https://jurnal.pustakagalerimandiri.co.id/index.php/pustakadata
DOI : https://doi.org/10.55382/jurnalpustakadata.v5i1.1032

Jurnal Pustaka Data

Pusat Akses Kajian Database, Analisa Teknologi, dan Arsitektur Komputer

Vol. 5 No. 1 (2025) 205 – 213
E ISSN: 2809-7483

Sistem Monitoring Server dan Website CV. Technos Studio Menggunakan Laravel Scheduller dan Telegram Bot

Muhammad Ilham Arief Prabowo[1], Yushlih Narsil Shadrii[2], Bintang Alnur Ikhsan[3], Muhammad Faizal[4], Putu Linda Andaniari[5]

Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetaahuan Alam, Universitas Halu Oleo
[1]meanhills019@gmail.com. [2]yushlihofficial@gmail.com. [3]bintangalnur88@gmail.com. [4]22muh.izal@gmail.com. [5]lindaputu973@gmail.com

Abstract

In today's digital era, the continuity of website and server services is crucial for the operations of companies, especially those in the information technology sector. CV. Techno's Studio requires a monitoring system capable of real-time tracking and automatic notifications when disruptions occur. This research designs and develops a monitoring system based on the Laravel framework integrated with the Telegram Bot API. The system utilizes scheduled commands and the GuzzleHttp library to periodically check the status of websites, store monitoring results in a database, and send notifications to users when issues are detected. The implementation results demonstrate that the system can efficiently perform automated monitoring and deliver timely alerts, although some limitations remain, such as dependency on the Laravel Scheduler and the absence of retry or escalation mechanisms for failed notifications.

Keywords: Monitoring, Laravel, Telegram Bot, Website, Scheduled Command, GuzzleHttp.

Abstrak

Dalam era digital saat ini, keberlangsungan layanan *website* dan *server* sangat krusial bagi operasional perusahaan, terutama yang bergerak di bidang teknologi informasi. CV. Techno's Studio memerlukan sistem *monitoring* yang mampu melakukan pemantauan terhadap *website* dan *server* secara *real-time* serta memberikan notifikasi secara otomatis ketika terjadi gangguan. Penelitian ini merancang dan membangun sistem *monitoring* berbasis Laravel yang terintegrasi dengan Telegram Bot API. Sistem ini memanfaatkan *scheduled command* dan *library* GuzzleHttp untuk memeriksa status *website* secara berkala, menyimpan hasil *monitoring* dalam *database*, serta mengirimkan notifikasi ke pengguna jika terdeteksi masalah. Hasil implementasi menunjukkan bahwa sistem ini mampu melakukan pemantauan secara otomatis dan memberikan notifikasi secara efisien, meskipun masih terdapat beberapa keterbatasan seperti ketergantungan pada *Laravel Scheduler* dan tidak adanya mekanisme *retry* atau eskalasi notifikasi.

Kata kunci: *Monitoring, Laravel, Telegram Bot, Website, Scheduled Command, GuzzleHttp*.

Submitted : 14-06-2025 | Reviewed : 18-06-2025 | Accepted : 27-06-2025

1. Pendahuluan

Dalam era digital yang terus berkembang, kebutuhan akan sistem *monitoring* yang efektif dan *real-time* menjadi semakin penting. Sistem monitoring telah menjadi komponen vital dalam mendukung operasional berbagai sektor, mulai dari industri, pendidikan, kesehatan, hingga teknologi informasi. Sistem ini memungkinkan pengguna untuk mengawasi, menganalisis, dan mengambil tindakan berdasarkan data yang terkumpul secara terus-menerus, sehingga dapat meningkatkan efisiensi, akurasi, serta mendukung pengambilan keputusan yang tepat waktu [1].

Namun, dalam implementasinya, banyak organisasi masih menghadapi berbagai kendala dalam proses monitoring, seperti keterlambatan dalam memperoleh informasi, kurangnya visibilitas terhadap data yang relevan, serta ketergantungan terhadap metode manual yang rawan kesalahan dan tidak efisien [2]. Hal ini berdampak pada lambatnya respons terhadap masalah, peningkatan risiko kesalahan, hingga penurunan produktivitas.

Sistem *monitoring* sendiri merupakan siklus kegiatan yang mencakup proses pengumpulan data, pengamatan ulang, pelaporan, dan tindakan atas pemberitahuan dari suatu proses yang sedang berjalan [1][2]. Dalam konteks teknologi informasi, pemantauan terhadap performa *web server* menjadi sangat krusial karena *web server* berfungsi menerima permintaan dari klien (*HTTP request*) dan memberikan tanggapan berupa konten dalam bentuk halaman *web* (*HTTP response*) [3]. Gangguan seperti *downtime* dapat disebabkan oleh kerusakan perangkat keras/lunak, gangguan jaringan, lonjakan trafik, hingga serangan peretas, yang pada akhirnya menurunkan kualitas layanan digital [4][14].

Sistem informasi sendiri merupakan organisasi sistem yang menjumlahkan kebutuhan proses transaksi harian guna menunjang fungsi manajemen operasi dan strategi organisasi secara keseluruhan. Sistem ini mampu menyediakan laporan-laporan penting yang dibutuhkan untuk pengambilan keputusan [9]. Salah satu teknologi yang banyak digunakan dalam pengembangan sistem informasi adalah *framework* Laravel.

Laravel merupakan kerangka kerja pengembangan *web* berbasis PHP yang menggunakan pola arsitektur *Model-View-Controller* (MVC). Laravel dirancang untuk meningkatkan kualitas perangkat lunak dengan cara mengurangi biaya pengembangan dan pemeliharaan, serta memberikan sintaks yang ekspresif dan efisien dalam membangun aplikasi [5]. Dengan fleksibilitas tersebut, Laravel menjadi pilihan populer dalam pengembangan sistem informasi, termasuk sistem *monitoring*.

