Perancangan Dan Implementasi Basis Data Berbasis Dokumen Terdistribusi Pada Pengembangan Produk Coofis Verse Di Pt. Arm Solusi

1st Salsa Tiffani Rahmadona
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
salsatiffanirahmadon@student.telkomuniversity.ac.id

2st Muhammad Iqbal
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
miqbal.staff.telkomuniversity.ac.id

3st Taufik Sulaeman Puspanegara
PT. ARM Solusi
Bandung, Indonesia
taufiksu@gmail.com

Abstrak **- PT. ARM Solusi adalah perusahaan teknologi yang berfokus pada big data, data analitik, kolaborasi, otomatisasi administrasi, aplikasi integrasi, dan API. Mereka mengotomasikan proses administrasi secara paperless melalui produk *Collaboration Office* Nota Dinas Elektronik (Coofis NDE), yang mengelola tata persuratan secara elektronik. Untuk meningkatkan efisiensi dan skalabilitas, PT. ARM Solusi mengembangkan Coofis NDE menjadi Coofis Verse menggunakan arsitektur *microservice*. Arsitektur *monolithic* sebelumnya memiliki keterbatasan dalam skalabilitas, fleksibilitas, dan pemeliharaan. *Microservice* memecah aplikasi menjadi layanan kecil yang bisa dioperasikan dan dikembangkan secara mandiri. MongoDB, sebagai *database* berbasis dokumen non-relasional, digunakan untuk menyimpan data dalam format JSON yang fleksibel, cocok untuk lingkungan *microservice*. Hasil perancangan menunjukkan bahwa arsitektur *microservice* dan MongoDB meningkatkan efisiensi, skalabilitas, dan fleksibilitas sistem Coofis Verse. Layanan dalam Coofis Verse dapat berkembang independen tanpa terpengaruh perubahan skema data layanan lain. MongoDB mendukung pemulihan otomatis, memastikan ketersediaan dan konsistensi data tinggi dalam lingkungan terdistribusi, menjadikan sistem lebih tangguh dan responsif terhadap kebutuhan pengguna.**

Kata Kunci **- PT. ARM Solusi, Coofis, *Microservice*, Basis Data Berbasis Dokumen, MongoDB**

I. PENDAHULUAN

Di era digital, teknologi informasi dan komunikasi mempengaruhi hampir semua aspek kehidupan, termasuk administrasi dan tata kelola pemerintahan. Implementasi teknologi dalam administrasi publik dan manajemen surat menyurat berkembang pesat [1]. Salah satu inovasi adalah aplikasi tata persuratan berbasis *web* untuk mengelola surat-menyurat dalam perusahaan atau institusi, mencakup pembuatan, pengelolaan, penyimpanan, hingga pengarsipan surat.

PT. Andal Rancang Multi Solusi (PT. ARM Solusi) adalah perusahaan teknologi yang fokus mengotomasikan proses administrasi secara *paperless* melalui produk *Collaboration Office* Nota Dinas Elektronik (Coofis NDE). Namun, Coofis NDE saat ini masih menggunakan arsitektur *monolithic* yang kurang efisien dan memiliki keterbatasan dalam skalabilitas, fleksibilitas, serta pemeliharaan. Untuk mengatasi hal ini, PT. ARM Solusi mengembangkan Coofis NDE menjadi Coofis Verse dengan arsitektur *microservice*, yang memecah aplikasi menjadi beberapa layanan kecil yang dapat dioperasikan dan dikembangkan secara mandiri, meningkatkan fleksibilitas, skalabilitas, dan kemudahan dalam pengembangan serta pemeliharaan aplikasi [2].

Penggunaan *database* berbasis dokumen non-relasional seperti MongoDB sangat penting dalam pengembangan ini. MongoDB, dengan format dokumen JSON yang fleksibel, cocok untuk lingkungan *microservice*. MongoDB memungkinkan setiap layanan dalam Coofis Verse memiliki skema data yang berbeda dan tersimpan dalam koleksi yang sesuai, memungkinkan perkembangan independen tanpa terpengaruh oleh perubahan skema data layanan lain. MongoDB juga mendukung replikasi dan pemulihan otomatis, memastikan ketersediaan dan konsistensi data yang tinggi. Dengan demikian, transisi ke arsitektur *microservice* dengan MongoDB meningkatkan efisiensi, skalabilitas, dan fleksibilitas Coofis Verse secara signifikan [3].

