

Rancang Bangun Backend *Service* Berbasis *REST* API untuk Sistem Manajemen Jadwal Perkuliahan di FTS UIKA

Muhammad Ihsan Fadillah¹, Safaruddin Hidayat Al Ikhsan², Berlina Wulandari³

^{1,2,3}Program Studi Teknik Informatika, Universitas Ibn Khaldun Bogor

Email Koresponding: Muhammadihsanf270@gmail.com¹

Published: Agustus, 2026

ABSTRAK

Penyusunan jadwal perkuliahan di Fakultas Teknik dan Sains Universitas Ibn Khaldun Bogor saat ini masih dilakukan menggunakan perangkat pengolah angka (*spreadsheet*) secara semi-manual. Hal ini menyebabkan pengelolaan data akademik belum berjalan optimal dan sangat rentan terhadap kesalahan penginputan. Penelitian ini bertujuan untuk merancang dan membangun *backend service* berbasis REST API sebagai penyedia layanan data terpusat yang aman, terstruktur, dan memiliki skalabilitas tinggi. Metode pengembangan sistem yang digunakan adalah *Waterfall*, yang meliputi tahap analisis kebutuhan, perancangan, implementasi, pengujian, dan *deployment*. Arsitektur *server* dikembangkan menggunakan lingkungan *runtime* Node.js dengan kerangka kerja Express.js, serta sistem manajemen basis data PostgreSQL yang dikelola melalui abstraksi Prisma ORM. Pengujian fungsionalitas sistem dievaluasi menggunakan metode *Black Box Testing* pada setiap rute API, dilanjutkan dengan tahap *deployment* dan konfigurasi pada lingkungan *server*. Hasil penelitian menunjukkan bahwa seluruh *endpoint* utama untuk pengelolaan data akademik berhasil merespons permintaan dengan tepat dan mengembalikan data berformat JSON secara *stateless* dengan kode status HTTP yang valid. Layanan web ini telah beroperasi secara stabil pada *server* dan siap diintegrasikan secara fleksibel dengan berbagai *platform* antarmuka klien untuk mendukung efisiensi operasional akademik fakultas.

Kata Kunci: *Backend Service*, REST API, Node.js, Express.js, PostgreSQL, *Waterfall*.

© The Author(s). 2021 Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0>

PENDAHULUAN

Penjadwalan mata kuliah merupakan masalah optimasi yang rumit karena harus menyelaraskan berbagai sumber daya yang terbatas, seperti ruang kelas, dosen, dan waktu [1]. Kendati krusial, masih banyak institusi pendidikan yang bergantung pada metode konvensional yang bersifat semi-manual [2]. Metode pencocokan data secara visual ini dinilai sangat tidak efisien, rentan memicu terjadinya bentrok jadwal, dan proses penyelesaiannya dapat memakan waktu hingga satu minggu [3]. Kondisi inefisiensi ini secara nyata terjadi di Fakultas Teknik dan Sains (FTS) Universitas Ibn Khaldun (UIKA) Bogor, di mana penyusunan jadwal masih memanfaatkan perangkat lunak pengolah angka (*spreadsheet*) sederhana. Pengelolaan data masih menggunakan *spreadsheet* terbukti sangat rentan terhadap human error pada saat penginputan data [4]. Ketidadaan sistem otomatisasi yang terpusat ini berpotensi besar menimbulkan konflik operasional, seperti bentrok penggunaan ruang kelas maupun keterlambatan distribusi informasi jadwal kepada civitas akademika [5].

Untuk memberikan solusi atas kendala tersebut, diperlukan pengembangan Backend Service yang arsitekturnya terpisah secara independen dari antarmuka pengguna. Pemisahan logika bisnis pada sisi *backend* ini memungkinkan pengelolaan data akademik menjadi lebih aman dan terstruktur [6]. Arsitektur perangkat lunak yang diimplementasikan adalah *Representational State Transfer* (REST) API.

Arsitektur REST API beroperasi dengan prinsip stateless dan mentransmisikan data menggunakan format *JavaScript Object Notation* (JSON) yang sangat ringan [7]. Penerapan standar komunikasi ini menjamin efisiensi dalam pertukaran data, sehingga hasil penjadwalan akademik dapat diakses dan diintegrasikan secara fleksibel ke berbagai platform antarmuka dengan tingkat skalabilitas yang tinggi di masa depan [8].