CV. Techno's Studio, sebagai perusahaan yang bergerak di bidang teknologi informasi, sangat bergantung pada ketersediaan dan keandalan sistem digitalnya. Oleh karena itu, dibutuhkan sistem monitoring yang bersifat otomatis, dapat bekerja secara *real-time*, dan mampu memberikan notifikasi secara cepat ketika terjadi gangguan. Selama ini, sistem *monitoring* konvensional yang digunakan masih memerlukan pemantauan manual dan kurang responsif terhadap kejadian tidak terduga. Hal ini menjadi celah yang dapat berdampak serius terhadap kontinuitas layanan dan kepercayaan klien.

Beberapa penelitian sebelumnya telah dilakukan untuk mengembangkan sistem *monitoring server*, seperti yang dilakukan oleh Michael et al. [6] menggunakan protokol SNMP untuk memantau *resource server* dan memberikan notifikasi status, atau Rasyidi dan Pratama [7] yang memanfaatkan Grafana dan Prometheus serta integrasi Telegram untuk *alert* otomatis. Penelitian lain oleh Rahman et al. [8] juga menggunakan kombinasi Prometheus-Grafana dengan notifikasi Telegram dalam mendeteksi kondisi layanan Apache dan MySQL.

Di sisi lain, penelitian oleh Herdiansyah et al. [11] menggunakan *framework* Laravel untuk sistem informasi *quality control* berbasis *web*, namun belum difokuskan pada *monitoring server* secara otomatis. Pemanfaatan Laravel untuk pengembangan sistem monitoring yang terintegrasi dengan notifikasi Telegram masih jarang dijadikan fokus utama. Hal serupa juga terlihat dari penelitian Maulana [12] dan Leonardo et al. [13] yang lebih banyak membahas penggunaan Telegram Bot API untuk pengiriman informasi secara umum atau akademik, tanpa integrasi *real-time* dalam konteks *monitoring server*.

Beberapa penelitian lainnya menekankan pentingnya *monitoring server* untuk menjaga performa dan ketersediaan layanan, seperti Sari et al. [10], Prasetyo et al. [15], serta Azi et al. [14], namun belum secara khusus menggabungkan Laravel dengan notifikasi Telegram untuk menciptakan sistem yang responsif dan efisien.

Penelitian ini menghadirkan kebaruan dalam bentuk pengembangan sistem *monitoring* otomatis berbasis Laravel yang diintegrasikan dengan Telegram Bot API. Sistem ini dapat melakukan pengecekan kondisi layanan server secara berkala dan memberikan notifikasi secara langsung kepada pengguna apabila terdeteksi gangguan. Pendekatan ini diharapkan dapat menjadi solusi efektif yang lebih adaptif terhadap kebutuhan layanan digital saat ini, serta mengisi celah penelitian sebelumnya yang belum menggabungkan efisiensi pengembangan Laravel dengan notifikasi real-time berbasis Telegram.

Submitted : 14-06-2025 | Reviewed : 18-06-2025 | Accepted : 27-06-2025

2. Metode Penelitian

Sistem *monitoring* ini dirancang untuk memantau status *website* secara *real-time* dengan memanfaatkan kombinasi teknologi *web-based monitoring* dan notifikasi Telegram. Sistem *monitoring* ini menggunakan pendekatan berbasis *scheduled command* di Laravel untuk memeriksa status *website* secara berkala. Metode ini dipilih karena kemudahan pengelolaan dan integrasinya dengan komponen lain.

2.1. Arsitektur Sistem

Backend dalam sistem ini menggunakan Laravel 11, sebuah *framework* PHP yang andal dan modern. Laravel 11 berperan sebagai inti dari arsitektur sistem dengan fungsi utama mengelola logika bisnis, mengatur penjadwalan tugas-tugas terotomatisasi, serta menjembatani integrasi dengan berbagai komponen lainnya seperti *database*, layanan eksternal, dan antarmuka pengguna. Dengan fitur-fitur canggih seperti *routing* yang efisien, sistem *queue*, *scheduler*, serta dukungan terhadap arsitektur berbasis *service* dan *event-driven*, Laravel 11 memungkinkan pengembangan *backend* yang terstruktur, skalabel, dan mudah dikelola.

Library GuzzleHttp digunakan dalam sistem ini untuk melakukan HTTP *request* ke *website* yang dimonitor. GuzzleHttp merupakan library HTTP client untuk PHP yang memungkinkan *backend* mengirim permintaan HTTP secara efisien dan menerima respons dari server tujuan. Dalam konteks *monitoring*, GuzzleHttp berperan penting dalam mengambil data atau status dari *website* eksternal, baik untuk keperluan pengecekan ketersediaan layanan, pengambilan konten, maupun integrasi API. Dengan dukungan terhadap *asynchronous request*, *middleware*, dan penanganan *error* yang fleksibel, GuzzleHttp membantu sistem menjalankan proses komunikasi antar server secara stabil dan cepat, sekaligus memudahkan *developer* dalam menulis dan mengelola kode yang berkaitan dengan HTTP.

Database dalam sistem ini digunakan sebagai media penyimpanan utama untuk berbagai jenis data yang diperlukan dalam operasional layanan. Data yang disimpan mencakup informasi klien, daftar *website* yang dimonitor, serta *log* hasil monitoring yang mencatat aktivitas, status, dan respon dari setiap *website* yang diawasi. Dengan struktur yang terorganisir, database memungkinkan *backend* mengakses dan memanipulasi data secara efisien, baik untuk keperluan autentikasi pengguna, pengelolaan konfigurasi pemantauan, maupun pelaporan hasil monitoring. Penyimpanan log secara historis juga memudahkan dalam analisis performa *website* dari waktu ke waktu, serta menjadi dasar untuk notifikasi atau tindakan otomatis ketika terjadi gangguan pada *website* yang dipantau.

