Perancangan dan Implementasi Sistem Backend untuk Pemantauan Data Measurement While Drilling (MWD) Secara Waktu Nyata Menggunakan Node.js, Express.js, dan PostgreSQL

Maulana Hafiz
Program Studi Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
maruworks@student.telkomuniversity.ac.id

Andrew Brian Osmond
Program Studi Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
abosmond@telkomuniversity.ac.id

Reza Rendian Septiawan
Program Studi Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
zaseptiawan@telkomuniversity.ac.id

***Abstrak*— Teknologi *Measurement While Drilling* (MWD) memungkinkan parameter pengeboran bawah permukaan dikirimkan dari dasar sumur ke permukaan secara *real-time*, sehingga mendukung pengambilan keputusan operasional yang cepat dan akurat. Penelitian ini menyajikan perancangan dan implementasi subsistem *backend* pada sistem pemantauan data MWD berbasis *web* untuk meningkatkan kontinuitas, keandalan, dan integrasi data pengeboran. *Backend* dikembangkan menggunakan *Node.js*, *Express.js*, *TypeScript*, *Prisma ORM*, dan *PostgreSQL* sebagai pusat pemrosesan data. Sistem menerima data dari perangkat *gateway*, memvalidasi dan mengurai data berformat *WITS*, menyinkronkan *timestamp* berdasarkan kedalaman pengeboran, menyimpan data ke basis data, serta mendistribusikannya ke aplikasi klien melalui *REST API* dan *WebSocket*. *Backend* juga mengimplementasikan autentikasi berbasis *JWT* dengan kontrol akses berbasis peran untuk *administrator*, *engineer*, dan *operator*, serta menyediakan fitur ekspor data dalam format *CSV*, *JSON*, *LAS*, dan *PDF plot*. Hasil pengujian menunjukkan bahwa sistem mempertahankan integritas 1.000 data uji tanpa duplikasi maupun kehilangan *field*, seluruh 15 skenario pengujian fungsional dan kontrol akses berhasil dilewati, serta *REST API* mencatat latensi persentil ke-95 di bawah 42 ms pada beban dasar dan di bawah 34 ms pada pengujian hingga 10 pengguna virtual konkuren dengan tingkat kesalahan 0%. Hasil tersebut menunjukkan bahwa *backend* yang diusulkan mampu mendukung pemantauan MWD *real-time* secara andal pada lingkungan *local deployment*.**

***Kata kunci*— *Measurement While Drilling*, backend, REST API, WebSocket, PostgreSQL, Node.js**

I. PENDAHULUAN

Dalam operasi pengeboran minyak dan gas, ketersediaan informasi kondisi bawah permukaan secara cepat dan akurat menjadi faktor penting dalam mendukung pengambilan keputusan operasional. Measurement While Drilling (MWD) merupakan salah satu teknologi yang digunakan untuk memperoleh informasi tersebut dengan mengirimkan parameter pengeboran dari dasar sumur ke permukaan selama proses pengeboran berlangsung. Berdasarkan identifikasi kebutuhan yang dilakukan bersama mitra, terdapat empat kebutuhan utama dalam pengembangan sistem pemantauan, yaitu menjaga kontinuitas pembaruan data, mengurangi ketergantungan perangkat lunak sisi klien pada lingkungan Linux, menyediakan pencatatan waktu otomatis (timestamp) yang dikaitkan dengan kedalaman pengeboran, serta menyajikan informasi pemantauan secara terintegrasi.

Untuk memenuhi kebutuhan tersebut, sistem pemantauan MWD real-time berbasis web dirancang dan dikembangkan dengan model penerapan lokal (local deployment). Sistem ini terdiri atas gateway penerima data, layanan backend, basis data, dan antarmuka pengguna berbasis web. Penelitian ini berfokus pada perancangan dan implementasi subsistem backend sebagai pusat pemrosesan data dalam sistem. Backend bertugas menerima data MWD/WITS dari gateway, memvalidasi dan mengurai data yang diterima, menyinkronkannya dengan informasi timestamp dan kedalaman pengeboran, menyimpannya ke basis data, serta mendistribusikannya ke aplikasi frontend secara real-time melalui WebSocket. Selain itu, backend juga menyediakan layanan REST API untuk autentikasi, pengambilan data historis, dan ekspor data. Melalui pengembangan tersebut, penelitian ini bertujuan menghasilkan backend yang mampu mendukung pemantauan data MWD secara andal, terintegrasi, dan real-time pada lingkungan local deployment.

