

# Desain Sistem Backend Real Time Data Processing Untuk Webiste Faculty net Monitoring

1<sup>st</sup> David Junior  
Fakultas Teknik Elektro  
Universitas Telkom  
Bandung, Indonesia

davidjunior@student.telkomuniversity.ac.id

2<sup>nd</sup> Uke Kurniawan Usman  
Fakultas Teknik Elektro  
Universitas Telkom  
Bandung, Indonesia

ukeusman@telkomuniversity.ac.id

3<sup>rd</sup> Akhmad Hambali  
Fakultas Teknik Elektro  
Universitas Telkom  
Bandung, Indonesia

ahambali@telkomuniversity.ac.id

**Abstrak** — Kualitas koneksi jaringan sangat bergantung pada stabilitas jaringan. Oleh karena itu, pemantauan rutin sangat penting untuk memantau kondisi lalu lintas jaringan. Dengan memantau perangkat jaringan secara *real-time*, dapat dengan mudah mengetahui status lalu lintas jaringan dan menyelesaikan masalah dengan cepat. Untuk itu, diperlukan antarmuka berbasis *website* yang memungkinkan untuk memantau jaringan pada setiap perangkat *access point*. Dalam pengembangannya, *website* ini menggunakan *framework* Express JS sebagai *framework back-end* dan menggunakan MySQL sebagai *database* untuk menyimpan data monitoring jaringan.

**Kata kunci**— *real-time, database, Website,*

## I. PENDAHULUAN

Teknologi *web* saat ini berkembang pesat dan terbagi menjadi beberapa bidang fokus utama, termasuk pengembangan *back-end*. Bagian *back-end* ini mengacu pada program yang berjalan di server dan berfungsi sebagai wadah untuk logika fungsional inti dan operasi aplikasi perangkat lunak atau sistem informasi. Sistem *back-end* bertanggung jawab untuk menyediakan data atau layanan yang diminta oleh sistem atau aplikasi *front-end* melalui metode yang terprogram. Komponen backend mencakup logika aplikasi inti, *database*, integrasi data dan aplikasi, API, dan proses lainnya.

## II. KAJIAN TEORI

### A. Webiste

*Website* merupakan kumpulan halaman *web* yang terkait dan saling berhubungan, yang dapat diakses melalui internet. Situs *web* biasanya memiliki nama domain yang unik dan dapat diakses melalui alamat IP atau URL (Uniform Resource Locator). Situs *web* dapat berisi berbagai jenis konten, seperti teks, gambar, video, audio, dan lain-lain.

### B. Database

*Database* adalah kumpulan data yang terorganisir dan terstruktur, yang dapat diakses dan dikelola melalui sistem komputer. *Database* dapat berupa koleksi data yang besar dan kompleks, seperti data pelanggan, data produk, data transaksi, dan lain-lain.

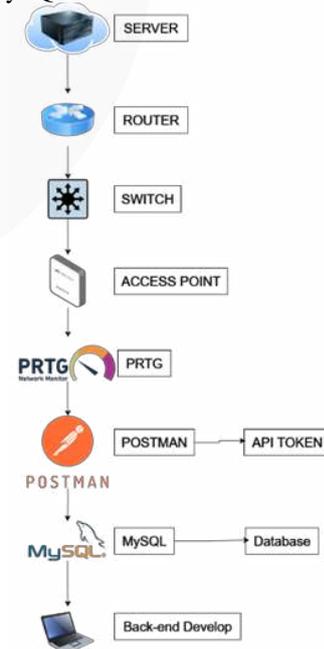
### C. API Token

API Token adalah sebuah kode unik yang digunakan untuk mengakses dan mengautentikasi permintaan ke sebuah API (Application Programming Interface). Token ini berfungsi sebagai identitas pengguna atau aplikasi yang meminta akses ke API, sehingga API dapat memverifikasi dan mengizinkan akses ke sumber daya yang diminta.

## III. METODE

### A. Perancangan Sistem

Pada bagian ini, menunjukkan desain sistem perancangan monitoring jaringan pada *back-end*. Desain sistem monitoring jaringan pada *back-end* yang terdiri dari data monitoring yang telah di ambil dari perangkat acces point akan di tampilkan pada *software* PRTG Network Monitor, ssetalah itu, data yang telah di tampilkan pada *software* PRTG Network Monitor akan di ambil datanya dengan API Token menggunakan *software* Postman, dan data yang telah ada di *software* Postman akan di masukkan ke dalam *database* menggunakan MySQL.



GAMBAR 1

Desain Sistem *Back-end*

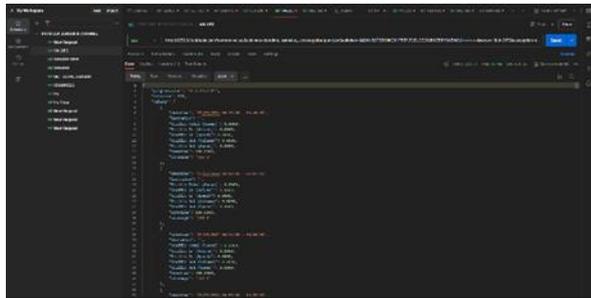
B. Implementasi Sistem

Tampilan Implementasi desain sistem *back-end* pada *website* *faculty*net monitor.