Sistem notifikasi dalam aplikasi ini menggunakan Telegram Bot API untuk mengirim pesan otomatis kepada pengguna ketika terdeteksi bahwa suatu *website* sedang mengalami gangguan atau tidak dapat diakses (*down*). Dengan memanfaatkan Telegram Bot, sistem dapat secara langsung dan *real-time* mengirimkan informasi penting ke akun Telegram pengguna, sehingga mereka dapat segera mengetahui adanya masalah tanpa harus terus-menerus memantau sescara manual. Integrasi ini tidak hanya meningkatkan responsivitas pengguna terhadap insiden, tetapi juga memberikan pengalaman komunikasi yang cepat dan efisien melalui platform yang ringan dan mudah diakses dari berbagai perangkat.

2.2. Alur Kerja

Alur kerja sistem *monitoring* ini terdiri dari enam tahapan utama yang berjalan secara otomatis dan berkesinambungan. Pertama, proses dimulai ketika Laravel *Scheduler* menjalankan *command* pada interval waktu yang telah ditentukan sebelumnya, sehingga sistem dapat melakukan pemantauan secara berkala tanpa intervensi manual. Kedua, *command* tersebut akan mengambil data dari *database*, khususnya informasi mengenai *client* dan daftar *website* yang berstatus aktif (*is_active = true*). Ketiga, setiap URL *website* yang terdaftar kemudian diperiksa dengan mengirimkan HTTP *request* menggunakan GuzzleHttp, untuk memastikan bahwa situs tersebut dapat diakses dengan benar. Keempat, hasil dari proses pemeriksaan ini, termasuk status kode respons, akan dicatat ke dalam tabel *monitoring_logs* di *database* sebagai arsip *log* monitoring. Kelima, jika status *website* menunjukkan adanya masalah (misalnya status code ≥ 400), sistem secara otomatis akan mengirimkan notifikasi kepada pengguna melalui Telegram Bot API, berdasarkan token dan *chat* ID yang telah dikonfigurasi. Terakhir, sistem akan memperbarui informasi mengenai waktu pemeriksaan dan notifikasi terakhir di *database*, agar data tetap sinkron dan dapat dijadikan acuan pada monitoring berikutnya. Dengan alur ini, sistem dapat memberikan pemantauan yang efisien, responsif, dan terdokumentasi dengan baik.

2.3. Metode Sistem

Tabel Sistem *monitoring* ini menggunakan pendekatan berbasis *scheduled command* di Laravel untuk memeriksa status *website* secara berkala. Metode ini dipilih karena kemudahan pengelolaan dan integrasinya dengan komponen lain.

Pada tahap pengambilan data, sistem akan menjalankan command yang bertugas mengambil informasi dari database mengenai *client* dan *website* yang memiliki status aktif, yaitu dengan nilai *is_active = true*. Langkah ini bertujuan untuk memastikan bahwa hanya webisite yang masih aktif dan terdaftar secara valid yang akan dimonitor oleh

sistem. Dengan menyaring data berdasarkan status aktif, sistem dapat bekerja secara efisien, menghindari pemrosesan yang tidak perlu terhadap entitas yang sudah tidak digunakan atau dinonaktifkan. Pendekatan ini juga menjaga keakuratan proses *monitoring* dan memastikan bahwa sumber daya sistem difokuskan pada pemantauan layanan yang benar-benar masih relevan bagi pengguna.

Pada tahap pemeriksaan status *website*, *command* yang dijalankan akan mengirimkan HTTP *request* ke URL masing-masing *website* yang dimonitor dengan menggunakan *library* GuzzleHttp. *Libirary* ini memungkinkan sistem melakukan komunikasi langsung dengan *website* target untuk memperoleh respons dari *server*. Respons tersebut kemudian dianalisis berdasarkan status code yang diterima. Jika status code menunjukkan nilai dalam rentang 200 hingga 399, maka *website* dianggap berfungsi dengan baik. Namun, jika status code berada pada angka 400 atau lebih, hal ini menandakan adanya masalah pada *website*, seperti kesalahan klien (4xx) atau kesalahan server (5xx). Tahap ini menjadi inti dari proses monitoring karena menentukan kondisi terkini dari setiap website yang diawasi secara otomatis dan berkala.

Pada tahap pencatatan *log monitoring*, setiap hasil dari pemeriksaan *website* akan direkam secara sistematis ke dalam tabel *monitoring_logs* di *database*. Setiap entri *log* mencakup informasi penting seperti waktu pemeriksaan, status *code* yang diterima dari *website*, serta pesan *error* jika terjadi kegagalan dalam respons atau koneksi. Pencatatan ini berfungsi sebagai arsip historis yang dapat digunakan untuk melakukan pelacakan performa *website* dari waktu ke waktu, analisis gangguan, dan pembuatan laporan. Dengan menyimpan data ini secara rutin, sistem memungkinkan administrator atau pengguna untuk mengevaluasi stabilitas dan keandalan layanan website yang dimonitor secara lebih mendalam dan terukur.

Pada tahap pengiriman notifikasi, sistem akan secara otomatis mengirimkan pesan ke Telegram jika terdeteksi bahwa sebuah website dalam kondisi *down*, ditandai dengan status code ≥ 400. Proses ini dilakukan melalui Telegram Bot API, dengan menggunakan bot token dan *chat* ID yang telah dikonfigurasi sebelumnya untuk setiap pengguna. Notifikasi yang dikirim berisi informasi detail mengenai *website* yang mengalami gangguan, seperti nama *website*, URL, status *code* yang diterima, waktu kejadian, dan pesan *error* jika ada. Dengan adanya notifikasi *real-time* ini, pengguna dapat segera mengetahui adanya masalah dan mengambil tindakan yang diperlukan tanpa harus terus memantau sistem secara manual. Fitur ini meningkatkan kecepatan respon terhadap insiden dan memastikan keberlangsungan layanan digital yang dimonitor.