Dalam mendukung kelancaran operasional layanan manajemen jadwal, pemilihan teknologi server menjadi faktor yang sangat krusial. Pengembangan backend pada sistem ini diimplementasikan menggunakan *framework* Express.js yang berjalan di atas lingkungan Node.js. Pemilihan teknologi ekosistem JavaScript ini didasarkan pada studi komparasi yang membuktikan bahwa *framework* tersebut memiliki tingkat latensi yang jauh lebih rendah dibandingkan dengan teknologi penyedia layanan sejenis [9]. Lebih lanjut, dalam penerapan arsitektur REST API, teknologi berbasis Node.js mampu memberikan performa throughput yang sangat tinggi [10]. Karakteristik ini sangat ideal untuk menangani proses komputasi pertukaran data yang padat serta menjamin stabilitas server dalam mengelola jadwal perkuliahan di FTS UIKA.

TINJAUAN PUSTAKA

Web Service dan REST API

Web Service merupakan sistem perangkat lunak yang dirancang secara spesifik untuk memfasilitasi interaksi dan pertukaran data antar mesin melalui jaringan terdistribusi [11]. Dalam pengembangannya, gaya arsitektur REST diterapkan karena memanfaatkan protokol dasar HTTP dan bekerja dengan prinsip stateless. Prinsip ini memastikan bahwa setiap permintaan (*request*) dari klien ke server bersifat independen dan memuat seluruh informasi yang dibutuhkan tanpa menyimpan status sesi dari permintaan sebelumnya [12]. Standar utama dalam pertukaran data pada REST API adalah *JavaScript Object Notation* (JSON), yang dipilih karena ukurannya yang ringan sehingga mempercepat proses transmisi beban data antar platform [7].

Node.js dan Express.js

Kinerja backend sangat krusial dalam menentukan *response time* aplikasi, terutama saat menjalankan proses komputasi yang berat [10]. Lingkungan server pada arsitektur ini dibangun di atas Node.js, sebuah *runtime environment* yang memungkinkan bahasa tingkat tinggi JavaScript dieksekusi secara asinkron di sisi server. Node.js memanfaatkan arsitektur *event-driven* dan operasi input/output non-pemblokiran (*non-blocking I/O*), menjadikannya sangat ringan dalam menangani banyak permintaan jaringan secara bersamaan (*concurrency*). Kompleksitas pengelolaan siklus request dan response pada modul inti Node.js disederhanakan menggunakan Express.js, sebuah kerangka kerja aplikasi web minimalis yang menyediakan fungsionalitas kokoh untuk membangun struktur routing API [13].

Manajemen Basis Data Relasional

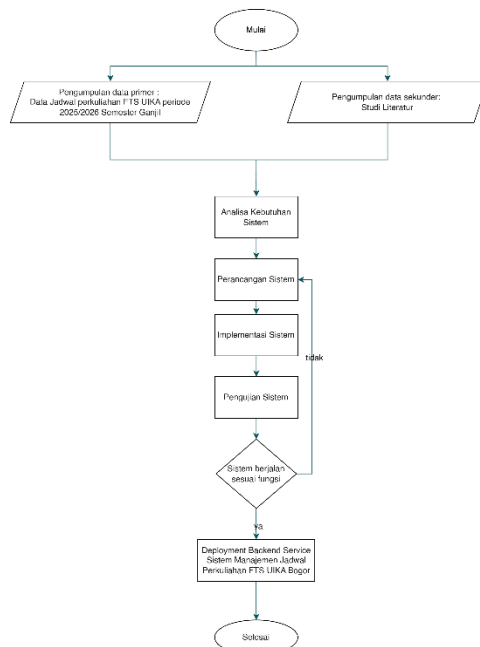
Penyimpanan dan integritas data sistem dikelola menggunakan PostgreSQL, sebuah sistem manajemen basis data relasional yang mengutamakan stabilitas struktur dan konsistensi transaksi data. Karakteristik ini memastikan arsitektur database mampu mengelola beban kerja relasional yang padat dengan efisien [14]. Interaksi antara logika server Node.js dan PostgreSQL dijumpai melalui *Object Relational Mapping* (ORM) Prisma. Prisma menyediakan fitur *type-safety* dan manajemen skema secara intuitif untuk memastikan keamanan akses data, meminimalisasi kesalahan penulisan kueri, serta meningkatkan tingkat pemeliharaan (*maintainability*) kode pada sistem skala menengah hingga besar [15].

