Jurnal Spektran
Vol. 11, No. 1, Januari 2023, Hal. xxx - xxx
p-ISSN: 2302-2590, e-ISSN : 2809-7718
http://ojs.unud.ac.id/index.php/jsn/index

Pengembangan Application Programming Interface pada Database MongoDB menggunakan FastAPI

Afdal Wahyi[1], Eko Rudiawan Jamzuri[2]

[1,2]Department of Electrical Engineering, Politeknik Negeri Batam, Batam, Kepulauan Riau, Indonesia

afdalwahyi534@gmail.com
ekorudiawan@polibatam.ac.id *(Corresponding author)*

ABSTRAK

Pengembangan aplikasi online dan seluler saat ini merupakan aspek penting dalam dunia perangkat lunak yang dinamis. Permintaan akan proses perizinan yang efisien, responsif, dan terhubung dengan layanan dan sumber daya lain semakin meningkat. *Application Programming Interface* (API) memainkan peran penting dalam memastikan interoperabilitas dan komunikasi yang efektif antara komponen perangkat lunak. *Representational State Transfer API* (RESTAPI) adalah jenis API yang banyak digunakan, menggunakan protokol HTTP untuk mentransfer data dalam format JSON atau XML. Inovasi terkini dalam pengembangan API adalah FastAPI, sebuah kerangka pemrograman web Python yang efisien dan kuat. Integrasi FastAPI dengan MongoDB, basis data NoSQL yang fleksibel, memberikan solusi efektif dalam pengembangan aplikasi modern. FastAPI memungkinkan pembuatan API dengan validasi data otomatis dan dokumentasi terintegrasi, sementara MongoDB mendukung penyimpanan data yang dapat berkembang dan mengatasi pertumbuhan skala horizontal. Hasil penelitian ini menunjukkan waktu respons menggunakan koneksi kabel LAN yang lebih rendah dengan rata-rata 33.4 ms dibandingkan jaringan WiFi dengan rata-rata 1674.9 ms, sehingga integrasi FastAPI dan MongoDB mampu menghasilkan aplikasi yang responsif, efisien, dan mampu menangani data yang terus berkembang pesat, menjadikannya solusi yang sesuai dengan kebutuhan aplikasi kontemporer.

Kata kunci: *Pengembangan aplikasi, RESTAPI, MongoDB, FastAPI*

Development of Application Programming Interface on MongoDB Database using FastAPI

ABSTRACT

Online and mobile application development is today an important aspect in the dynamic world of software. Demand for licensing processes that are efficient, responsive, and connected to other services and resources is increasing. Application Programming Interface (API) plays an important role in ensuring interoperability and effective communication between software components. Representational State Transfer API (RESTAPI) is a widely used type of API, using the HTTP protocol to transfer data in JSON or XML format. The latest innovation in API development is FastAPI, an efficient and powerful Python web programming framework. FastAPI integration with MongoDB, a flexible NoSQL database, provides an effective solution in modern application development. FastAPI enables the creation of APIs with automated data validation and integrated documentation, while MongoDB supports scalable data stores and handles horizontal scale growth. The results of this study indicate that the response time using a lower LAN cable connection averages 33.4 ms compared to the WiFi network with an average of 1674.9 ms. Therefore, the integration of FastAPI and MongoDB is capable of producing responsive and efficient applications that can handle rapidly growing data. This makes it a solution that aligns with the needs of contemporary applications.

Keywords: *Application development, RESTAPI, MongoDB, FastAPI*

1 PENDAHULUAN

Pengembangan aplikasi berbasis *online* dan seluler telah menjadi salah satu aspek yang masih dinamis dan penting dalam dunia perangkat lunak di era teknologi informasi yang berkembang pesat [1]. Permintaan akan proses perizinan yang efisien, responsif, dan terhubung dengan layanan dan sumber daya lainnya semakin meningkat. Selain itu, volume data yang terus bertambah semakin menantang pengembang aplikasi [2].

Dalam dunia pengembangan perangkat lunak, *Application Programming Interface* (API) merupakan komponen yang sangat penting dan tidak bisa diabaikan. API memungkinkan pengguna untuk berinteraksi dengan layanan atau aplikasi lain secara konsisten dan standar, yang sangat penting dalam memastikan interoperabilitas dan komunikasi yang efektif antara berbagai komponen perangkat lunak [3], [4]. Representational State Transfer API (RESTAPI) adalah jenis API yang menggunakan protokol HTTP seperti *GET*, *POST*, *PUT*, dan *DELETE* untuk mentransfer data dalam format JSON atau XML [5]. RESTAPI merupakan standar desain arsitektur yang sering digunakan dalam pengembangan aplikasi web [6], [7]. RESTAPI juga menjadi pilihan yang populer untuk pengembangan API karena sederhana, mudah digunakan, dan dapat diimplementasikan di berbagai bahasa pemrograman [8].

Salah satu inovasi terkini yang mencuri perhatian dalam pengembangan API adalah FastAPI. FastAPI adalah sebagai salah satu kerangka pemrograman web Python yang relatif baru dan telah mencuri perhatian secara signifikan. Keunggulannya terletak pada kemampuannya yang luar biasa dalam membuat API dengan cepat dan mudah [9]. FastAPI telah dirancang dengan tujuan menyediakan metode yang sederhana untuk membuat API yang memiliki validasi data otomatis dan dokumentasi yang terintegrasi [10].