3. Hasil dan Pembahasan

3.1. Identifikasi Masalah

Pada sistem *monitoring* yang dibuat ini menujukkan beberapa kekurangan yang kami sadari. Sistem *monitoring* ini memiliki beberapa keterbatasan penting yang perlu diperhatikan untuk memastikan keandalannya. Pertama, terdapat ketergantungan penuh pada Laravel *Scheduler*; artinya, jika *scheduler* tidak aktif atau gagal dijalankan (misalnya karena kesalahan konfigurasi *cron job*), maka seluruh proses *monitoring* termasuk pemeriksaan *website* dan pengiriman notifikasi tidak akan berjalan. Kedua, sistem tidak memiliki mekanisme *retry* saat terjadi kegagalan dalam pengiriman notifikasi, misalnya karena masalah koneksi internet atau *error* pada API Telegram. Akibatnya, informasi penting bisa hilang tanpa ada usaha sistem untuk mengulang proses pengiriman. Ketiga, sistem tidak menerapkan mekanisme eskalasi, yaitu notifikasi hanya dikirim satu kali sesuai interval waktu yang ditentukan, tanpa pengulangan atau peningkatan urgensi jika masalah belum terselesaikan. Hal ini dapat menyebabkan penurunan kewaspadaan pengguna terhadap insiden yang berkepanjangan.

3.2. Analisis Code

Pada bagian ini, dilakukan analisis terhadap kode program yang digunakan dalam sistem monitoring website. Analisis ini bertujuan untuk memahami struktur, logika, dan fungsionalitas dari setiap bagian kode yang dibangun.

Method handle() dalam *command* Laravel berfungsi sebagai titik awal eksekusi proses *monitoring website* secara otomatis. Proses ini dimulai dengan mengambil data *client* dan *website* yang memiliki status aktif (*is_active = true*) dari *database*. Jika tidak ditemukan data yang sesuai, sistem akan mencatat *log* bahwa tidak ada data yang diproses, lalu menghentikan eksekusi *command* agar tidak membuang sumber daya secara sia-sia. Selanjutnya, *method* ini akan membuat *instance* dari HttpClient menggunakan *library* GuzzleHttp, dengan pengaturan *timeout* selama 10 detik untuk menghindari proses yang menggantung terlalu lama ketika *website* tidak merespons. Setelah itu, sistem akan melakukan iterasi terhadap setiap *client* dan *website* aktif, dan untuk setiap pasangan data tersebut, akan memanggil *method* checkAndLogWebsite(). *Method* ini bertanggung jawab untuk memeriksa status *website*, mencatat *log* hasilnya, dan mengirim notifikasi jika diperlukan. Dengan struktur ini, *handle*() memastikan proses monitoring berjalan efisien, terstruktur, dan dapat menangani kegagalan dengan baik.

Method checkAndLogWebsite() bertanggung jawab untuk menangani proses pemeriksaan dan pencatatan status *website* secara menyeluruh. Proses dimulai

dengan menghitung waktu respon *website* menggunakan fungsi *microtime*(), yang memungkinkan sistem mengukur durasi permintaan secara presisi hingga satuan mikrodetik. Setelah itu, method ini akan memanggil *checkWebsite*() untuk melakukan HTTP *request* ke URL website dan mengambil status *code* sebagai indikator kondisi *website*. Hasil pemeriksaan tersebut kemudian dicatat ke dalam tabel *monitoring_logs*, termasuk informasi seperti waktu pemeriksaan, status *code*, pesan *error* (jika ada), dan durasi respon. Selain itu, *method* ini juga akan memperbarui data *website* di *database*, khususnya waktu pemeriksaan terakhir, agar sistem memiliki referensi terbaru untuk monitoring selanjutnya. Jika hasil pemeriksaan menunjukkan bahwa website sedang *down* (*status code* ≥ 400), maka *method* akan secara otomatis memanggil *sendNotification*() untuk mengirimkan notifikasi ke pengguna melalui Telegram. Dengan alur kerja ini, *checkAndLogWebsite*() memastikan bahwa setiap *website* yang dimonitor dianalisis secara menyeluruh dan bahwa pengguna mendapatkan informasi penting secara tepat waktu.

Method checkWebsite() berfungsi untuk melakukan pengecekan langsung terhadap ketersediaan dan respon sebuah *website* melalui *request* HTTP *GET* yang dikirim menggunakan *library* GuzzleHttp. Saat mengirim permintaan ke URL yang dituju, *method* ini dilengkapi dengan mekanisme penanganan berbagai jenis *exception*, seperti *RequestException*, *ConnectException*, dan *exception* lainnya yang mungkin muncul akibat masalah jaringan, DNS, atau *server* tidak merespons. Hal ini penting untuk menjaga agar proses *monitoring* tidak terganggu oleh errior tak terduga dan tetap dapat melanjutkan pemeriksaan untuk *website* lainnya. Setelah menerima respons, *method* akan mengembalikan nilai *true* jika status *code* yang diterima adalah 200, yang menunjukkan bahwa *website* berfungsi dengan normal. Sebaliknya, jika status *code* berbeda dari 200 atau terjadi *exception*, maka *method* akan mengembalikan *false*, yang menandakan bahwa *website* dalam kondisi bermasalah atau tidak dapat diakses. Dengan pendekatan ini, *checkWebsite*() menjadi komponen inti dalam menentukan apakah *website* dalam keadaan aktif atau down.

Method sendNotification() bertugas untuk mengelola dan mengirimkan notifikasi ke Telegram ketika sebuah *website* terdeteksi mengalami gangguan. Proses diawali dengan memeriksa apakah *client* memiliki *bot token* dan *chat* ID yang valid, karena informasi ini diperlukan untuk mengirim pesan melalui Telegram Bot API. Selanjutnya, sistem akan memeriksa apakah notifikasi memang perlu dikirim, dengan membandingkan waktu notifikasi terakhir dengan interval yang telah ditentukan untuk mencegah pengiriman pesan yang terlalu sering.