Pemodelan Sistem dan Pengujian Sistem

Spesifikasi perancangan perangkat lunak direpresentasikan pada berbagai tingkat abstraksi menggunakan standar *Unified Modeling Language* (UML). Pemodelan ini mendeskripsikan secara konkret interaksi pengguna (*Use Case*), modularitas sistem (*Component*), hingga rancangan distribusi infrastruktur server pada lingkungan nyata (*Deployment*) [16]. Setelah tahap implementasi selesai, setiap endpoint API divalidasi menggunakan metode *Black Box Testing*. Pendekatan pengujian ini berfokus pada evaluasi fungsionalitas input dan output tanpa memeriksa kode internal, untuk menjamin bahwa seluruh rute API telah memberikan respons JSON yang valid sesuai dengan spesifikasi permintaan klien [17].

METODE PENELITIAN

Penelitian ini menerapkan metode *Waterfall* sebagai pendekatan utama dalam pengembangan backend service. Metode ini dipilih karena menyediakan alur kerja yang sistematis dan berurutan, di mana setiap tahapan harus diselesaikan terlebih dahulu sebelum melangkah ke fase berikutnya [31]. Sesuai dengan batasan masalah, tahapan pengembangan perangkat lunak dalam penelitian ini dibatasi pada lima fase utama: Analisis, Perancangan, Implementasi, Pengujian, dan Deployment. Alur penelitian ditunjukkan pada **Error! Reference source not found..**



Gambar 1 *Flowchart* Metode Penelitian Waterfall

Perancangan Sistem

Tahap ini dilakukan untuk mengidentifikasi dan mendefinisikan spesifikasi kebutuhan sistem dalam pengembangan *Backend Service*. Analisis mencakup identifikasi kebutuhan fungsional berupa pendefinisian layanan *endpoint* REST API yang harus disediakan sistem, seperti modul autentikasi, master data akademik, pengajaran, dan penjadwalan. Selain itu, dilakukan analisis kebutuhan non-fungsional yang difokuskan pada aspek keandalan (*reliability*), performa (*performance*), dan ketepatan (*correctness*) server dalam merespons setiap permintaan klien menggunakan format data JSON yang standar.

Perancangan Sistem

Tahap perancangan merupakan proses menerjemahkan hasil analisis ke dalam cetak biru arsitektur perangkat lunak. Pemodelan sistem dilakukan menggunakan *Unified Modeling Language* (UML) yang mencakup perancangan *Use Case Diagram*, *Class Diagram*, *Component Diagram*, dan *Deployment Diagram* untuk memetakan alur interaksi serta arsitektur infrastruktur server. Pada tahap ini juga dilakukan perancangan logis struktur basis data relasional serta spesifikasi detail dari *endpoint* REST API.

Implementasi Sistem

Tahap implementasi adalah proses penerjemahan rancangan arsitektur sistem ke dalam bentuk kode program aktual. *Backend Service* dibangun memanfaatkan lingkungan runtime Node.js yang dipadukan dengan kerangka kerja Express.js untuk manajemen *routing* API. Sementara itu, struktur basis data yang telah dirancang diimplementasikan ke dalam *Relational Database Management System* (RDBMS) PostgreSQL dan dikelola menggunakan abstraksi Prisma *Object-Relational Mapping* (ORM) guna memastikan integritas dan keamanan transaksi data (*type-safe*).

Pengujian

Tahap pengujian dilakukan untuk memvalidasi fungsionalitas sistem secara keseluruhan sebelum digunakan. Pengujian *Backend Service* ini dilakukan menggunakan metode *Black Box Testing* yang berfokus pada evaluasi input dan output aplikasi. Pengujian ini bertujuan untuk memverifikasi bahwa setiap *endpoint* API (GET, POST, PATCH, DELETE) mampu memproses parameter dengan

benar dan mengembalikan respons berformat JSON yang valid sesuai dengan spesifikasi yang telah ditentukan.