GAMBAR 2  
Laptop Yang Terhubung Dengan Server

Pada gambar 2 mempilkan laptop yang telah terhubung dengan server untuk proses pengambilan data.



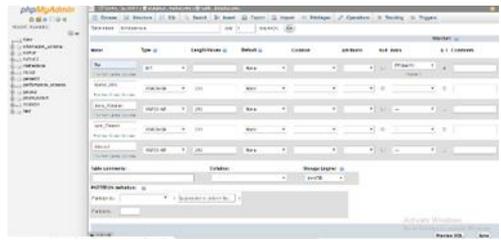
GAMBAR 3  
Data Monitoring Jaringan Pada PRTG

Pada gambar 3 menampilkan data monitoring jaringan pada PRTG yang telah di dapatkan dari server.



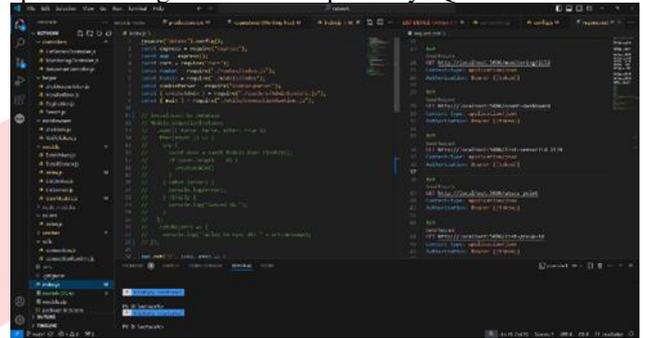
GAMBAR 4  
Data Monitoring Jaringan Pada Postman

Pada gambar 4 menampilkan data pada Postman yang telah di ambil dari PRTG Network Monitor menggunakan API Token.



GAMBAR 5  
Database Pada MySQL

Pada gambar 5 menampilkan database yang telah di ambil dari postman dengan API Token pada MySQL.



GAMBAR 6  
Source Code Back-end

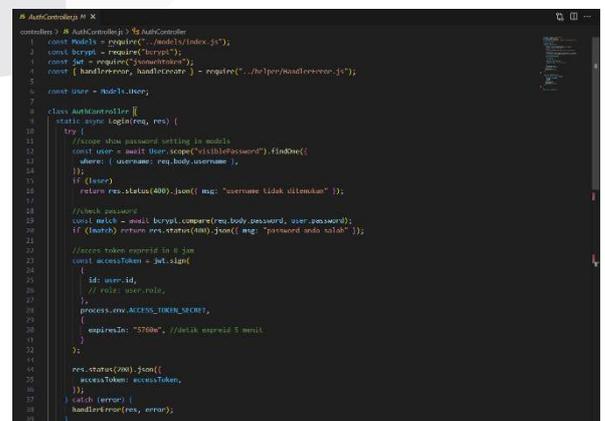
Pada Gambar 6 menampilkan *source code back-end developer* yang data – data monitoring jaringannya telah di ambil dari database MySQL.

IV. HASIL DAN PEMBAHASAN

Pada tahap hasil dan pembahasan dari perencanaan adalah pengujian untuk mengetahui proses pengembangan pada *back-end* berjalan sesuai dengan yang diharapkan. Pembuatan *website faculty*net monitoring menggunakan metode pengujian kualitas *website* dengan menggunakan pengujian fungsionalitas *website*. Pengujian desain sistem pengembangan *back-end* dilakukan untuk memastikan bahwa data monitoring akan tampil pada *website faculty*net monitoring.

A. Membuat Model AuthController

Pembuatan *model authcontroller* bertujuan untuk melakukan autentikasi control ketika ada user yang melakukan *login* pada *website faculty*net monitor.



GAMBAR 7  
Source Code Model AuthController

B. Pembuatan Dashboard Controller

Pembuatan dashboard controller bertujuan untuk menampilkan data – data jumlah *access point* pada *dashboard website* sebagai halaman utaman ketika pengguna sudah melakukan *login*.

```
# dashboardController.js
const { Router } = require('express');
const { errorHandler } = require('../helpers/handleError.js');

class DashboardController {
  static async countDashboard(req, res) {
    try {
      let countRouter = 0;
      let countAccessPoint = 0;
      let countSensorPerBuilding = 0;

      const Model = this.$context.$find('Model');
      const data = await Model.find().then((data) => {
        data.forEach((item) => {
          countRouter += item.routerCount;
          countAccessPoint += item.accessPointCount;
          countSensorPerBuilding += item.sensorCount;
        });
      });

      res.json({
        routerCount: countRouter,
        accessPointCount: countAccessPoint,
        sensorCount: countSensorPerBuilding
      });
    } catch (error) {
      errorHandler(res, error);
    }
  }

  static async getDashboard(req, res) {
    const data = await this.$context.$get('Model').query();
    // console.log('Dashboard Data:', data);
    if (data.error) {
      return errorHandler(res, data);
    }
    // console.log('Dashboard Data:', data);
    return res.json(data);
  }
}

module.exports = DashboardController;
```