Namun, tidak hanya API yang memainkan peran penting dalam pengembangan perangkat lunak modern. *Database* atau basis data juga menjadi unsur kunci dalam ekosistem aplikasi kontemporer. Salah satu sistem basis data NoSQL yang paling fleksibel dan dapat diskalakan adalah MongoDB. MongoDB adalah basis data berbasis dokumen yang memungkinkan penyimpanan data dalam format dokumen BSON (*Binary JSON*) [11], [12]. Ini memberikan fleksibilitas dalam mengelola data yang terus berubah dan berkembang. MongoDB memiliki berbagai manfaat, termasuk kemampuan untuk melakukan *query* data yang kuat, pembuatan indeks otomatis, dan kemampuan untuk mengatasi pertumbuhan skala horizontal dengan mudah [13]–[16].

Integrasi FastAPI dan MongoDB dapat memberikan solusi yang efektif dan efisien untuk pengembangan API. FastAPI menyediakan dukungan yang kuat untuk *Hypertext Transfer Protocol* (HTTP), yang merupakan protokol komunikasi standar untuk API web. MongoDB memungkinkan penyimpanan dan manipulasi data yang efisien dalam format dokumen [17]–[19].

Dalam konteks ini, topik utama yang akan dieksplorasi dalam penelitian ini adalah integrasi antara FastAPI, yang sangat mampu dalam pengembangan API dengan dukungan HTTP yang kuat dan MongoDB yang memungkinkan penyimpanan dan manipulasi data yang efisien. Penelitian ini akan menginvestigasi bagaimana integrasi kedua teknologi ini, termasuk penggunaan HTTP sebagai penghubung yang dapat membantu pengembang dalam menciptakan aplikasi modern yang cepat, efisien, dan responsif, dengan kemampuan untuk menyimpan dan mengelola data dalam format yang fleksibel.

2 METODE

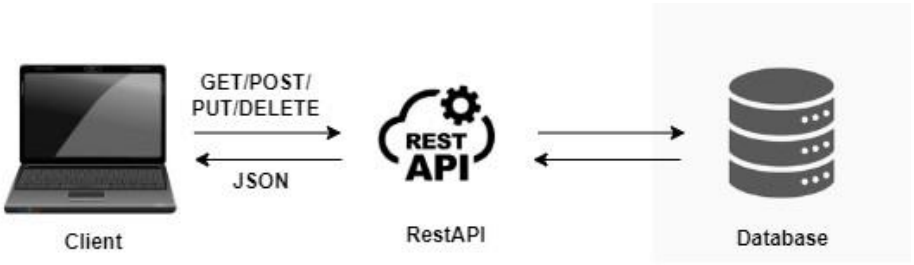
Pada bagian ini mendeskripsikan metode yang digunakan untuk menyelesaikan permasalahan integrasi FastAPI dan MongoDB dalam pengembangan perangkat lunak dengan menerapkan metodologi terstruktur yang dimulai dengan studi literatur untuk memahami konsep dasar, kemudian merancang perangkat lunak dengan skema database MongoDB dan endpoint FastAPI, dilanjutkan dengan implementasi berfokus pada validasi data dan kinerja, dan akhirnya melibatkan evaluasi melalui berbagai pengujian untuk memastikan kinerja dan keamanan yang optimal [14], [20]. Pendekatan ini dirancang untuk memberikan solusi yang efektif dan efisien dalam mengintegrasikan FastAPI, MongoDB, RestAPI dan protokol HTTP, dengan penekanan pada validasi data, keamanan, dan kinerja yang dapat memenuhi kebutuhan aplikasi dengan baik.

2.1 Perancangan Arsitektur Sistem

Dalam perancangan arsitektur sistem pada penelitian ini, menguraikan beberapa aspek penting dalam pengembangan API yang berinteraksi dengan database MongoDB menggunakan FastAPI. Perancangan sistem ini adalah tahap awal yang sangat krusial dalam memastikan bahwa API yang dikembangkan dapat berfungsi dengan baik dan sesuai dengan kebutuhan aplikasi yang diinginkan.

Pada gambar 1 menjelaskan tentang cara kerja komunikasi RESTAPI dengan *database*. *Client* adalah entitas atau sistem yang ingin berinteraksi dengan database melalui RESTAPI. Client mengirim permintaan HTTP ke *endpoint-endpoint* API yang telah dirancang, seperti permintaan *GET, POST, PUT, atau DELETE*, sesuai dengan

aksi yang ingin dilakukan terhadap data dalam database. Permintaan HTTP dari *client* mengandung informasi seperti metode HTTP (*GET, POST, PUT, DELETE*), URI (*Uniform Resource Identifier*) *endpoint* API, dan seringkali juga mengandung data dalam format tertentu jika diperlukan, seperti data pengiriman (*POST* atau *PUT*) atau parameter pencarian (*GET*). API berperan sebagai perantara antara client dan database. Ketika API menerima permintaan dari client, ia akan memproses permintaan tersebut, mengurai informasi yang diberikan, memvalidasi, mengautentikasi pengguna jika diperlukan, dan meneruskan permintaan ke database sesuai dengan operasi yang diminta. API akan berkomunikasi dengan database untuk mengambil atau memodifikasi data sesuai permintaan, dan kemudian mengembalikan respons ke client. Respons dari API berisi status HTTP (seperti *200 OK, 404 Not Found, 500 Internal Server Error*) bersama dengan data yang diminta atau hasil operasi yang dilakukan terhadap database. *Database* adalah penyimpanan data yang sebenarnya. Ini adalah tempat di mana data sebenarnya disimpan, dikelola, dan diakses. Ketika API menerima permintaan untuk mengambil atau memodifikasi data, ia akan berkomunikasi dengan database untuk mengeksekusi operasi yang diminta, seperti mengambil data, menambahkan data baru, memperbarui data yang ada atau menghapus data. *Database* akan mengembalikan hasil operasi ke API, yang selanjutnya akan mengirimkan respons yang sesuai ke client dalam bentuk JSON.