Deployment

Tahap deployment merupakan proses penyiapan dan konfigurasi akhir agar *Backend Service* yang telah dikembangkan dapat dijalankan secara stabil. Pada tahap ini, seluruh komponen aplikasi, konfigurasi lingkungan (*environment variables*), koneksi layanan pendukung, serta basis data PostgreSQL disiapkan pada lingkungan server atau lingkungan pengujian sehingga rute REST API dapat diakses dengan baik oleh platform klien.

HASIL DAN PEMBAHASAN

Analisis

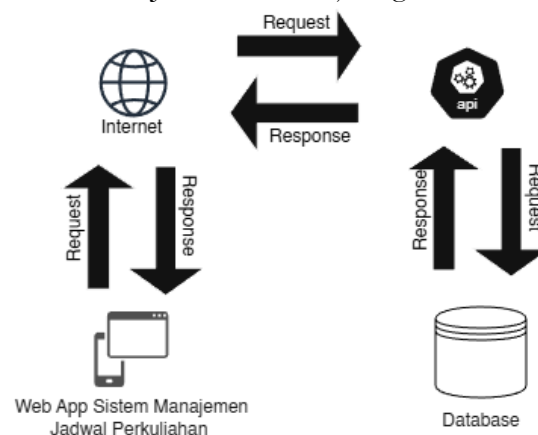
Tahap analisis dilakukan untuk mengidentifikasi kebutuhan sistem backend yang akan dikembangkan. Analisis meliputi identifikasi kebutuhan fungsional berupa layanan endpoint REST API, kebutuhan non-fungsional, serta arsitektur sistem.

1) Analisis Kebutuhan Fungsional

Kebutuhan fungsional backend service ini dipetakan ke dalam beberapa modul utama yang menyediakan rute (*routes*) API untuk diakses oleh *client*. Modul tersebut meliputi Autentikasi (mengelola token login), Master Data Akademik (mengelola data fakultas, program studi, ruangan, waktu, dll), Kurikulum, Pengajaran (mengelola penugasan dan batasan dosen), Penjadwalan (memproses algoritma genetika), serta Perubahan Jadwal.

2) Analisis Arsitektur Sistem

Arsitektur sistem dibangun dengan model *client-server*. Aplikasi berbasis web (sebagai *client*) mengirimkan request HTTP berformat JSON melalui jaringan internet menuju server.



Gambar 2 Arsitektur Sistem

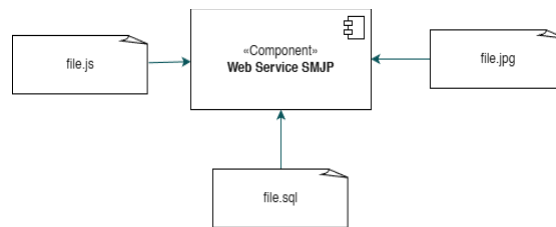
Berdasarkan Gambar 2, arsitektur sistem mengadopsi model *client-server* yang berkomunikasi melalui jaringan internet. Aplikasi *client* (Web App Sistem Manajemen Jadwal) bertugas mengirimkan *request* HTTP, yang kemudian diterima dan diproses oleh *Web Service* (API). *Web Service* ini berinteraksi langsung dengan *database* untuk melakukan operasi pengolahan data, lalu mengembalikan hasilnya dalam bentuk *response* JSON kepada *client*.

Perancangan Sistem

Tahap perancangan dilakukan untuk menerjemahkan kebutuhan sistem ke dalam pemodelan Unified Modeling Language (UML) dan struktur basis data.

3) Component Diagram

Component Diagram memvisualisasikan struktur modular dari *backend service*. Diagram ini menunjukkan ketergantungan antar komponen di dalam sistem.