II. KAJIAN TEORI

A. *Microservices*

Microservice adalah suatu desain infrastruktur yang dapat digunakan untuk membuat suatu aplikasi menjadi terdiri dari berbagai layanan sendiri tetapi tetap saling terhubung satu sama lain. *Microservice* ini biasanya bertujuan untuk meningkatkan fleksibilitas, skalabilitas, dan efisiensi dalam pengembangan perangkat lunak.

Tidak seperti aplikasi *monolithic* , *microservice* membagi aplikasi menjadi layanan yang lebih kecil yang saling

terhubung dan lebih terukur. Arsitektur *microservice* merupakan alternatif arsitektur yang lebih fleksibel dan terukur. Pada arsitektur *microservice*, sistem informasi dirancang untuk terdistribusi dan menyediakan layanan secara lebih fokus dan spesifik [4].

Setiap layanan dalam arsitektur *microservice* memiliki skala yang lebih kecil dibandingkan dengan aplikasi *monolithic* , dan dapat meningkatkan produktivitas pengembangan dan pemeliharaan aplikasi. Sehingga ketika menggunakan arsitektur *microservice* dan terjadi suatu masalah pada salah satu layanan, hanya layanan yang bermasalah saja yang dilakukan perbaikan dan tidak akan berpengaruh pada layanan yang lain.

B. Basis Data *(Database)*

Basis data adalah kumpulan data terorganisir yang memungkinkan penyimpanan, akses, dan pengelolaan data secara efisien. Basis data mendukung penyimpanan data dalam jumlah besar dengan struktur yang memungkinkan akses cepat dan akurat, dikelola oleh *Database Management System* (DBMS). DBMS menyediakan antarmuka antara pengguna atau aplikasi dan data, serta bertanggung jawab atas penyimpanan, pemulihan, dan pengelolaan transaksi data, memastikan integritas dan keamanan data. Basis data dapat diakses oleh berbagai departemen dan pengguna dalam perusahaan, mendukung analisis dan pemodelan data [5].

Ada berbagai jenis basis data sesuai kebutuhan dan karakteristik data. Basis data relasional adalah yang paling umum, menyimpan data dalam tabel-tabel yang saling berhubungan melalui kunci primer dan kunci asing. Keuntungan utama basis data relasional adalah kemampuannya menangani *query* kompleks dan menjaga konsistensi data melalui normalisasi. Sementara itu, basis data non-relasional atau NoSQL seperti MongoDB, Cassandra, dan CouchDB dikembangkan untuk menangani jenis data yang lebih beragam dan kebutuhan skala besar yang tidak dapat dipenuhi oleh basis data relasional. NoSQL mendukung penyimpanan data dalam bentuk dokumen, key-value, graf, atau kolom lebar, memungkinkan penyimpanan data semi-terstruktur atau tidak terstruktur [6].

C. NoSQL

Istilah NoSQL pertama kali diciptakan oleh Carlo Strozzi pada tahun 1998 untuk merujuk pada basis data non-relasional. Pada tahun 2009, Eric Evans kembali memperkenalkan istilah ini. NoSQL adalah jenis basis data yang tidak memerlukan skema tetap dan tidak memiliki relasi antar tabel, sehingga sering disebut sebagai basis data non-relasional. Semua dokumen dalam NoSQL berbentuk JSON yang mudah dibaca dan dipahami. NoSQL *database* adalah sistem manajemen data non-relasional yang tidak memerlukan skema tetap. Istilah ini merupakan singkatan dari "non-SQL" atau "not only SQL". Non-relational *database* adalah jenis basis data yang tidak menggunakan model tabel tradisional seperti yang ada pada basis data relasional. Basis data ini menyimpan data dalam format yang lebih fleksibel, seperti *graph-based*, *column-based*, *key-value*, dan *document-oriented* [7].

D. Document Oriented Database

Document-oriented database (basis data berbasis dokumen) adalah salah satu jenis basis data NoSQL yang dirancang untuk menyimpan, mengambil, dan mengelola data sebagai koleksi dokumen. Basis data ini menonjol karena kemampuannya untuk mengelola dokumen-dokumen yang berisi data semi-terstruktur atau tidak terstruktur, seperti dokumen dalam format JSON atau BSON. Dokumen-dokumen ini tidak memiliki skema yang ketat, memungkinkan fleksibilitas dalam perubahan struktur data seiring waktu tanpa memerlukan perubahan pada seluruh basis data. Pendekatan *"schema-on-read"* sering kali digunakan, di mana skema data dapat ditafsirkan saat data dibaca daripada saat data disimpan. Dengan demikian, *document-oriented database* cocok digunakan untuk aplikasi modern yang memerlukan adaptabilitas tinggi terhadap perubahan struktur data dan performa tinggi dalam pengambilan data berbasis dokumen [8].