Gambar 1. Komunikasi RESTAPI dengan *Database*

2.2 MongoDB

MongoDB adalah salah satu sistem database NoSQL yang paling populer dan digunakan secara luas di seluruh dunia [14], [21]. MongoDB memanfaatkan model dokumen, yang berarti data disimpan dalam format BSON (*Binary JSON*) yang mirip dengan JSON [22]. Keuntungan utama MongoDB adalah fleksibilitasnya yang tinggi dalam mengelola data yang terstruktur secara dinamis, yang sangat cocok untuk aplikasi modern yang memerlukan penyimpanan data yang dapat berubah-ubah dengan mudah [19], [23]. Kemampuannya untuk berintegrasi dengan bahasa pemrograman lain termasuk Python, membuatnya menjadi pilihan yang ideal dalam pengembangan perangkat lunak yang membutuhkan penyimpanan data yang scalable dan fleksibel [20].

2.3 Representational State Transfer API (RESTAPI)

RESTAPI adalah standar desain arsitektur popular yang sering digunakan dalam pengembangan aplikasi web [6], [7]. Ini berfokus pada sumber daya *(resources)* yang direpresentasikan dengan URL *(Uniform Resource Locator)* dan berkomunikasi melalui metode HTTP (seperti *GET, POST, PUT, DELETE*). RESTAPI mendukung operasi CRUD *(Create, Read, Update, Delete)* dan bekerja dengan baik untuk mengintegrasikan aplikasi dengan perangkat lunak lain serta memungkinkan penggunaan cache dan skalabilitas yang baik [24], [25].

3 HASIL DAN PEMBAHASAN

Pada bagian ini menjelaskan mengenai cara pengembangan aplikasi perangkat lunak dari metode yang digunakan yaitu metode RESTAPI. Hasil pengujian disajikan dalam bentuk gambar serta penjelasan.

3.1 Integrasi FastAPI dan MongoDB

FastAPI menyediakan dukungan untuk MongoDB melalui ekstensi *pymongo*. Ekstensi *pymongo* dapat digunakan untuk membuat koneksi ke database MongoDB, melakukan operasi CRUD, dan mengelola koleksi.

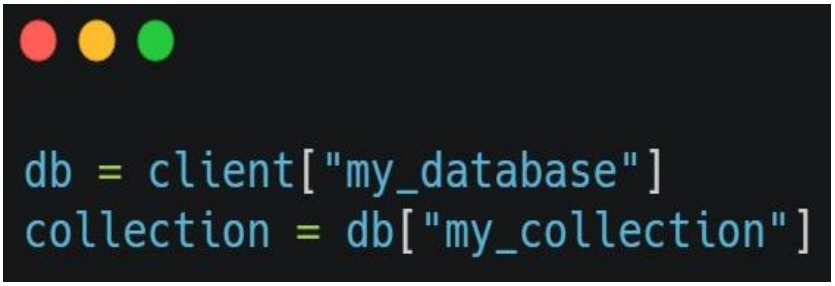
3.2.1 Membuat koneksi dan koleksi ke database MongoDB

Ekstensi *pymongo* menyediakan kelas *MongoClient* yang dapat digunakan untuk membuat koneksi ke *database* MongoDB. Untuk membuat koneksi ke *database* MongoDB, dapat menggunakan metode *connect()* pada

kelas *MongoClient*. Metode *connect()* menerima URL koneksi ke *database* MongoDB sebagai parameter. Berikut adalah contoh kode untuk membuat koneksi ke *database* MongoDB:

Gambar 2. Kode membuat koneksi MongoDB

Kode di atas akan membuat koneksi ke *database* MongoDB yang berjalan pada host localhost dengan port 27017. Setelah koneksi berhasil dibuat, selanjutnya dapat menggunakan objek *client* untuk mengakses *database* dan koleksi dalam *database* tersebut. Berikut adalah contoh kode untuk mengakses *database* dan koleksi dalam *database* MongoDB:

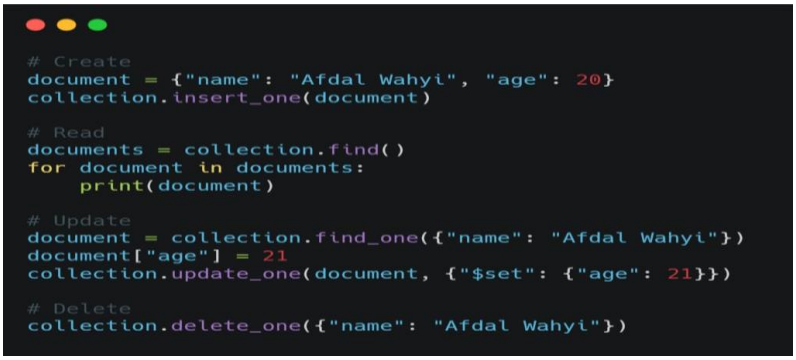


Gambar 3. Kode mengakses *database* dan koleksi dalam MongoDB

Kode di atas akan membuat objek db yang mewakili *database* my_database dan objek collection yang mewakili koleksi my_collection dalam *database* my_database. Objek db dan *collection* dapat digunakan untuk melakukan operasi CRUD pada data dalam d*atabase* MongoDB.