Jika semua kondisi terpenuhi, *method* akan mengambil 5 *log downtime* terakhir dari database terkait *website* yang bersangkutan untuk memberikan konteks historis. Informasi tersebut kemudian digunakan untuk memformat pesan notifikasi, yang mencakup nama *website*, URL, status *down* saat ini, serta riwayat *downtime* sebelumnya. Setelah pesan disusun, sistem akan mengirimkan notifikasi ke Telegram dengan memanggil *method sendTelegramMessage*(). Terakhir, *method* akan memperbarui waktu notifikasi terakhir untuk *website* tersebut di database, agar referensi waktu untuk pengiriman notifikasi berikutnya tetap akurat. Dengan alur ini, *sendNotification*() memastikan notifikasi dikirim secara informatif, tepat sasaran, dan tidak berlebihan.

Method formatDowntimeMessage() bertugas untuk mengubah data *log downtime* yang bersifat teknis menjadi pesan yang lebih informatif dan mudah dipahami oleh pengguna. Proses ini dimulai dengan memformat *log downtime*, yaitu data hasil monitoring yang mencatat waktu dan status *website* ketika mengalami gangguan. *Method* ini kemudian mengelompokkan *log* berdasarkan tanggal, sehingga pengguna dapat melihat kapan saja *website* mengalami masalah, dengan urutan yang lebih terstruktur. Selain itu, untuk meningkatkan keterbacaan dan pemahaman, *method* juga menambahkan deskripsi status *code*, seperti "404 *Not Found*" atau "500 *Internal Server Error*", sehingga pengguna tidak hanya melihat angka status, tetapi juga mengetahui arti dari setiap kode tersebut. Dengan pendekatan ini, *formatDowntimeMessage*() memastikan bahwa pesan notifikasi yang dikirim melalui Telegram bersifat jelas, terorganisir, dan membantu pengguna dalam menganalisis masalah yang terjadi pada website mereka.

Method sendTelegramMessage() bertugas untuk mengirim pesan notifikasi ke Telegram dengan memanfaatkan Telegram *Bot* API. *Method* ini akan melakukan HTTP *POST* ke *endpoint* API Telegram dengan menyertakan *bot* token, *chat* ID, dan pesan yang telah diformat sebelumnya sebagai parameter. Tujuan utamanya adalah memastikan bahwa pengguna menerima informasi tentang status *website* mereka secara *real-time*.

Selama proses pengiriman, *method* ini juga dilengkapi dengan mekanisme penanganan *exception*. Jika terjadi kesalahan seperti masalah jaringan, kesalahan konfigurasi token, atau chat ID tidak valid *method* ini akan menangkap *exception* tersebut dan mencatatnya ke dalam *log* sistem. Hal ini penting untuk mencegah kegagalan sistem secara keseluruhan dan untuk memudahkan proses *debugging*. Dengan demikian, *sendTelegramMessage*() memastikan bahwa notifikasi dikirim secara andal dan sistem tetap stabil meskipun menghadapi kendala teknis.

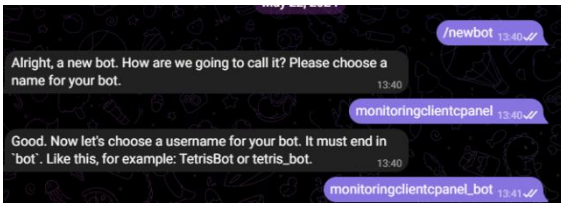
Submitted : 14-06-2025 | Reviewed : 18-06-2025 | Accepted : 27-06-2025

Method shouldNotify() berfungsi untuk menentukan apakah notifikasi perlu dikirim ke pengguna dengan cara membandingkan waktu saat ini dengan waktu notifikasi terakhir yang tercatat di database. Proses ini mengacu pada interval waktu minimum yang telah ditentukan, misalnya setiap 30 menit atau 1 jam sekali, untuk mencegah pengiriman notifikasi yang berlebihan atau spam.

Jika selisih antara waktu sekarang dan waktu notifikasi terakhir lebih besar atau sama dengan interval yang ditetapkan, maka method akan mengembalikan nilai true, yang berarti notifikasi dapat dikirim. Sebaliknya, jika selisih waktunya masih terlalu dekat, *method* akan mengembalikan false, menandakan bahwa notifikasi belum perlu dikirim ulang. Dengan cara ini, *shouldNotify*() membantu menjaga kedisiplinan sistem dalam mengelola frekuensi notifikasi, sehingga pengguna hanya menerima informasi yang benar-benar relevan dan tidak terganggu oleh pemberitahuan berulang dalam waktu singkat.

3.3. Penggunaan Sistem

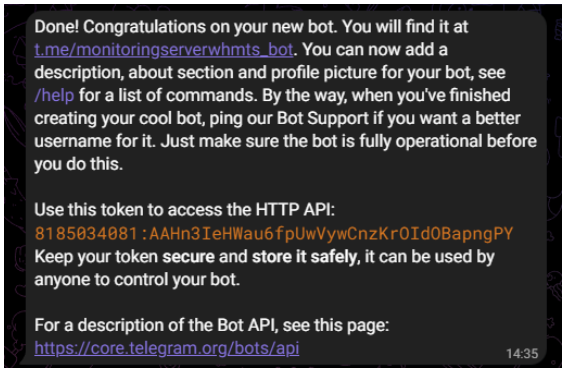
Dalam menggunakan sistem *monitoring* ini akan dilakukan melalui beberapa langkah mulai dari membuat *bot* yang akan digunakan mengirimkan *request* ke *web* yang dipantau, hingga melakukan penyetelan.