II. KAJIAN TEORI

A. Measurement While Drilling (MWD)

Measurement While Drilling (MWD) merupakan teknologi yang digunakan pada industri pengeboran minyak dan gas untuk memperoleh data kondisi bawah permukaan secara real-time selama proses pengeboran berlangsung. Sistem MWD memungkinkan berbagai parameter pengeboran, seperti kedalaman, inklinasi, azimut, tekanan, temperatur, dan parameter lainnya dikirimkan dari dasar

sumur menuju permukaan tanpa menghentikan proses pengeboran. Informasi tersebut digunakan oleh operator untuk memantau kondisi pengeboran serta mendukung pengambilan keputusan secara cepat dan akurat.

Data hasil pengukuran umumnya dikirim menggunakan standar Wellsite Information Transfer Specification (WITS). Standar ini mendefinisikan format pertukaran data sehingga perangkat dari berbagai produsen dapat saling berkomunikasi menggunakan struktur data yang seragam.

B. Backend

Backend merupakan bagian dari sistem yang bertanggung jawab menangani logika aplikasi, pemrosesan data, komunikasi dengan basis data, serta penyediaan layanan kepada aplikasi klien. Pada sistem pemantauan MWD, backend berfungsi sebagai pusat pemrosesan yang menerima data dari gateway, melakukan validasi, penyimpanan, sinkronisasi timestamp, serta mendistribusikan data kepada aplikasi frontend.

Selain sebagai pusat pemrosesan, backend juga bertanggung jawab menjaga integritas data, menyediakan layanan autentikasi pengguna, serta mengelola komunikasi antara perangkat akuisisi data dengan aplikasi pemantauan.

C. REST API

REST API merupakan arsitektur layanan berbasis HTTP yang digunakan untuk memungkinkan komunikasi antara aplikasi klien dengan server. Setiap layanan diakses menggunakan endpoint tertentu dengan metode HTTP seperti GET, POST, PUT, dan DELETE.

Pada penelitian ini, REST API digunakan untuk proses autentikasi pengguna, pengelolaan sesi pengeboran, pengambilan data historis, konfigurasi sistem, serta ekspor data.

D. WebSocket

WebSocket merupakan protokol komunikasi yang memungkinkan pertukaran data dua arah secara terus-menerus (full duplex) melalui satu koneksi TCP. Berbeda dengan HTTP yang menggunakan mekanisme request-response, WebSocket memungkinkan server maupun klien mengirimkan data kapan saja tanpa perlu membuat koneksi baru.

Penggunaan WebSocket pada sistem MWD memungkinkan data pengeboran ditampilkan secara real-time pada dashboard tanpa memerlukan proses penyegaran (refresh) halaman.

E. PostgreSQL

PostgreSQL merupakan sistem manajemen basis data relasional (Relational Database Management System) yang bersifat open source. PostgreSQL mendukung transaksi ACID, integritas data, serta kemampuan pemrosesan data dalam jumlah besar.

Pada penelitian ini PostgreSQL digunakan sebagai media penyimpanan seluruh data hasil pengukuran MWD, data pengguna, konfigurasi sistem, status koneksi, serta riwayat failover.

F. JSON Web Token (JWT)

JSON Web Token (JWT) merupakan standar autentikasi berbasis token yang digunakan untuk mengidentifikasi pengguna setelah proses login berhasil dilakukan. JWT terdiri atas tiga bagian yaitu header, payload, dan signature.

Pada sistem yang dikembangkan, JWT digunakan untuk mengamankan akses REST API. Selain itu diterapkan Role-Based Access Control (RBAC) sehingga setiap pengguna memperoleh hak akses sesuai perannya sebagai administrator, engineer, ataupun operator.

III. METODE

A. Gambaran Umum Sistem

Sistem pemantauan MWD yang dikembangkan terdiri atas empat komponen utama, yaitu perangkat *gateway* sebagai penerima data dari perangkat akuisisi, layanan *backend*, basis data PostgreSQL, dan aplikasi *frontend* berbasis *web*. Data MWD diterima *backend* melalui beberapa jalur komunikasi, yaitu *HTTP Gateway*, *REST API*, *Serial Gateway*, serta *WebSocket* dari perangkat ESP32.