GAMBAR 8 Source Code Dashboard Controller

E. Pembuatan Monitoring Controller

Pembuatan *Monitoring Controller* bertujuan untuk melihat dan mengambil data – data *monitoring* dari *database MySQL* yang akan di tampilkan di *website* *faculty net monitor*.

```
# MonitoringController.js
const { Router } = require('express');
const { errorHandler } = require('../helpers/handleError.js');

class MonitoringController {
  static async getMonitoring(req, res) {
    try {
      // Mengambil data dari database dengan query
      // dan melakukan filter data yang akan ditampilkan
      const data = await this.$context.$get('Model').query();
      // console.log('Monitoring Data:', data);
      if (data.error) {
        return errorHandler(res, data);
      }
      // Mengambil data yang akan ditampilkan
      const dataToDisplay = data.filter((item) => {
        // Filter data yang akan ditampilkan
        return item.status === 'active';
      });
      // console.log('Monitoring Data (Filtered):', dataToDisplay);
      return res.json(dataToDisplay);
    } catch (error) {
      errorHandler(res, error);
    }
  }
}

module.exports = MonitoringController;
```

GAMBAR 10 Source Code Monitoring Controller

C. Pembuatan List Device Controller

Pembuatan *list device controller* bertujuan untuk melihat dan menampilkan data – data perangkat *access point* dari masing masing gedung.

```
# ListDeviceController.js
const { Router } = require('express');
const { errorHandler } = require('../helpers/handleError.js');

class ListDeviceController {
  static async getDevice(req, res) {
    try {
      const Model = this.$context.$find('Model');
      const data = await Model.find().then((data) => {
        const accumulator = data.reduce((accumulator, element) => {
          return accumulator.concat(element.devices);
        }, []);
      });
      res.json(accumulator);
    } catch (error) {
      errorHandler(res, error);
    }
  }

  static async getGroupDevice(req, res) {
    try {
      const Model = this.$context.$find('Model');
      const data = await Model.find().then((data) => {
        // console.log('Group Device Data:', data);
      });
    } catch (error) {
      errorHandler(res, error);
    }
  }
}

module.exports = ListDeviceController;
```

GAMBAR 8 Source Code List Device Controller

D. Pembuatan List Sensor Controller

Pembuatan *list sensor controller* bertujuan untuk melihat dan menampilkan data – data sensor dari masing masing perangkat *access point*.

```
# ListSensorController.js
const { Router } = require('express');
const { errorHandler } = require('../helpers/handleError.js');

class ListSensorController {
  static async getSensor(req, res) {
    try {
      const Model = this.$context.$find('Model');
      const data = await Model.find().then((data) => {
        // console.log('Sensor Data:', data);
      });
    } catch (error) {
      errorHandler(res, error);
    }
  }
}

module.exports = ListSensorController;
```

GAMBAR 9 Source Code List Sensor Controller

V. KESIMPULAN

Sistem monitoring jaringan yang dikembangkan dalam penelitian ini dirancang untuk mempermudah pemantauan lalu lintas jaringan secara *real-time* sehingga memungkinkan pengguna untuk melihat data *monitoring* secara *real-time*. Sistem ini juga dilengkapi dengan fitur autentikasi pengguna melalui *AuthController*, sehingga hanya pengguna yang berhak yang dapat mengakses data monitoring.

Sistem ini juga memiliki beberapa fitur lainnya, seperti *Dashboard Controller* untuk menampilkan data *access point* pada halaman utama, *List Device Controller* dan *List Sensor Controller* untuk menampilkan data perangkat dan sensor dari berbagai gedung, serta *Monitoring Controller* untuk mengambil dan menampilkan data *monitoring* dari *database* ke *website* *FacultyNet Monitor*. Hasil pengujian menunjukkan bahwa semua fitur berfungsi sesuai dengan harapan. Dengan demikian, sistem *back-end* yang dirancang dan di implementasikan telah memenuhi tujuan utama dalam menyediakan *platform* pemantauan jaringan yang mudah dan efisien.

REFERENSI

- [1] Oktiawati, U. Y., Erlangga, G., & Irving, V. (2022). PENGEMBANGAN APLIKASI MONITORING BANDWIDTH USAGE BERBASIS WEBSITE MENGGUNAKAN DJANGO. *AUTOMATA*, 3(2).
- [2] Herdiyatomoko, H. F. (2022). Back-End System Design Based on Rest Api. *Jurnal Tekinkom (Teknik Informasi dan Komputer)*, 5(1), 123-129.
- [3] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," pp. 130-139, 2000.

- [4] T. Haselmann, G. Thies, and G. Vossen, "Looking into a REST-based API for database-as-a-service systems," Proc. - 12th IEEE Int. Conf. Commer.

Enterp. Comput. CEC 2010, pp. 17–24, 2010, doi: 10.1109/CEC.2010.11.