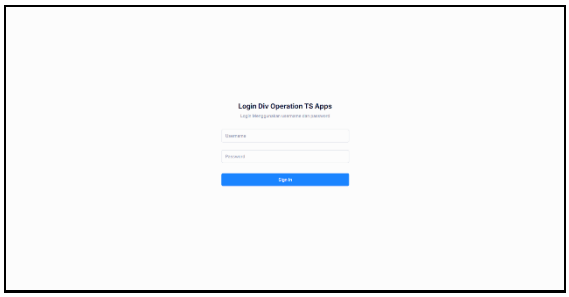
Gambar 1. Menggunakan BotFather.

Untuk menggunakan sistem pertama perlu membuat *bot* dengan menggunakan *BotFather*, untuk membuat bot Telegram yang digunakan dalam sistem *monitoring*, langkah pertama adalah menggunakan layanan *BotFather*, yaitu bot resmi dari Telegram yang digunakan untuk membuat dan mengelola bot lain. Anda dapat memulainya dengan mencari *@BotFather* di aplikasi Telegram, kemudian memulai percakapan dengannya dan mengirim perintah /*newbot*. *BotFather* akan meminta Anda untuk menentukan nama dan username unik untuk bot yang ingin dibuat. Setelah berhasil, *BotFather* akan memberikan *bot* token, yaitu kode autentikasi unik yang memungkinkan sistem Anda mengakses dan berkomunikasi dengan API Telegram atas nama *bot* tersebut.

Bot token ini sangat penting, karena akan digunakan oleh sistem monitoring untuk mengirim notifikasi ke pengguna melalui Telegram. Oleh karena itu, pastikan untuk menyimpan *bot* token tersebut dengan aman, dan kemudian masukkan token tersebut ke dalam konfigurasi sistem monitoring, baik melalui *database*, *file .env*, atau panel pengaturan yang disediakan. Tanpa *bot* token yang valid, sistem tidak akan dapat mengirimkan notifikasi Telegram.

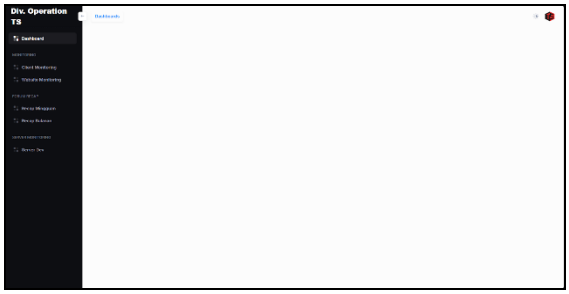


Gambar 2. Notifikasi Bot Token.

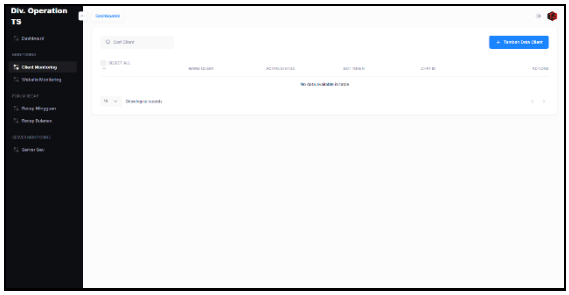


Gambar 3. Tampilan Login Sistem Monitoring.

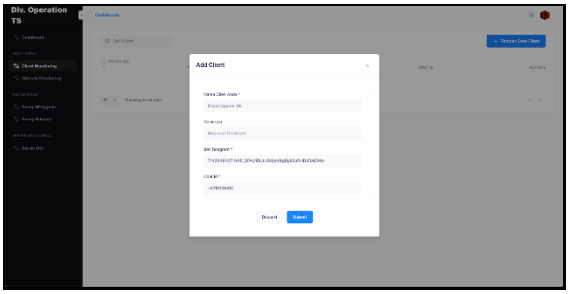
Sistem *monitoring* dapat diakses melalui *url* yang telah ditentukan dan melakukan login dengan *username* dan *password*. Lalu dengan mengakses menu *Client Monitoring* maka *bot* baru yang telah dibuat tadi dapat ditambahkan.



Gambar 4. Tampilan Dashboard Sistem Monitoring.



Gambar 5. Penambahan Bot di Sistem Monitoring.

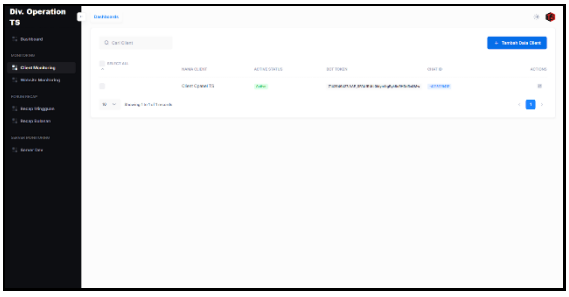


Gambar 6. Menambahkan Data Bot Pada Sistem Monitoring.

Setelah pengguna berhasil *login* ke sistem, langkah selanjutnya untuk mengintegrasikan bot Telegram adalah dengan masuk ke menu '*Client Monitoring'*. Di dalam menu ini, pengguna dapat menambahkan data *bot* Telegram yang akan digunakan untuk mengirim notifikasi. Caranya adalah dengan mengklik tombol "Tambah Data Client", yang akan memunculkan *popup form.*

Form ini berisi beberapa isian penting yang harus dilengkapi seperti, Nama *Client* untuk memberi identitas pada client atau pengguna yang dimonitor. *Chat* ID yang merupakan ID akun atau grup Telegram yang akan menerima notifikasi. Serta *Bot* Token yang merupakan token unik yang diberikan oleh *BotFather* saat membuat *bot*.

Setelah semua kolom terisi dengan benar, pengguna dapat menekan tombol "*Submit*" untuk menyimpan data. Jika proses berhasil, maka data *client* tersebut akan langsung muncul di tabel *client monitoring*, dan sistem akan mulai menggunakan informasi tersebut untuk mengirim notifikasi ketika *website* milik *client* mengalami gangguan. Proses ini memastikan bahwa setiap *client* memiliki konfigurasi *bot* Telegram yang spesifik dan siap digunakan oleh sistem *monitoring* secara otomatis.