Komunikasi antara *backend* dan *frontend* menggunakan kombinasi *REST API* dan *WebSocket* [1]. *REST API* digunakan untuk proses autentikasi, pengambilan data historis, konfigurasi sistem, serta ekspor data, sedangkan *WebSocket* digunakan untuk distribusi data MWD secara *real-time* .

B. Arsitektur Backend

Backend dikembangkan menggunakan *Node.js*, *Express.js*, *TypeScript*, *Prisma ORM*, dan PostgreSQL [2]. Arsitektur ini dirancang sebagai pusat pemrosesan data yang bertugas menerima data dari *gateway*, melakukan validasi, sinkronisasi *timestamp*, penyimpanan ke basis data, serta mendistribusikan data kepada *frontend*.

Seluruh proses *backend* dirancang untuk memenuhi kebutuhan fungsional sistem seperti penerimaan data, penyimpanan data, komunikasi *real-time*, autentikasi pengguna, hingga ekspor data [2].

C. Prinsip Kerja

Backend menerima data MWD dalam format *Wellsite Information Transfer Specification* (WITS) dari *gateway*. Data yang diterima pertama kali diproses oleh modul penerimaan data (*data ingestion*), kemudian diteruskan ke modul *parser* WITS untuk mengonversi data mentah menjadi struktur data internal yang dapat diproses oleh sistem. Proses *parsing* dilakukan dengan memetakan setiap parameter WITS ke atribut yang sesuai pada model data MWD sehingga data dapat diproses secara konsisten meskipun berasal dari perangkat atau *gateway* yang berbeda.

Setelah proses *parsing* selesai, data memasuki tahap validasi. Pada tahap ini sistem memeriksa keberadaan parameter penting, seperti kedalaman (*depth*), tekanan, temperatur, azimut, dan *toolface*, serta memastikan tipe data yang diterima sesuai dengan kebutuhan sistem. Apabila terdapat data yang tidak lengkap atau tidak sesuai format, sistem akan menolak data tersebut sehingga tidak disimpan ke dalam basis data.

Data yang telah lolos proses validasi selanjutnya diproses oleh modul sinkronisasi *timestamp* dan kedalaman. Modul ini secara otomatis menambahkan *timestamp* berdasarkan waktu *server* dan menghubungkannya dengan nilai kedalaman pengeboran yang diterima. Selanjutnya data disimpan ke dalam basis data PostgreSQL dalam bentuk yang

terstruktur sehingga dapat digunakan untuk pemantauan *real-time*, analisis historis, maupun proses ekspor data.

Setelah data berhasil disimpan, *frontend* dapat memperoleh informasi terbaru melalui layanan *REST API*, sedangkan komunikasi menggunakan *WebSocket* dimanfaatkan untuk menerima data yang dikirimkan secara langsung oleh perangkat ESP32 sehingga proses pemantauan dapat berlangsung secara *real-time* [3].

D. Struktur Proyek

Implementasi *backend* menerapkan struktur proyek berlapis (*layered architecture*) untuk memisahkan setiap tanggung jawab utama sistem. Pendekatan ini bertujuan meningkatkan keterbacaan kode, mempermudah proses pengembangan, serta memudahkan pemeliharaan ketika sistem dikembangkan lebih lanjut. Setiap modul memiliki fungsi yang spesifik sehingga perubahan pada satu bagian tidak memengaruhi bagian lainnya.

Lapisan *controller* bertanggung jawab menerima permintaan dari pengguna maupun *gateway* dan meneruskannya ke lapisan *service*. Seluruh logika bisnis sistem ditempatkan pada lapisan *service*, sedangkan lapisan *middleware* digunakan untuk proses autentikasi, otorisasi, serta validasi *request*. Selain itu, konfigurasi basis data dipisahkan pada direktori Prisma sehingga pengelolaan model basis data menjadi lebih terstruktur.