3.2.2 Operasi CRUD

Selanjutnya, membuat objek *collection* untuk melakukan operasi CRUD pada data dalam *database* MongoDB. Berikut adalah contoh kode untuk melakukan operasi CRUD pada data dalam *database* MongoDB:



Gambar 4. Kode untuk melakukan operasi CRUD

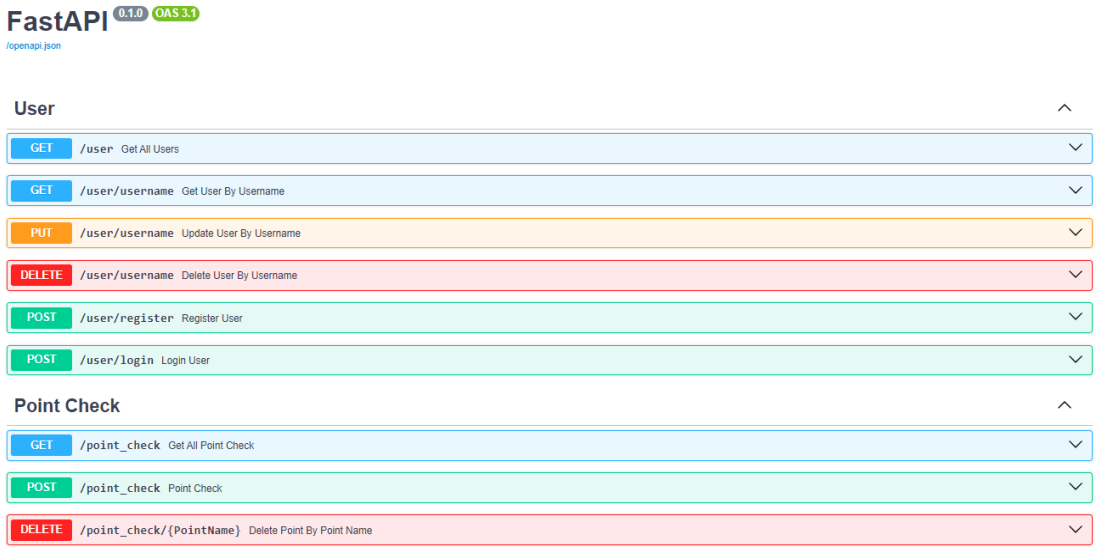
3.2 Evaluasi dan Pengujian

Dalam bagian ini, dilakukan evaluasi dan pengujian transfer data untuk mengevaluasi kehandalan dan efisiensi sistem transfer data yang dikembangkan. Pengujian dilakukan dengan koneksi kabel LAN dan jaringan WIFI.

3.3.1. Evaluasi sistem

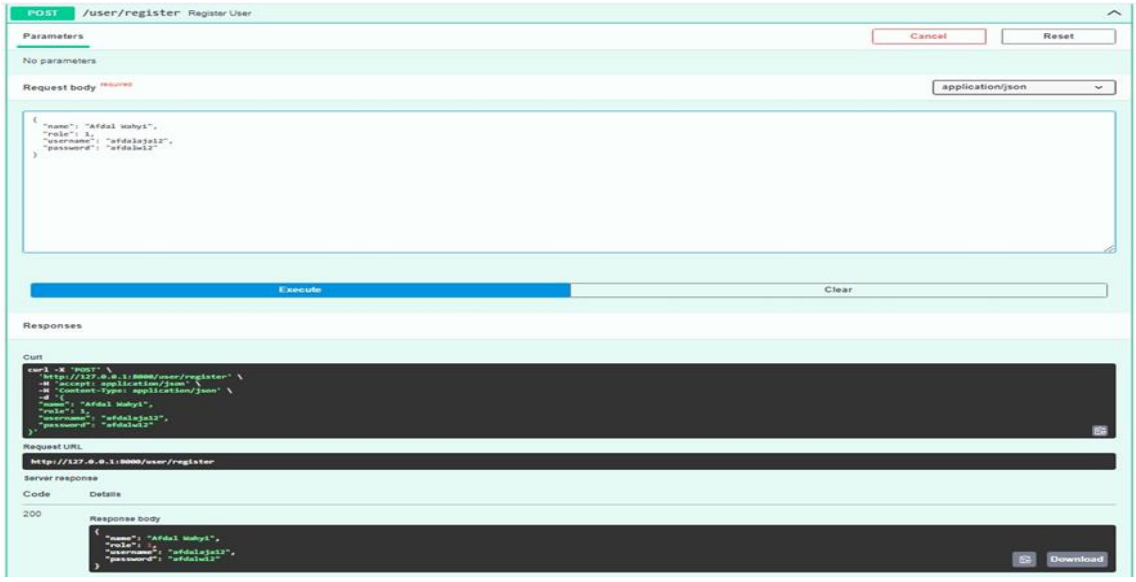
Perancangan desain RESTAPI dalam penelitian ini akan mengandalkan penggunaan FastAPI, yang merupakan sebuah *framework* Python yang efisien dan kuat untuk mengembangkan API web. Selain itu, data yang

dihasilkan oleh API akan disimpan dan dikelola dalam basis data MongoDB, yang merupakan salah satu basis data NoSQL yang terkenal untuk penyimpanan dokumen. Berikut adalah demonstrasi API yang sudah dikembangkan.