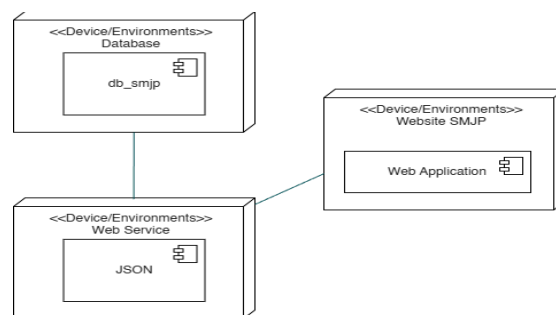


Gambar 3 *Component Diagram*

Berdasarkan Gambar 3, Component Diagram mengilustrasikan pembagian komponen logika bisnis yang berjalan di dalam Web Service. Komponen pengendali rute (*route controllers*) bertugas menerima *request*, yang kemudian diproses oleh lapisan layanan (*services*). Lapisan ini berinteraksi dengan komponen repositori data yang memiliki akses langsung ke *Database Management System* (DBMS) PostgreSQL.

4) Deployment Diagram

Arsitektur fisik infrastruktur server dirancang agar beban kerja aplikasi dapat terdistribusi dengan baik.

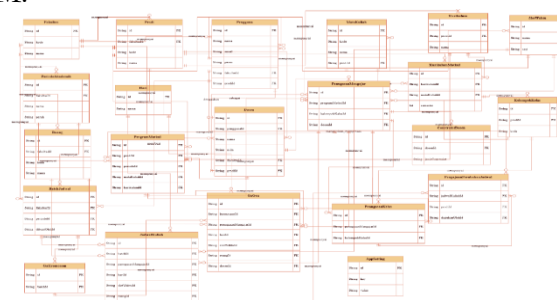


Gambar 4 *Deployment Diagram*

Berdasarkan Gambar 4, Deployment Diagram menunjukkan topologi jaringan fisik sistem. Terdapat pemisahan (*separation of concerns*) yang jelas antara *node client* (peramban web pengguna), *node web server* yang menjalankan *runtime* Node.js untuk layanan REST API, dan *node database server* yang menjalankan mesin basis data relasional.

5) Perancangan Database

Basis data dirancang menggunakan *Entity Relationship Diagram* (ERD) dan diimplementasikan melalui skema Prisma ORM.



Gambar 5 Perancangan Database

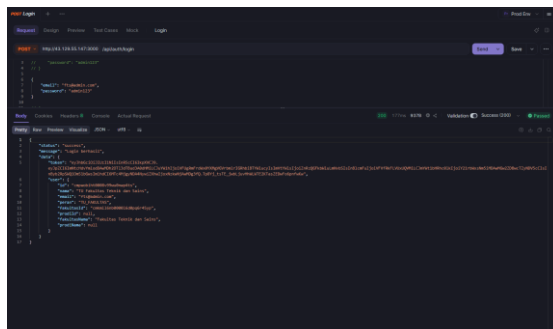
Berdasarkan Gambar 5, skema relasi basis data mendeskripsikan pemetaan 22 tabel yang saling terhubung. Tabel-tabel tersebut dikelompokkan menjadi tabel master (seperti Fakultas, Ruang, Dosen) dan tabel transaksional (seperti PenugasanMengajar dan JadwalKuliah). Integritas data dijaga melalui relasi *Primary Key* dan *Foreign Key* yang dikelola secara otomatis oleh konfigurasi Prisma.

Implementasi Sistem

Tahap implementasi merupakan proses penerjemahan rancangan ke dalam kode program aktual. Lingkungan *server* dibangun menggunakan Node.js dengan kerangka kerja Express.js.

1) Implementasi Endpoint Login

Endpoint ini berfungsi sebagai gerbang keamanan dengan memvalidasi *email* dan *password* pengguna.

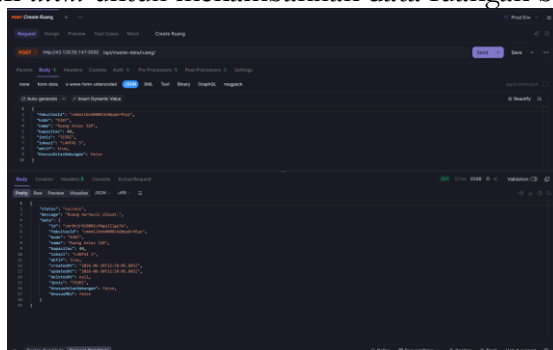


Gambar 6 Respon *Endpoint Login*

Berdasarkan Gambar 6, respons dari *endpoint* login mengembalikan kode status 200 OK. *Body response* menghasilkan struktur JSON yang memuat token otorisasi berjenis *JSON Web Token (JWT)*. Token ini akan digunakan oleh *client* sebagai otentikasi (pada *header* HTTP) untuk setiap *request* pada rute-rute API yang bersifat terproteksi.