TABEL 1
Struktur Direktori Backend

Direktori / File	Fungsi
src/	Direktori root berisi semua source code backend
src/controllers/	Menangani request dari klien atau gateway
src/routes/	Mendefinisikan endpoint API frontend/gateway
src/services/	Berisi logika aplikasi inti (penyimpanan, WebSocket)
src/middlewares/	Menangani autentikasi, otorisasi, validasi request
src/utils/	Berisi fungsi helper seperti validasi payload
src/prisma/	Konfigurasi klien Prisma untuk database
src/server.ts	Titik masuk utama aplikasi
prisma/schema.prisma	Skema database mendefinisikan model
package.json	Konfigurasi proyek Node.js
.env	Konfigurasi environment

E. Konfigurasi Server

Implementasi *backend* diawali dengan konfigurasi *server* sebagai layanan utama yang menjalankan seluruh fungsi sistem pemantauan MWD. Saat aplikasi dijalankan, *server* membaca konfigurasi yang tersimpan pada berkas *.env*, seperti nomor *port*, alamat basis data, *API key*, dan konfigurasi layanan lainnya. Selanjutnya *middleware* yang diperlukan diaktifkan untuk menangani proses *Cross-Origin Resource Sharing* (CORS), *request parsing*, autentikasi, serta penanganan kesalahan (*error handling*).

Setelah konfigurasi selesai dilakukan, sistem mendaftarkan seluruh *endpoint REST API* yang digunakan oleh aplikasi. Berkas **server.ts** berfungsi sebagai titik masuk (*entry point*) aplikasi yang bertugas menjalankan *server*, menginisialisasi koneksi ke basis data, serta mengaktifkan layanan *WebSocket* dan *Serial Gateway* apabila konfigurasi tersebut tersedia.

F. Konektivitas Basis Data

Backend menggunakan Prisma ORM sebagai lapisan akses data yang menghubungkan aplikasi dengan basis data PostgreSQL. Seluruh konfigurasi koneksi dilakukan melalui variabel lingkungan (*environment variable*) DATABASE_URL sehingga informasi koneksi tidak dituliskan secara langsung pada kode program.

Implementasi koneksi dilakukan menggunakan satu instans PrismaClient yang digunakan bersama oleh seluruh layanan aplikasi. Pendekatan ini mengurangi pembuatan koneksi secara berulang sehingga penggunaan sumber daya menjadi lebih efisien dan konsisten. Melalui Prisma ORM, proses penyimpanan, pembaruan, penghapusan, maupun pengambilan data dapat dilakukan menggunakan model objek sehingga pengembangan aplikasi menjadi lebih mudah dan aman.

G. Penerimaan Data dari Gateway

Backend dirancang untuk menerima data MWD melalui beberapa media komunikasi, yaitu *REST API*, *HTTP Gateway*, *Serial Gateway*, serta *WebSocket* dari perangkat ESP32. Pendekatan ini memberikan fleksibilitas sehingga sistem dapat menerima data dari berbagai jenis perangkat akuisisi yang digunakan pada lingkungan pengeboran.

Data yang diterima tidak langsung disimpan ke basis data. Sebelum proses penyimpanan dilakukan, data terlebih dahulu melalui proses validasi, *parsing*, sinkronisasi *timestamp*, serta pemetaan parameter ke struktur basis data. Pendekatan tersebut bertujuan menjaga konsistensi data yang tersimpan.

Pada jalur *HTTP Gateway*, sistem menyediakan *endpoint* /api/gateway/mwd-data yang diamankan menggunakan mekanisme autentikasi berbasis *API key*. Kunci autentikasi dapat dikirim melalui *header* x-gateway-key maupun *Authorization Bearer Token*. Setelah proses autentikasi berhasil dilakukan, *controller* akan meneruskan data ke layanan ingestGatewayPayloads() untuk diproses lebih lanjut.

H. Parsing dan Validasi Data MWD/WITS

Data yang diterima sistem dapat berupa parameter MWD secara langsung maupun data dalam format WITS. Oleh karena itu, *backend* menyediakan modul *parsing* yang mampu membaca berbagai bentuk masukan dan mengubahnya menjadi struktur data yang seragam sesuai kebutuhan sistem.

Setelah proses *parsing* selesai dilakukan, sistem memvalidasi seluruh parameter yang diterima. Validasi meliputi pemeriksaan tipe data, kelengkapan parameter wajib, serta kesesuaian nilai numerik terhadap batas yang ditentukan. Dengan demikian, hanya data yang valid yang dapat disimpan ke dalam basis data.

I. Penyimpanan Data dan Kalibrasi

Data MWD yang telah lolos proses validasi kemudian disimpan ke dalam basis data PostgreSQL melalui lapisan layanan (*service layer*) yang dibangun menggunakan Prisma ORM. Pemisahan proses penyimpanan dari *controller* bertujuan agar logika bisnis tetap terorganisasi dan mudah dipelihara.