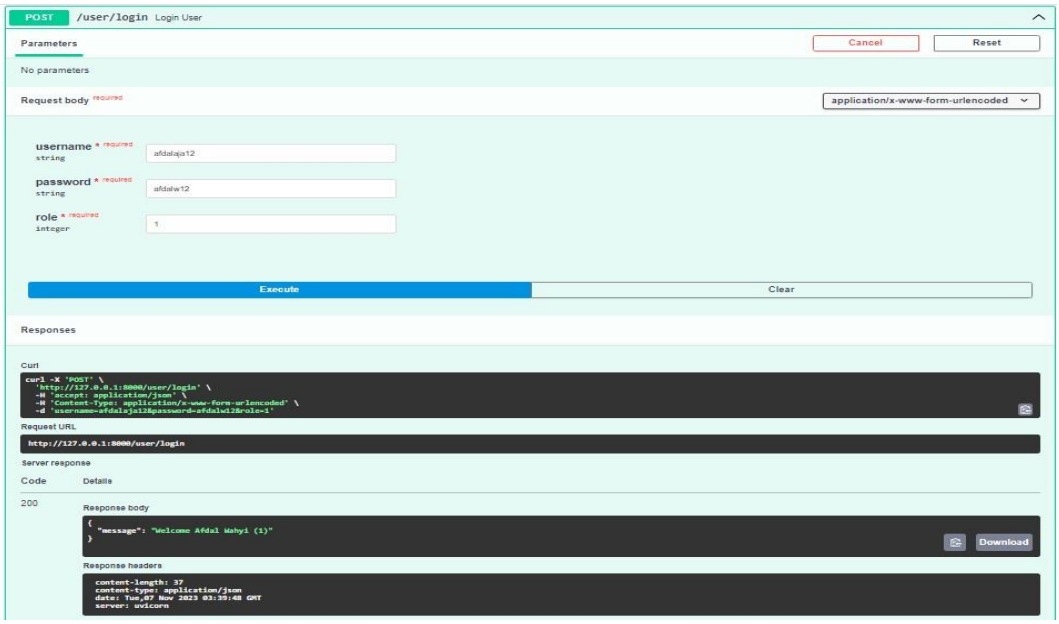
Gambar 5. Desain endpoint API

Pada gambar 5 merupakan desain endpoint API adalah langkah penting dalam perancangan API. *Endpoint-endpoint* API adalah titik akhir yang akan digunakan oleh klien (seperti aplikasi web atau perangkat mobile) untuk berinteraksi dengan sistem yang digunakan. Setiap *endpoint* memiliki metode HTTP tertentu (*GET, POST, PUT, DELETE*) untuk berkomunikasi dengan server.



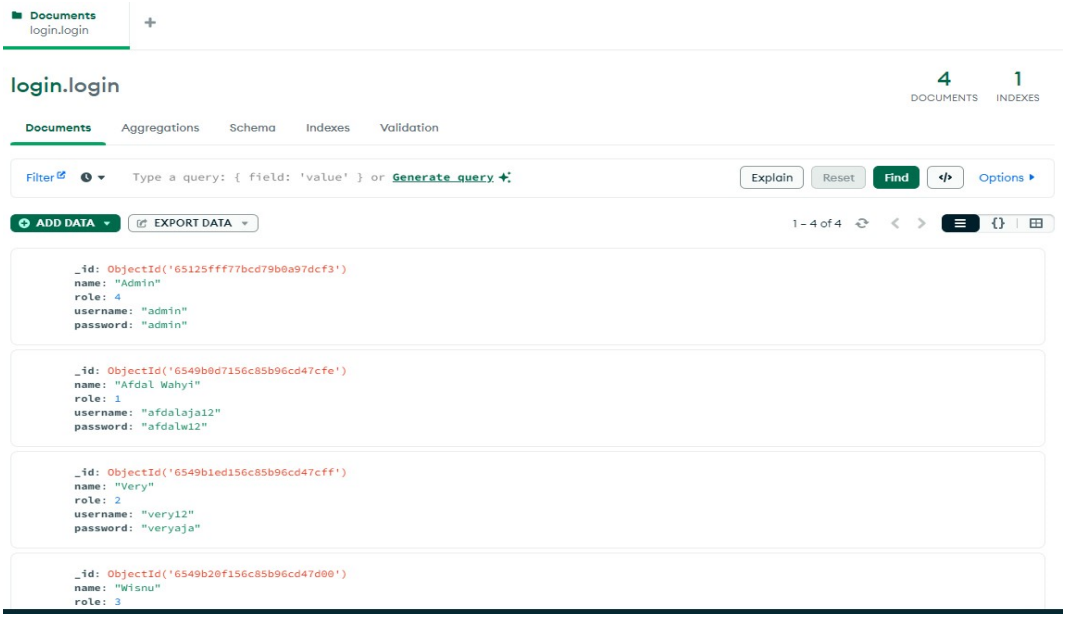
Gambar 6. *Register User*

Pada gambar 6, terdapat implementasi pendaftaran *user* yang menggunakan *method POST* dengan *endpoint* API "/user/register". Dalam proses ini, data yang dikirimkan oleh *user* terdapat tipe data string dan integer. Proses ini dirancang untuk menerima, memproses, dan menyimpan data pengguna ke dalam sistem. Jika proses pendaftaran berhasil dieksekusi, *server* akan memberikan *respons* dengan kode status 200 OK dan mengembalikan data yang sesuai dengan permintaan yang telah dimasukkan oleh *user*.



