 dari nilai1 dan Sensor 2 dari nilai2) berdasarkan label nama sensor (nama\_sensor). Garis rata-rata untuk masing-masing sensor juga ditampilkan sebagai anotasi pada grafik.
  + Tombol "Export" dengan ikon *download*, yang ketika diklik akan mengunduh file transaksi\_sensor.xlsx. File ini berisi semua data dari tabel TransaksiSensor di database iot\_25.
* Di bawah grafik, terdapat tiga kartu ringkasan (*summary card*):
  + Sensor 1 (Rata-rata): Menampilkan nilai rata-rata dari dataNilai1 yang diambil dari 10 data sensor terakhir. Dilengkapi dengan ikon dan persentase perubahan (yang dalam kode contoh dihasilkan secara acak).
  + Sensor 2 (Rata-rata): Menampilkan nilai rata-rata dari dataNilai2. Serupa dengan Sensor 1, persentase perubahan juga dihasilkan acak.
  + Korelasi: Menampilkan nilai korelasi antara dataNilai1 dan dataNilai2, beserta interpretasi kekuatan korelasi (kuat, sedang, atau lemah).
* Tampilan dashboard dirancang agar responsif dan memiliki animasi sederhana untuk meningkatkan pengalaman pengguna.
* (Pada laporan fisik, bagian ini idealnya akan menyertakan *screenshot* dari tampilan web dashboard yang berhasil dijalankan, menunjukkan grafik data sensor, kartu ringkasan, dan tombol ekspor.)

Pembahasan:

Pembuatan dashboard IoT menggunakan Laravel dan Chart.js terbukti menjadi pendekatan yang efektif. Laravel menyediakan struktur yang kuat untuk backend, pengelolaan data (model TransaksiSensor), dan routing. Penggunaan Eloquent ORM memudahkan interaksi dengan database untuk mengambil data sensor. Paket maatwebsite/excel menyederhanakan proses pembuatan fungsionalitas ekspor data ke format Excel, yang sangat berguna untuk analisis data lebih lanjut secara offline.

Di sisi *frontend*, Chart.js menawarkan kemudahan dalam membuat grafik yang interaktif dan kaya fitur dengan berbagai opsi kustomisasi. Data dari *controller* Laravel dapat dengan mudah diteruskan ke *view* dan kemudian digunakan oleh JavaScript untuk menginisialisasi grafik. Perhitungan statistik tambahan seperti rata-rata dan korelasi langsung di JavaScript *frontend* (berdasarkan data yang sudah diambil) memungkinkan *update* tampilan yang cepat tanpa perlu *request* tambahan ke *server* untuk setiap kalkulasi sederhana tersebut.

Desain CSS yang diterapkan memberikan tampilan modern dan profesional pada *dashboard*. Penggunaan *framework* CSS atau *utility classes* dapat lebih mempercepat proses *styling* di proyek yang lebih besar. Fungsionalitas tombol pemilih rentang waktu yang pada contoh ini masih dasar (memperbarui dengan data acak) menunjukkan potensi untuk dikembangkan lebih lanjut agar dapat mengambil dan memfilter data dari *server* berdasarkan rentang waktu yang dipilih pengguna, sehingga *dashboard* menjadi lebih dinamis dan analitis.

Secara keseluruhan, praktikum ini berhasil mendemonstrasikan langkah-langkah penting dalam membuat *interface web dashboard* IoT yang fungsional untuk visualisasi dan ekspor data sensor.

4. Appendix (Lampiran, jika diperlukan)

Kode Program GraphController.php:

PHP

<?php

namespace App\Http\Controllers;

use App\Exports\TransaksiSensorExport;

use Maatwebsite\Excel\Facades\Excel;

use App\Models\TransaksiSensor;

class GraphController extends Controller

{

/\*\*

\* Menampilkan grafik transaksi sensor.

\*

\* @return \Illuminate\View\View

\*/

public function index()

{

// Mengambil data transaksi sensor

$transaksiSensors = TransaksiSensor::latest()->take(10)->get();

// Mengambil data label

$labels = $transaksiSensors->pluck('nama\_sensor');

// Mengambil data nilai1 dan nilai2 untuk grafik

$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');

}

}

Kode Program TransaksiSensorExport.php:

PHP

<?php

namespace App\Exports;

use App\Models\TransaksiSensor;

use Maatwebsite\Excel\Concerns\FromCollection;

class TransaksiSensorExport implements FromCollection

{

/\*\*

\* @return \Illuminate\Support\Collection

\*/

public function collection()

{

return TransaksiSensor::all();

}

}

Kode Program routes/web.php (relevan):

PHP

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\GraphController;

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

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

**Lampiran**

  
  