Selain menyimpan data utama MWD, sistem juga menyimpan setiap nilai WITS berdasarkan konfigurasi yang telah ditentukan. Sebelum disimpan, nilai mentah (*raw value*) dikalibrasi menggunakan parameter *bias offset* dan *scale factor* sehingga menghasilkan nilai pengukuran yang telah disesuaikan. Nilai hasil kalibrasi kemudian disimpan ke tabel WitsDataValue sehingga baik nilai asli maupun nilai hasil kalibrasi tetap tersedia untuk keperluan analisis maupun pelacakan data.

J. Komunikasi WebSocket Waktu Nyata

Sistem mengimplementasikan komunikasi *WebSocket* untuk menerima data secara langsung dari perangkat ESP32. Pada implementasi ini, *backend* bertindak sebagai klien *WebSocket* yang terhubung ke *endpoint* yang disediakan oleh perangkat ESP32.

Setiap data yang diterima melalui *WebSocket* akan diproses menggunakan mekanisme yang sama dengan data yang diterima melalui *HTTP Gateway*. Dengan demikian, seluruh data yang masuk akan melalui proses validasi, *parsing*, sinkronisasi *timestamp*, serta penyimpanan ke basis data menggunakan alur pemrosesan yang seragam.

K. Desain REST API

REST API berfungsi sebagai media komunikasi utama antara *frontend*, perangkat *gateway*, dan basis data. Seluruh layanan yang berkaitan dengan autentikasi, pengelolaan sesi, pengelolaan data MWD, konfigurasi sistem, ekspor data, hingga pemantauan koneksi disediakan melalui kumpulan *endpoint REST API*.

TABEL 2
Ringkasan Modul Rest Api Backend

Modul	Endpoint Representatif	Fungsi	Akses
Auth	POST /api/auth/login	Login dan penerbitan JWT	Publik / JWT
Gateway	POST /api/gateway/ mwd-data	Menerima data MWD dari gateway	API key
Data MWD	GET, POST /api/mwd-data(/:id)	Melihat, membuat record MWD	JWT; tulis: admin
Sesi	GET, POST /api/mwd-sessions(/:id)	Mengelola sesi pemantauan	JWT; tulis: admin
Ekspor	POST /api/exports/historical	Ekspor CSV/JSON/LAS/PDF	admin/engineer
Koneksi	GET /api/connection-status	Log koneksi dan failover	JWT

Setiap *endpoint* dirancang sesuai dengan fungsi masing - masing serta dilengkapi mekanisme autentikasi dan otorisasi agar hanya pengguna yang memiliki hak akses yang dapat menjalankan operasi tertentu.

L. Autentikasi dan Kontrol Akses

Sistem menerapkan mekanisme autentikasi menggunakan *JSON Web Token* (JWT). Setelah proses *login* berhasil dilakukan menggunakan nama pengguna atau alamat *email* beserta kata sandi, sistem akan menghasilkan *access token* yang digunakan pada setiap permintaan menuju *REST API*.

Selain autentikasi, sistem juga menerapkan *Role-Based Access Control* (RBAC) dengan tiga peran utama, yaitu *administrator*, *engineer*, dan *operator*. Setiap *endpoint* memiliki aturan akses yang berbeda sehingga hanya pengguna dengan hak akses tertentu yang dapat melakukan operasi seperti pengelolaan pengguna, konfigurasi sistem, maupun ekspor data.

M. Ekspor Data

Untuk mendukung proses dokumentasi dan analisis pasca pengeboran, sistem menyediakan fasilitas ekspor data ke berbagai format, yaitu *CSV*, *JSON*, *LAS*, dan *PDF Plot*. Pengguna dapat memilih data yang akan diekspor berdasarkan sesi pemantauan, rentang waktu, maupun rentang kedalaman pengeboran.

Proses ekspor dilakukan oleh *backend* dengan mengambil data dari basis data, menyusunnya sesuai format yang dipilih, kemudian mengirimkan berkas kepada pengguna melalui *REST API*.

N. Status Koneksi dan Penanganan Failover

Sistem menyediakan modul pemantauan status koneksi untuk mengawasi kondisi komunikasi antara *backend* dengan perangkat *gateway*, *Serial Gateway*, maupun ESP32. Informasi status koneksi disimpan pada tabel ConnectionStatus sehingga pengguna dapat mengetahui kondisi setiap sumber data secara *real-time*.