Gambar 7. *Login User*

Pada gambar 7 merupakan implementasi proses *login user* melalui metode HTTP dengan menggunakan *endpoint* API "/user/login". Proses ini merupakan langkah penting untuk memungkinkan *user* mengakses sistem. Dalam proses ini, user diminta untuk memasukkan tiga informasi utama: *username, password*, dan *role* yang harus sesuai dengan data register sebelumnya. Setelah berhasil *login*, *server* akan memberikan *respons* dengan format "Welcome [nama user] [role]". Contoh respons dari server menampilkan nama user "Afdal Wahyi" dan role "1".

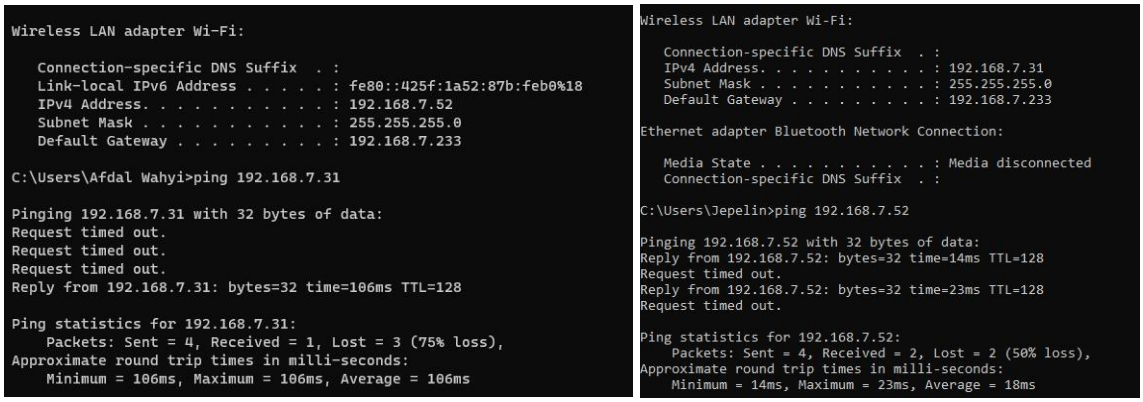


Gambar 8. *Database MongoDB*

Pada gambar 8 adalah implementasi penyimpanan data yang kuat dan andal dengan menggunakan *database* MongoDB. *Database* ini berfungsi sebagai penyimpanan pusat untuk seluruh data yang masuk ke sistem. Setiap kali data baru dimasukkan, baik melalui proses pendaftaran pengguna *(register)* maupun proses *login*, data tersebut akan disimpan secara aman dalam database MongoDB dan dapat diakses untuk interaksi selanjutnya dalam sistem. Implementasi database MongoDB pada gambar 8 merupakan langkah kunci untuk mengelola data pengguna dengan baik dan menjaga integritas data. Database ini mampu menyimpan data dalam format yang fleksibel dan mendukung pengembangan sistem yang andal dan responsif.

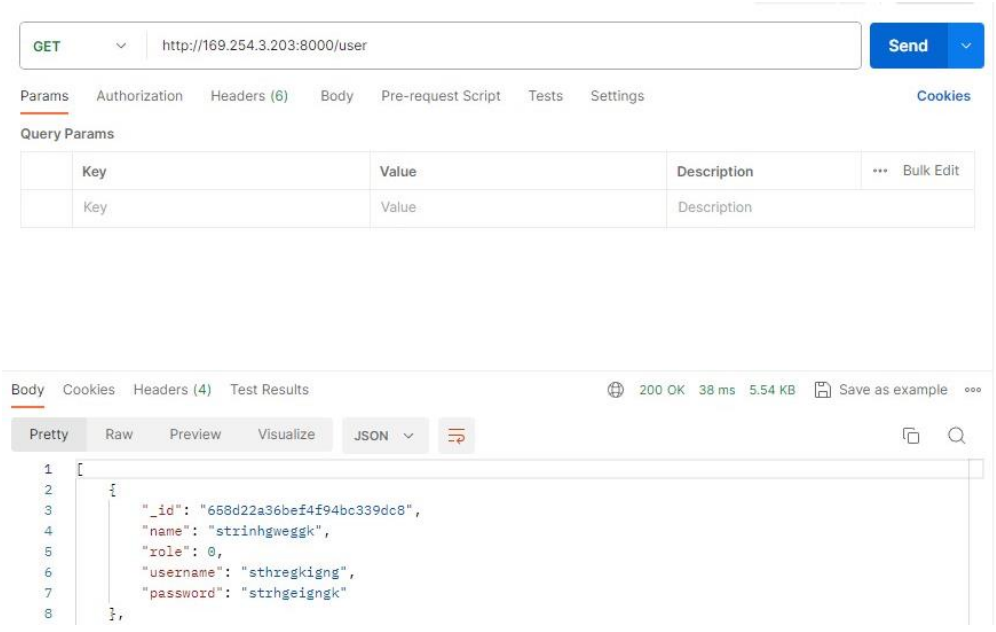
3.3.2. Pengujian sistem

Pada tahap pengujian sistem, dilakukan koneksi antara dua perangkat menggunakan dua metode yang berbeda, yakni melalui jaringan Wi-Fi dan kabel LAN. Proses pengujian ini didukung oleh perangkat lunak Postman, sebuah aplikasi yang umumnya digunakan untuk menguji dan mengelola API.