2) Implementasi *Endpoint* Master Data Ruang

Pengelolaan infrastruktur fasilitas fisik diimplementasikan melalui rute *endpoint* ruang. *Endpoint* ini menerima *request payload* dari *client* untuk menambahkan data ruangan baru ke dalam *database*.

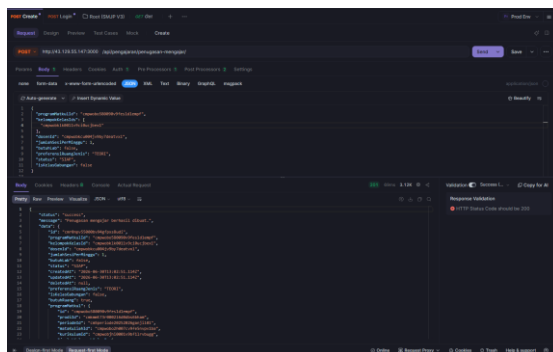


Gambar 7 Respon *Endpoint Ruang*

Berdasarkan Gambar 7, saat *client* mengirimkan data ruangan baru, sistem akan memvalidasi *request body* dan menyimpannya ke *database*. Sistem kemudian mengembalikan kode HTTP 201 *Created* beserta format JSON berisi detail ID unik ruangan, kapasitas, lokasi, serta jenis ruangan yang baru saja dibuat.

3) Implementasi *Endpoint* Penugasan Mengajar

Untuk melihat distribusi beban mengajar dosen, sistem menyediakan *endpoint* penugasan mengajar.

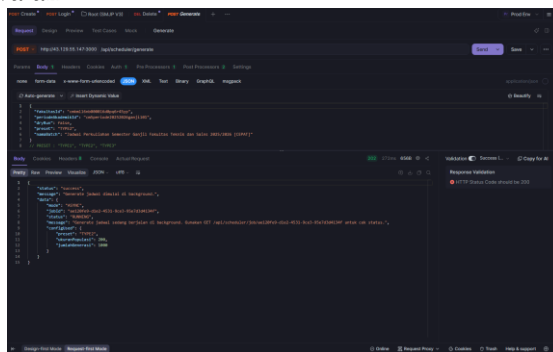


Gambar 8 Respon *Endpoint Penugasan Mengajar*

Berdasarkan Gambar 8, *endpoint* ini mendemonstrasikan penanganan relasi data yang kompleks pada *database*. *Response* yang dihasilkan tidak hanya memuat data penugasan, melainkan juga menggabungkan (*join*) objek entitas yang berelasi dengannya, seperti detail Dosen, Mata Kuliah, dan Kelompok Kelas dalam satu hierarki struktur JSON terpadu.

4) Implementasi *Endpoint Generate Jadwal*

Fitur paling esensial dalam sistem ini diproses melalui rute *generate* jadwal, yang bertugas memicu eksekusi algoritma penjadwalan.



Gambar 9 Respon *Endpoint Generate Jadwal*

Berdasarkan Gambar 9, karena komputasi penjadwalan membutuhkan waktu eksekusi yang panjang, *endpoint* ini dirancang untuk bekerja secara asinkron (*background job*). Ketika dipanggil, *server* langsung memberikan respons kode 200 OK yang memuat *jobId* dengan status "RUNNING", menginformasikan kepada *client* bahwa proses penyusunan jadwal sedang dieksekusi di balik layar tanpa memblokir lalu lintas *request* lainnya.

Pengujian

Tahap pengujian bertujuan untuk memverifikasi fungsionalitas sistem secara keseluruhan sebelum diimplementasikan. Pengujian pada *backend service* ini menggunakan metode *Black Box Testing* yang berfokus pada evaluasi fungsionalitas input dan output pada setiap *endpoint* API, tanpa perlu memeriksa struktur logika internal kode program. Pengujian dilakukan dengan mengirimkan berbagai skenario *request* (skenario normal maupun pengujian *error*) menggunakan perangkat pengujian API untuk memastikan *server* mengembalikan *response* yang tepat. Pengujian dapat dilihat pada Tabel 1.