Selain melakukan pencatatan status koneksi, sistem juga mencatat setiap kejadian *failover* apabila terjadi perpindahan sumber data akibat gangguan komunikasi. Informasi tersebut meliputi sumber awal, sumber tujuan, penyebab perpindahan, serta waktu penyelesaiannya sehingga proses pemantauan menjadi lebih andal dan mudah dievaluasi.

IV. HASIL DAN PEMBAHASAN

Pengujian perangkat lunak dilakukan untuk mengonfirmasi bahwa backend, bersama dengan frontend dan basis data yang berinteraksi dengannya, dapat dengan andal melakukan penerimaan, pemrosesan, penyimpanan, penyajian, dan pengelolaan untuk data MWD.

A. Validasi Data dan Penanganan Payload

Pengujian validasi data dilakukan untuk mengevaluasi kemampuan *backend* dalam menerima dan memproses *payload* yang dikirimkan dari *gateway* sesuai dengan struktur data yang telah ditentukan. Pengujian ini bertujuan memastikan bahwa setiap data yang masuk telah memenuhi persyaratan sebelum disimpan ke dalam basis data. Skenario pengujian meliputi *payload* yang valid, *payload* dengan *field* wajib yang hilang, nilai parameter pengeboran yang bernilai *null*, parameter numerik di luar rentang yang ditentukan, serta *payload* berukuran besar.

Setiap *payload* yang diterima terlebih dahulu melalui proses autentikasi, *parsing*, validasi tipe data, serta pemeriksaan kelengkapan parameter. Apabila seluruh proses validasi berhasil dilewati, data akan diteruskan ke tahap penyimpanan. Sebaliknya, apabila ditemukan ketidaksesuaian format maupun parameter yang tidak lengkap, *backend* akan mengembalikan respons kesalahan (*error response*) dan membatalkan proses penyimpanan. Hasil pengujian menunjukkan bahwa seluruh mekanisme validasi berjalan sesuai rancangan sehingga hanya data yang valid yang dapat diproses lebih lanjut.

B. Penyimpanan Data dan Integritas

Pengujian integritas data dilakukan untuk memastikan bahwa setiap data MWD yang diterima *backend* dapat disimpan secara konsisten ke dalam basis data PostgreSQL tanpa mengalami kehilangan, perubahan, maupun duplikasi. Selain itu, pengujian juga memastikan bahwa data yang telah disimpan dapat diambil kembali melalui *REST API* dan ditampilkan pada aplikasi *frontend* dengan informasi yang tetap konsisten [1].

Sebanyak 1.000 data uji dikirimkan ke sistem menggunakan format yang sama seperti kondisi operasional. Setelah seluruh data selesai diproses, dilakukan perbandingan antara jumlah data pada sumber, jumlah data yang diterima *backend*, jumlah data yang tersimpan di PostgreSQL, jumlah data yang dikembalikan oleh *REST API*, serta jumlah data yang ditampilkan pada *frontend*. Hasil pengujian disajikan pada Tabel 3.

TABEL 3
Hasil Pengujian Integritas Data

Parameter	Hasil
Record yang dibuat di sumber	1.000
Record yang diterima oleh backend	1.000
Record yang disimpan di PostgreSQL	1.000
Record yang dikembalikan oleh API	1.000
Record yang ditampilkan di frontend	1.000
Record duplikat	0
Timestamp null	0
Field wajib tidak valid	0

Berdasarkan hasil pengujian, seluruh 1.000 *record* berhasil diterima, disimpan, dan ditampilkan kembali tanpa kehilangan data maupun duplikasi. Selain itu, tidak ditemukan *timestamp* kosong maupun *field* wajib yang tidak valid [4]. Hasil tersebut menunjukkan bahwa mekanisme pemrosesan dan penyimpanan data pada *backend* telah mampu menjaga integritas data selama seluruh alur pemrosesan.

C. Pengujian Fungsional dan Kontrol Akses

Pengujian fungsional dilakukan menggunakan *Playwright* pada peramban Chromium untuk memverifikasi bahwa seluruh fungsi sistem berjalan sesuai dengan kebutuhan fungsional yang telah ditetapkan. Pengujian dilakukan secara *end-to-end*, mulai dari proses autentikasi pengguna hingga visualisasi data pada *dashboard* [5].