Gambar 9. Uji coba koneksi antar dua device

Pada gambar 9 menunjukkan momen uji coba koneksi antara dua perangkat. Visualisasi ini membantu memverifikasi keberhasilan koneksi dan memastikan bahwa interaksi antar perangkat terjadi sesuai harapan. Hasil dari pengujian ini mencakup evaluasi stabilitas koneksi, kecepatan transfer data, dan responsivitas sistem terhadap permintaan. Analisis ini memberikan wawasan yang mendalam tentang performa keseluruhan sistem dalam mengelola koneksi antar perangkat. Berikut ini merupakan hasil pengujian yang telah dilakukan.



Gambar 10. Proses pengujian dengan Postman

Pada gambar 10 merupakan tampilan antarmuka pengguna Postman yang digunakan untuk melakukan pengujian dengan metode GET. Proses ini dirancang untuk menampilkan kecepatan waktu respons (response time) dan ukuran data yang diinput. Pengujian dilakukan dengan mengirimkan permintaan API dengan variasi ukuran data di bawah dua kondisi koneksi yang berbeda, yaitu menggunakan jaringan WiFi dan kabel LAN.

Tabel 1. Pengujian menggunakan jaringan WiFi

Testing	Samples	Data Size(B)	Time Response(ms)
Testing 1	5	694	1658
Testing 2	10	1240	332
Testing 3	15	1820	334
Testing 4	20	2400	1040
Testing 5	25	2980	3058
Testing 6	30	3510	3004
Testing 7	35	4040	1105
Testing 8	40	4580	171
Testing 9	45	5110	4008
Testing 10	50	5670	2039
Average			1674.9

Tabel 1 menunjukkan hasil pengujian respons API di bawah kondisi jaringan WiFi. Sepuluh pengujian dilakukan dengan variasi ukuran data, dan parameter kinerja yang diamati meliputi ukuran data (byte) dan waktu respons (ms).

Tabel 2. Pengujian menggunakan kabel LAN

Testing	Samples	Data Size(B)	Time Response(ms)
Testing 1	5	709	54
Testing 2	10	1210	72
Testing 3	15	1740	34
Testing 4	20	2280	17
Testing 5	25	2820	17
Testing 6	30	3360	24
Testing 7	35	3890	36
Testing 8	40	4480	21
Testing 9	45	5010	21
Testing 10	50	5540	38
Average			33.4

Tabel 2 menampilkan hasil pengujian respons API menggunakan koneksi kabel LAN. Sama seperti pada pengujian WiFi, parameter kinerja yang diamati adalah ukuran data dan waktu respons. Data tersebut memberikan pemahaman mendalam tentang efisiensi sistem dalam menangani permintaan dengan variasi ukuran data, menggambarkan sejauh mana kinerja optimal dapat dipertahankan melalui kabel LAN.

Pada tabel 1 dan 2 terlihat bahwa respons API di bawah kabel LAN secara konsisten menunjukkan waktu respons yang lebih rendah dengan rata-rata 33.4 ms dibandingkan jaringan WiFi dengan rata-rata 1674.9 ms. Hal ini menandakan bahwa implementasi API dan database mampu memberikan kinerja yang lebih baik ketika diakses melalui koneksi kabel LAN. Selain itu, terdapat kecenderungan waktu respons yang efektif meskipun ukuran data meningkat di bawah kabel LAN, menandakan efisiensi dan stabilitas dalam menangani permintaan dengan ukuran data yang bervariasi.

4 KESIMPULAN

Dalam pengembangan aplikasi berbasis online dan seluler, integrasi antara FastAPI dan MongoDB telah terbukti memberikan solusi efektif dan efisien. FastAPI, sebagai kerangka pemrograman web Python yang efisien, memungkinkan pengembangan API dengan validasi data otomatis dan dokumentasi terintegrasi. Di sisi lain,

MongoDB sebagai basis data NoSQL yang fleksibel memberikan solusi penyimpanan data yang dapat berkembang dan mampu mengatasi pertumbuhan skala horizontal.

Penelitian ini berhasil mengintegrasikan FastAPI dengan MongoDB, menggambarkan cara membuat koneksi antara FastAPI dan MongoDB, serta melibatkan operasi CRUD pada data. Hasil pengujian menunjukkan bahwa waktu respons menggunakan koneksi kabel LAN yang lebih rendah dengan rata-rata 33.4 ms dibandingkan jaringan WiFi dengan rata-rata 1674.9 ms. Sehingga integrasi ini mampu menghasilkan aplikasi yang responsif, efisien, dan mampu menangani data yang terus berkembang pesat.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk memberikan arahan kepada penulis dalam penyusunan jurnal dan penyelesaian tugas akhir, serta Politeknik Negeri Batam terkhusus *Barelang Robotic and Artificial Intelligence Lab* (BRAIL) yang telah memfasilitasi dalam melakukan penelitian.

DAFTAR PUSTAKA

[1] M. Danuri, M. Informatika, J. Teknologi, and C. Semarang, "PERKEMBANGAN DAN TRANSFORMASI TEKNOLOGI DIGITAL."

[2] A. Afandi, "POTENSI APLIKASI ZOOM CLOUD MEETINGS DALAM PEMBELAJARAN DI ERA DIGITAL Developing High Order Thinking Stimulation Model for Pre-Service Teachers' Science Education View project." [Online]. Available: https://www.researchgate.net/publication/343859632

[3] R. Yusuf Azhari, "Web Service Framework : flask dan fastAPI," 2022. [Online]. Available: https://jurnal.universitasputrabangsa.ac.id/index.php/tiij

[4] A. C. Wijaya, I. Gede, A. Wibawa, D. I. Dewa, M. Bayu, and A. Darmawan, "PENGEMBANGAN RESTFUL API UNTUK MODEL MACHINE LEARNING INDOOR-OUTDOOR DALAM APLIKASI PEMINJAMAN RUANGAN."