Tabel 1 Blackbox Testing

No	URL API Pengujian	Skenario Pengujian	Hasil yang diharapkan	Hasil Uji
1	[POST] {BASE_URL}/api/auth/login	Kirim request POST dengan email dan password yang valid.	Success response dengan status code 200 dan mengembalikan token JWT.	Berhasil
2	[GET] {BASE_URL}/api/master-data/ruang	Kirim request GET beserta token autentikasi yang valid.	Success response dengan status code 200 dan mengembalikan daftar data ruang.	Berhasil
3	[POST] {BASE_URL}/api/master-data/ruang	Kirim request POST dengan body data ruang yang lengkap dan valid beserta token autentikasi.	Success response dengan status code 201 dan data ruang berhasil ditambahkan.	Berhasil
4	[GET] {BASE_URL}/api/pengajaran/pengajaran-mengajar	Kirim request GET beserta token autentikasi yang valid.	Success response dengan status code 200 dan mengembalikan daftar data penugasan mengajar.	Berhasil
5	[POST] {BASE_URL}/api/pengajaran/pengajaran-mengajar	Kirim request POST dengan body data penugasan mengajar yang lengkap dan valid beserta token autentikasi.	Success response dengan status code 201 dan data penugasan mengajar berhasil ditambahkan.	Berhasil
6	[POST] {BASE_URL}/api/schedule/batch/{id}/resolve-conflicts	Kirim request POST dengan ID batch yang valid beserta token autentikasi.	Success response dengan status code 200 dan sistem berhasil menjalankan proses penyelesaian konflik jadwal.	Berhasil
7	[GET] {BASE_URL}/api/schedule/batch	Kirim request GET beserta token autentikasi yang valid.	Success response dengan status code 200 dan	Berhasil

			mengembalikan daftar batch jadwal.	
--	--	--	---------------------------------------	--

Deployment

Tahap *deployment* merupakan proses penyiapan, instalasi, dan konfigurasi akhir agar *backend service* yang telah dibangun dapat beroperasi secara stabil di lingkungan *server* produksi dan siap melayani permintaan data (*request*) dari aplikasi *client*.

Proses *deployment* berhasil dilakukan dengan mengatur konfigurasi variabel lingkungan (*environment variables*). Konfigurasi ini mencakup pengaturan port *server*, pendefinisian koneksi *string* DATABASE_URL untuk mengamankan akses ke layanan PostgreSQL, serta konfigurasi lapisan keamanan *Cross-Origin Resource Sharing* (CORS) pada *middleware* Express.js. Pengaturan CORS ini diimplementasikan untuk memberikan batasan akses, memastikan bahwa REST API ini hanya menerima dan merespons permintaan pertukaran data yang berasal dari domain aplikasi *frontend* resmi Sistem Manajemen Jadwal Perkuliahan FTS UIKA.

KESIMPULAN

Penelitian ini telah berhasil merancang, membangun, dan melakukan *deployment backend service* berbasis REST API untuk Sistem Manajemen Jadwal Perkuliahan di Fakultas Teknik dan Sains Universitas Ibn Khaldun Bogor. Layanan web terpusat ini dikembangkan menggunakan lingkungan Node.js dan Express.js, serta mampu mengelola seluruh entitas data master akademik dan hasil optimasi jadwal secara terstruktur melalui integrasi basis data PostgreSQL dengan Prisma ORM. Hasil pengujian fungsionalitas menggunakan metode *Black Box Testing* membuktikan bahwa seluruh rute API berjalan dengan valid, dan sistem telah berhasil dikonfigurasi pada lingkungan server produksi secara stabil dengan menerapkan kebijakan keamanan *Cross-Origin Resource Sharing* (CORS). Adapun saran untuk pengembangan selanjutnya adalah menambahkan sistem pemantauan log (*logging system*) untuk melacak performa lalu lintas data pada server secara real-time, serta menerapkan mekanisme keamanan tingkat lanjut seperti pembatasan laju permintaan (*rate limiting*) untuk melindungi ketersediaan layanan dari potensi lonjakan akses yang tidak wajar.