Sebanyak lima belas skenario pengujian dikelompokkan ke dalam lima kategori, yaitu autentikasi dan kontrol akses, pengelolaan sesi pemantauan, visualisasi data, pengelolaan data historis dan ekspor, serta mekanisme koneksi dan pemulihan (*recovery*). Ringkasan hasil pengujian ditunjukkan pada Tabel 4.

TABEL 4
Hasil Pengujian Fungsional Berdasarkan Kategori

Kategori	Skenario	Lulus	Gagal
Autentikasi dan kontrol akses	4	4	0
Sesi dan pemantauan dashboard	3	3	0
Perilaku visualisasi	2	2	0
Data historis dan ekspor	4	4	0
Koneksi dan pemulihan	2	2	0
Total	15	15	0

Seluruh skenario pengujian berhasil dijalankan tanpa ditemukan kegagalan. Hasil tersebut menunjukkan bahwa implementasi autentikasi menggunakan *JSON Web Token* (JWT), pembatasan hak akses berbasis peran (*Role-Based Access Control*), serta seluruh fungsi utama sistem telah bekerja sesuai dengan rancangan yang telah ditetapkan.

D. Performa API

Pengujian performa dilakukan untuk mengukur waktu respons layanan *REST API* pada kondisi *local deployment*. Pengujian menggunakan *k6* dengan tiga *endpoint* yang mewakili fungsi utama sistem, yaitu layanan pengambilan daftar sesi MWD, data MWD terbaru, dan data historis. Masing-masing *endpoint* diuji sebanyak tiga kali dengan total 90 permintaan.

Parameter yang diamati meliputi rata-rata waktu respons (*average response time*), latensi persentil ke-95 (*p95 latency*), serta tingkat kesalahan (*error rate*). Ringkasan hasil pengujian ditunjukkan pada Tabel 5.

TABEL 5
Performa Api Dasar

Endpoint	Permintaan	Rata-rata	p95	Tingkat Kesalahan
Sesi MWD	90	4,30 ms	5,72 ms	0%
Data MWD	90	28,14 ms	42,30 ms	0%
Data historis	90	20,04 ms	29,64 ms	0%

Berdasarkan hasil pengujian, seluruh *endpoint* menunjukkan waktu respons yang jauh di bawah kebutuhan nonfungsional sistem yaitu kurang dari 500 ms. *Endpoint* data MWD memiliki latensi tertinggi sebesar 42,30 ms pada nilai *p95*, sedangkan seluruh pengujian menghasilkan *error rate* sebesar 0%. Hasil tersebut menunjukkan bahwa *REST*

API mampu memberikan respons yang cepat dan stabil pada kondisi operasi normal.

E. Performa Akses Konkuren

Selain pengujian pada kondisi normal, dilakukan pula pengujian akses konkuren untuk mengevaluasi kemampuan *backend* dalam melayani beberapa pengguna secara bersamaan. Pengujian dilakukan menggunakan *k6* dengan jumlah *virtual user* sebanyak 1, 5, dan 10 pengguna.

Parameter yang diamati meliputi rata-rata waktu respons, latensi *p95*, serta *throughput* yang dihasilkan selama pengujian. Hasil pengujian ditampilkan pada Tabel 6.

TABEL 6
Hasil Pengujian Akses Konkuren

Pengguna Virtual	Rata-rata	p95	Throughput
1	17,06 ms	32,66 ms	0,997 req/s
5	16,77 ms	27,75 ms	4,982 req/s
10	18,18 ms	33,90 ms	9,942 req/s

Hasil pengujian menunjukkan bahwa peningkatan jumlah pengguna tidak menyebabkan kenaikan waktu respons yang signifikan. Nilai *throughput* meningkat hampir secara linear seiring bertambahnya jumlah pengguna virtual, sedangkan latensi tetap berada di bawah 34 ms pada nilai *p95*. Kondisi tersebut menunjukkan bahwa arsitektur *backend* mampu menangani akses dari beberapa pengguna secara bersamaan tanpa mengalami penurunan performa yang berarti.

F. Pembahasan

Berdasarkan seluruh hasil pengujian, *backend* yang dikembangkan telah memenuhi kebutuhan fungsional maupun nonfungsional sistem pemantauan MWD. Seluruh data berhasil diterima, divalidasi, diproses, dan disimpan tanpa kehilangan maupun duplikasi, sehingga integritas data tetap terjaga selama proses pemantauan berlangsung. Implementasi *REST API* dan *WebSocket* juga memungkinkan pertukaran data secara efisien untuk mendukung pemantauan *real-time*, sementara mekanisme autentikasi berbasis JWT dan *Role-Based Access Control* berhasil membatasi hak akses sesuai peran pengguna.