[5] B. Di Martino, A. Posillipo, S. Nacchia, and S. A. Maisto, "A Q&A tool to produce an Ad-Hoc OpenAPI specification to identify equivalent REST Api Services," in *Proceedings - 2018 IEEE International Conference on Smart Computing, SMARTCOMP 2018*, Institute of Electrical and Electronics Engineers Inc., Jul. 2018, pp. 375–380. doi: 10.1109/SMARTCOMP.2018.00032.

[6] "Rest API Aplikasi Weshare".

[7] J. Yasmin, Y. Tian, and J. Yang, "A First Look at the Deprecation of RESTful APIs: An Empirical Study," in *Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 151–161. doi: 10.1109/ICSME46990.2020.00024.

[8] A. A. Prayogi, M. Niswar, Indrabayu, and M. Rijal, "Design and Implementation of REST API for Academic Information System," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1757-899X/875/1/012047.

[9] J. Chen, "Model Algorithm Research based on Python Fast API".

[10] D. V. Kornienko, S. V. Mishina, S. V. Shcherbatykh, and M. O. Melnikov, "Principles of securing RESTful API web services developed with python frameworks," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Nov. 2021. doi: 10.1088/1742-6596/2094/3/032016.

[11] A. J. Maulidin, F. Renaldi, and F. R. Umbara, "Online Integration of SQL and No-SQL Databases using RestAPIs: A Case on 2 furniture e-Commerce Sites," in *2020 3rd International Conference on Computer and Informatics Engineering, IC2IE 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 261–266. doi: 10.1109/IC2IE50715.2020.9274613.

[12] R. Andreoli, T. Cucinotta, and D. B. De Oliveira, "Priority-Driven Differentiated Performance for NoSQL Database-As-a-Service," *IEEE Transactions on Cloud Computing*, 2023, doi: 10.1109/TCC.2023.3292031.

[13] G. Arganata, E. Sakti Pramukantoro, and W. Yahya, "Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS Untuk Menyimpan Data Yang Beragam Dari Node Sensor," 2018. [Online]. Available: http://j-ptiik.ub.ac.id

[14] B. Cahyo Santoso, Y. Natasya, S. Willian, and F. Alfando, "Tinjauan Pustaka Sistematis terhadap Basis Data MongoDB."

[15] C. Aminuddin Hamka and T. Bharata Adji dan Selo Sulistyo, "Edu Komputika Journal Evaluasi Metode Load Balancing dan Fault Tolerance pada Sistem Database Server Aplikasi Chat," 2018. [Online]. Available: http://journal.unnes.ac.id/sju/index.php/edukom

[16] A. Halimi, A. Sudarmanto, and E. Utami, "ANALISIS PERBANDINGAN KINERJA WAKTU RESPON MYSQL 8.0 DAN NOSQL MONGODB MENGUNAKAN RESTAPI NODEJS PADA STUDI KASUS KELAS ONLINE".

[17] J. D. Merrick *et al.*, "CoRL: Environment Creation and Management Focused on System Integration," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.02182

[18] R. Munadi, M. Purnandi, and T. Y. Arif, "Evaluasi Teknik Penyadapan Lalu Lintas Data Dengan Session Hijacking Pada Protokol HTTP," *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, vol. 14, no. 2, p. 58, Sep. 2019, doi: 10.30872/jim.v14i2.1798.

[19] H. Ingo and D. Daly, "Automated system performance testing at MongoDB," in *Proceedings of the Workshop on Testing Database Systems, DBTest 2020*, Association for Computing Machinery, Inc, Jun. 2020. doi: 10.1145/3395032.3395323.

[20] I. Kadek Krisna Dwi Payana, L. Arida Ayu Rahning Putri, and J. Raya Kampus Unud, "Implementasi Sistem Analytics pada Produk menggunakan OLAP berbasis MongoDB," 2022.

[21] A. Alawini, L. Zhou, L. Kang, P. Rao, and P. C. Ho, "Teaching Data Models with TriQL," in *Proceedings of the 1st ACM SIGMOD International Workshop on Data Systems Education: Bridging Education Practice with Education Research, DataEd 2022*, Association for Computing Machinery, Inc, Jun. 2022, pp. 16–21. doi: 10.1145/3531072.3535320.

[22] B. Cherry, P. Benats, M. Gobert, L. Meurice, C. Nagy, and A. Cleve, "Static Analysis of Database Accesses in MongoDB Applications," pp. 1–5, 2022, doi: 10.1109/SANER2022.2022.00111.

[23] A. Septia Maharani *et al.*, "PERANCANGAN DATA BASE KASIR DAN PERSEDIAAN BARANG MENGGUNAKAN MONGODB," 2022.

[24] H. Ed-Douibi, J. L. Canovas Izquierdo, and J. Cabot, "Automatic generation of test cases for REST APIs: A specification-based approach," in *Proceedings - 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference, EDOC 2018*, Institute of Electrical and Electronics Engineers Inc., Nov. 2018, pp. 181–190. doi: 10.1109/EDOC.2018.00031.

[25] P. Sinha* and K. A. Kumar, "REST APIs for Emerging Social Media Platforms," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 5, pp. 652–659, Mar. 2020, doi: 10.35940/ijitee.E2608.039520.