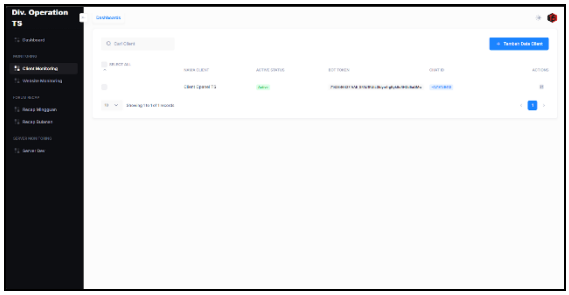


Gambar 7. Tampilan Sistem Monitoring Dengan Data Bot.

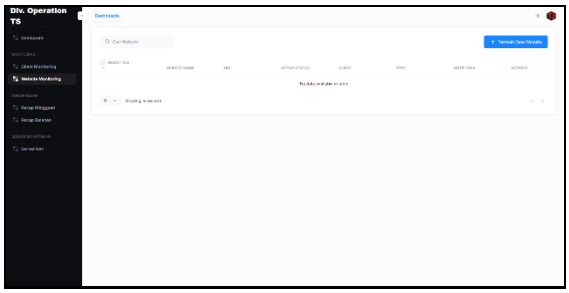
Untuk menambahkan data *website* yang ingin dimonitor oleh sistem, pengguna dapat masuk ke menu "*Website Monitoring*" setelah *login*. Di dalam menu ini, terdapat tombol "Tambah Data *Website*" yang digunakan untuk menambahkan informasi *website* baru ke dalam sistem.

Setelah tombol tersebut diklik, akan muncul *form input* yang harus diisi dengan data-data penting seperti, Nama *Website* untuk memberi label atau identitas pada *website*. *URL Website* berupa alamat lengkap website yang akan dimonitor (termasuk http:// atau https://). *Client* Terkait, pada bagian ini memilih *client* yang sudah terdaftar dan akan menerima notifikasi. Status Aktif yang menandai apakah *website* tersebut sedang dalam status monitoring aktif.

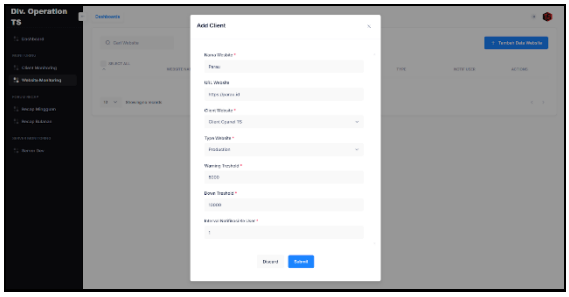
Setelah semua kolom diisi dengan benar, pengguna cukup menekan tombol "*Submit*". Jika proses penyimpanan berhasil, data *website* akan langsung muncul di dalam tabel *monitoring*, dan sistem akan mulai menjadwalkan pemeriksaan otomatis terhadap *website* tersebut sesuai konfigurasi yang telah ditetapkan.



Gambar 8. Penambahan Data Website di Sistem Monitoring.



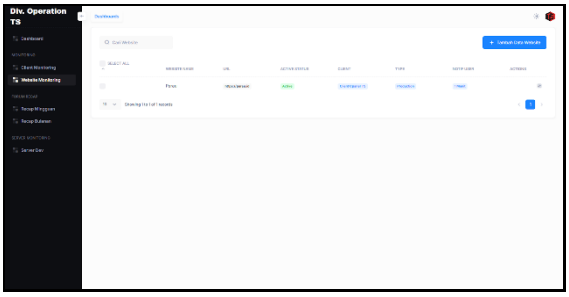
Gambar 9. Tampilan Data Website di Sistem Monitoring.



Gambar 10. Menambahkan Data Website Pada Sistem Monitoring.

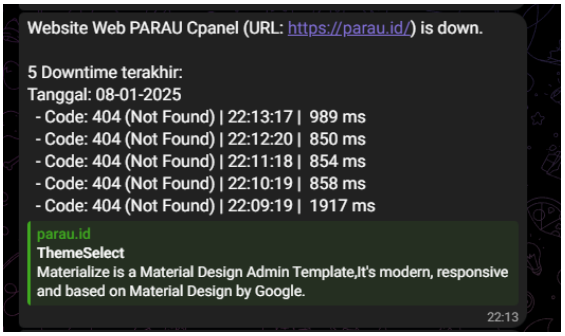
Jika proses penambahan data *website* berhasil, maka informasi *website* tersebut akan langsung ditampilkan dalam tabel di menu *Website Monitoring*. *Website* yang sudah masuk ke tabel ini secara otomatis akan mulai dimonitor oleh sistem sesuai jadwal yang ditentukan oleh Laravel *Scheduler*.

Submitted : 14-06-2025 | Reviewed : 18-06-2025 | Accepted : 27-06-2025



Gambar 11. Tampilan Sistem Monitoring Dengan Data Website.

Apabila suatu saat *website* yang dimonitor mengalami gangguan (*down*) misalnya tidak dapat diakses atau merespons dengan status *code* ≥ 400 maka sistem akan segera mencatat *log* kejadian tersebut ke dalam *database* dan secara otomatis mengirimkan notifikasi ke Telegram melalui *bot* yang telah dikonfigurasi sebelumnya. Notifikasi ini berisi informasi penting seperti nama *website*, *URL*, status *error*, serta waktu kejadian, sehingga *client* dapat segera mengetahui dan menangani gangguan yang terjadi. Dengan alur ini, sistem monitoring dapat memberikan respon cepat dan *real-time* terhadap gangguan layanan *website*.



Gambar 12. Pesan Notifikasi Down Website.