Dari sisi kinerja, hasil pengujian menunjukkan bahwa waktu respons seluruh layanan *REST API* berada jauh di bawah batas kebutuhan sistem sebesar 500 ms, bahkan pada pengujian dengan hingga 10 pengguna virtual secara bersamaan. Hal ini menunjukkan bahwa kombinasi *Node.js*, *Express.js*, *TypeScript*, *Prisma ORM*, dan PostgreSQL mampu memberikan performa yang baik pada lingkungan *local deployment*. Dengan demikian, arsitektur *backend* yang diusulkan layak digunakan sebagai pusat pemrosesan data pada sistem pemantauan *Measurement While Drilling* berbasis *web* serta memiliki potensi untuk dikembangkan lebih lanjut pada skala implementasi yang lebih besar.

V. KESIMPULAN

Penelitian ini telah berhasil merancang dan mengimplementasikan subsistem *backend* pada sistem pemantauan *Measurement While Drilling* (MWD) berbasis *web* dengan model *local deployment*. *Backend* dikembangkan menggunakan *Node.js*, *Express.js*, *TypeScript*, *Prisma ORM*, dan *PostgreSQL* sebagai pusat pemrosesan data yang menerima data MWD/WITS dari *HTTP gateway*, koneksi serial, dan perangkat ESP32 melalui *WebSocket*. Data yang diterima diproses melalui tahapan validasi, *parsing*, kalibrasi, sinkronisasi *timestamp* berdasarkan kedalaman pengeboran, kemudian disimpan ke dalam basis data dan didistribusikan kepada *frontend* melalui *REST API* dan *WebSocket*. Selain itu, sistem mengimplementasikan autentikasi berbasis *JSON Web Token* (JWT), kontrol akses berbasis peran, serta mekanisme pemantauan status koneksi dan *failover* untuk meningkatkan keamanan dan keandalan sistem.

Hasil pengujian menunjukkan bahwa *backend* mampu mempertahankan integritas seluruh 1.000 *record* data uji tanpa kehilangan maupun duplikasi, serta berhasil melewati seluruh 15 skenario pengujian fungsional dan kontrol akses. Pengujian performa juga menunjukkan bahwa seluruh layanan *REST API* memiliki latensi di bawah kebutuhan sistem sebesar 500 ms dengan *error rate* sebesar 0%, baik pada kondisi normal maupun saat melayani hingga 10 pengguna virtual secara bersamaan. Hasil tersebut menunjukkan bahwa arsitektur *backend* yang dikembangkan mampu mendukung pemantauan data MWD secara *real-time*, andal, dan terintegrasi pada lingkungan *local deployment*. Penelitian selanjutnya dapat difokuskan pada evaluasi performa pada lingkungan pengeboran sebenarnya, pengembangan mekanisme *scaling* untuk meningkatkan kapasitas pemrosesan data, serta penyempurnaan fitur alarm dan *failover* guna meningkatkan keandalan sistem dalam kondisi operasional lapangan.

REFERENSI

[1] J. Chen, “Model Algorithm Research based on Python Fast API,” *Frontiers in Science and Engineering*, vol. 3, no. 9, pp. 7–10, Sep. 2023, doi: 10.54691/fse.v3i9.5591.

[2] M. M. Al-Debei, “Data Warehouse as a Backbone for Business Intelligence: Issues and Challenges,” 2011. [Online]. Available: http://www.eurojournals.com

[3] M. I. Firdian, E. Darwiyanto, and M. Adrian, “Web Scraping with HTML DOM Method for Website News API creation,” JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika).

[4] P. Annaiyappa, J. Macpherson, and E. Cayeux, “Clock Synchronization and Timestamping of Data on Acquisition at the Wellsite: Guidelines and Recommendations,” *SPE Drilling & Completion*, pp. 1–16, Jun. 2023, doi: 10.2118/208732-PA.

[5] R. A. Ruddro, “ENERGY EFFICIENCY MONITORING IN INSTITUTIONAL FACILITIES: A REVIEW OF UI/UX SOLUTIONS FOR IOT DASHBOARDS,” *International Journal of Scientific Interdisciplinary Research*, vol. 06, no. 01, pp. 137–162, Mar. 2025, doi: 10.63125/yhr4g345.