REFERENCES

- Ahmad, T. M., & Lamasitudju, C. A. (2025). Optimasi penjadwalan rapat berbasis web untuk mengurangi konflik jadwal menggunakan kombinasi algoritma Greedy dan Decision Tree. *Jurnal Algoritma*, 22(2). <https://doi.org/10.33364/algoritma/v.22-2.3080>
- Asrofi, Metra, P., & Baharsyah, B. (2025). Perancangan sistem cerdas penjadwalan mata kuliah otomatis berbasis web menggunakan algoritma genetika. *Jurnal Komputer Teknologi Informasi Sistem Informasi (JUKTISI)*, 4(1), 269–283. <https://doi.org/10.62712/juktisi.v4i1.407>
- Azzahidi, A. S., Wijayanto, B., & Darmawan, A. (2025). Performance evaluation of backend frameworks for REST API: A comparative study of Spring Boot, Flask, Express.js, Laravel FrankenPHP, and Gin. *Jurnal Teknik Informatika (Jutif)*, 6(4), 2405–2419. <https://doi.org/10.52436/1.jutif.2025.6.4.4811>
- Brown, E. (2019). *Web development with Node and Express*. O'Reilly Media.
- Efendi, R., & Maruloh. (2024). Sistem informasi absensi pada PT. Garuda Inti Sentosa berbasis teknologi web service RESTfull API. *INFOTECH Journal*, 10(1). <https://doi.org/10.31949/infotech.v10i1.9177>
- Fitriani, F. (2024). Penjadwalan kuliah berbasis website pada Perguruan Tinggi Politeknik Sampit [Skripsi/Laporan, Perguruan Tinggi].
- Hadinata, W., & Stianingsih, L. (2024). Analisis perbandingan performa RESTfull API antara Express.js dengan Laravel framework. *Jurnal Informatika dan Teknik Elektro Terapan*, 12(1). <https://doi.org/10.23960/jitet.v12i1.3845>
- Mone, F., & Simarmata, J. E. (2021). Application of genetic algorithm in scheduling subjects. *Barekeng: Jurnal Ilmu Matematika dan Terapan*, 15(4), 615–628. <https://doi.org/10.30598/barekengvol15iss4pp615-628>

- Ponggawa, V., Sendiang, M., & Mappadang, J. L. (2024). The development of Polimdo Restfull API to support data exchange effectively. <https://polimda-api.my.id/>
- Pratama, A. N., & Ardiani, F. (2023). Optimization of Model-View-ViewModel (MVVM) architecture pattern and RESTfull API on Android-based e-learning application.
- Pressman, R. S. (2010). Software engineering: A practitioner's approach (7th ed.). McGraw-Hill. <http://www.mhhe.com/pressman>
- Richardson, L., & Ruby, S. (2007). RESTful web services. O'Reilly Media.
- Shamlan, M. K., Edi, S. W. M., & Wacana, K. S. (2025). Pengujian perfoma UUID sebagai primary key pada database PostgreSQL dan MongoDB.
- Solihin, H. H., Suhernawan, M. R. P., Gunawan, H., & Versanika, D. V. (2025). Rancang bangun back-end aplikasi HaePe untuk penjadwalan tugas rumah tangga berbasis REST API. *Jurnal Penelitian dan Pengembangan Teknologi Informasi dan Komunikasi*, 14(1), 15–25. <https://doi.org/10.58761/jurtikstmikbandung.v14.i1.181>
- Yusmita, J. C., Arya, R., Wijaya, J. M., Suryaningrum, K. M., & Siswanto, R. R. (2025). Optimizing database access strategy: A performance analysis comparison of raw SQL and Prisma ORM. *Procedia Computer Science*, 1201–1210. <https://doi.org/10.1016/j.procs.2025.09.061>
- Zulkarnaini, Firdhayanti, A., Taufik, T., & Bachry, B. (2023). User acceptance testing through blackbox evaluation for corn distribution information system. *bit-Tech*, 6(2), 208–215. <https://doi.org/10.32877/bt.v6i2.1065>
- Zulmahendra, & Derta, S. (2020). Penjadwalan kuliah otomatis menggunakan algoritma genetika. *JSI: Jurnal Sistem Informasi (E-Journal)*, 12(2). <http://ejournal.unsri.ac.id/index.php/jsi/index>.