4. Kesimpulan

Sistem *monitoring* yang dikembangkan dalam penelitian ini berhasil memanfaatkan Laravel *Scheduler* dan Telegram *Bot* API untuk melakukan pemantauan *website* secara berkala dan memberikan notifikasi otomatis saat terjadi gangguan. Sistem ini mendukung pencatatan *log* status *website* dan menyediakan tampilan yang *user-friendly* melalui *dashboard* web. Hasil pengujian menunjukkan bahwa sistem mampu memberikan informasi gangguan secara cepat dan akurat, sehingga pengguna dapat segera melakukan penanganan.

Namun, terdapat beberapa kelemahan yang perlu diperhatikan, seperti ketergantungan terhadap Laravel *Scheduler*, belum adanya mekanisme *retry* pada notifikasi yang gagal, serta ketiadaan sistem eskalasi jika masalah tidak segera terselesaikan. Oleh karena itu, pengembangan lebih lanjut diperlukan untuk meningkatkan keandalan dan skalabilitas sistem, termasuk penambahan fitur notifikasi berulang dan pemantauan komponen server secara menyeluruh.

Daftar Rujukan

[1] R. B. Pradana, and S. A. Arnomo, "Rancang Bangun Sistem Monitoring Kualitas Udara Berbasis Android," *Jurnal Comasie (Computer & Science Industrial Engineering)*, vol. 9, no. 2, pp. 200 – 207, 2023.

[2] A. B. Wibowo, and R. D. Laksono, "Sistem Monitoring Penggunaan Air Sawah Menggunakan Arduino Uno," *Jurnal ELECTRA : Electrical Engineering Articles*, vol. 4, no. 2, pp. 28-35, 2024.

[3] D. Kartika, Riska, and Y. Mardiana, "Simulasi DNS Server Dan Web Server Dengan Sistem Operasi Debian, pada Jaringan Lokal Area Network," *Jurnal Media Komputer Science*, vol. 2, no. 1, pp. 83-92, 2023.

[4] W. Aldiwidianto, and I. G. L. E. Prismana, "Analisis Perbandingan High Availibility Pada Web Server Menggunakan Orchestration Tool Kubernetes Dan Docker Swarm," *JINACS : Journal of Informatics and Computer Science*, vol. 5, no. 1, pp. 138-148, 2023.

[5] M. Nugraha, L. Sakinah, R. A. Setiawan, H. Mulyani, "Rancang Bangun Sistem Informasi Penerimaan Mahasiswa Baru Berbasis Web Dengan Menggunakan Framework Laravel," *JITET (Jurnal Informatika dan Teknik Elektro Terapan)*, vol. 12, no. 2, pp. 1162-1169, 2024.

[6] A. Michael, H. Hermawan, and H. I. Pratiwi, "Sistem *Monitoring Server* Dengan Menggunakan SNMP," *Widyakala Journal*, 6(2), pp. 163-166, Sep. 2019, doi: https://doi.org/10.36262/widyakala.v6i2.218.

[7] B. Rasyidi, and F. Pratama, "Sistem *Monitoring Server* di PT. XYZ Media Indonesia Berbasis Grafana dan Prometheus," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 4, pp. 1456-1465, Oct. 2024, doi: https://doi.org/10.57152/malcom.v4i4.1546.

[8] D. Rahman, H. Amnur, and I. Rahmayuni, "Monitoring Server dengan Prometheus dan Grafana serta Notifikasi Telegram," *Jitsi: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 4, pp. 133-138, Des. 2020.

[9] A. S. Yuanda, "Perancangan Sistem Informasi Penjadwalan Imam Sholat di Masjid Berbasis Web Menggunakan Framework Laravel," *JCBD: Journal of Computers and Digital Business*, vol. 2, no. 2, pp. 42-48, May. 2023, doi: 10.56427/jcbd.v2i2.73.

[10] L. O. Sari, H. A. Suri, E. Safriani, and F. Jalil, "Rancang Bangunn Sistem Monitoring Bandwidth Server pada PT. Industri Kreatif Digital," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 3, no. 2, pp. 168-179, Oct. 2023, doi: https://doi.org/10.57152/malcom.v3i2.914.

[11] A. Herdiansah, R. I. Broman, and S. Maylinda, "Sistem Informasi *Monitoring dan Reporting Quality Control* Proses *Laminating* Berbasis *Web Framework* Laravel," *Jurnal TEKNO KOMPAK*, vol. 15, no. 2, pp. 13-24, 2021.

[12] M. I. Maulana, "Implementasi Bot Telegram pada Proses *Retrieval* Data dalam *Database*," *Indonesian Journal of Data Science (IJODAS)*, vol. 2, no. 1, pp. 13-20, Mar. 2021.

[13] G. C. Leonardo, Herianto, and Y. Iraawan, "Pemanfaatan Bot Telegram Sebagai Media Infromasi Akademik di STMIK Hang Tuah Pekanbaru," *JTIM: Jurnal Teknologi Informasi dan Multimedia*, vol. 1, no. 4, pp. 351-357, Feb. 2020.

[14] M. N. A. Azi, B. Arifwidodo, and E. Wahyudi, "Analisis Performansi Web Server Saat Menangani Permintaan *Client* Menggunakan Metode *Reserve Proxy Caching Nginx* dan *Varnish*," *JOURNAL OF TELECOMUNICATION, ELECTRONICS, AND CONTROL ENGINEERING (JTECE)*, vol. 5, no. 1, pp. 14-21, Jan. 2023, doi: 10.20895/JTECE.V5I1.843.

[15] E. P. Prasetyo, J. D. Irawan, and F. Ariwibisono, "RANCANG BANGUN SISTEM MONITORING SERVER VIRTUAL BERBASIS WEB MENGGUNAKAN SCRIPT MONITORING PADA PROXMOX VIRTUAL ENVIROMENT," *JATI(Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 1, pp. 179-185, Feb. 2022.

Submitted : 14-06-2025 | Reviewed : 18-06-2025 | Accepted : 27-06-2025