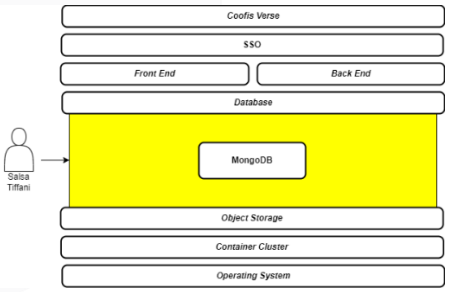
E. MongoDB

MongoDB adalah basis data NoSQL berbasis dokumen yang dirancang untuk menyimpan data dalam volume besar. Setiap basis data di MongoDB terdiri dari koleksi, yang masing-masing berisi berbagai dokumen. Dokumen-dokumen ini dapat berbeda satu sama lain dalam hal jumlah dan jenis bidang yang dimilikinya, serta ukurannya. Struktur dokumen dalam MongoDB mirip dengan cara pengembang membangun kelas dan objek dalam bahasa pemrograman mereka, di mana struktur data terdiri dari pasangan kunci-nilai yang jelas.

Tidak seperti basis data relasional tradisional yang memerlukan skema yang telah ditentukan sebelumnya, MongoDB memungkinkan fleksibilitas yang lebih besar karena dokumen-dokumen tidak perlu mengikuti skema tertentu. Bidang dapat ditambahkan secara dinamis sesuai kebutuhan. Model data di MongoDB memungkinkan representasi hubungan hierarkis, penyimpanan *array*, dan struktur kompleks lainnya dengan lebih mudah [9].

III. METODE

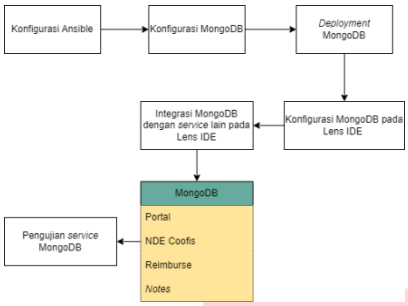
A. Arsitektur Coofis Verse



GAMBAR 1
Blok Diagram Sistem Berbasis Dokumen

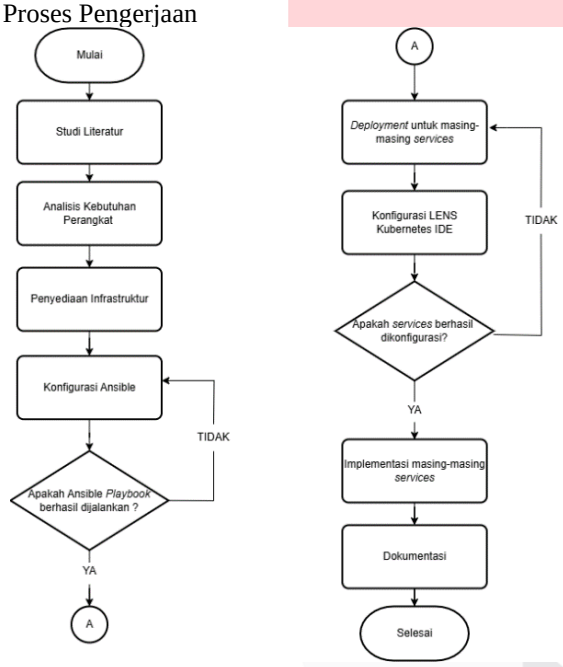
Pada Gambar terdiri dari beberapa tahapan perancangan *service database* MongoDB. Pada gambar tersebut dapat dilihat perancangan yang pertama adalah melakukan konfigurasi Ansible yang berupa konfigurasi mengenai penginstalan dari *tools* yang digunakan dalam pengerjaan proyek akhir ini, yaitu MongoDB. Selanjutnya, proses konfigurasi MongoDB, dimana pada tahap ini dilakukan pengaturan parameter-parameter utama MongoDB untuk memastikan kinerja dan keamanan yang optimal. Lalu, pada tahap *deployment* MongoDB, dilakukan persiapan dengan membuat beberapa file .yaml yang akan dikonfigurasikan ke dalam Lens. Selanjutnya, dilakukan tahap konfigurasi

MongoDB pada Lens, yaitu dengan melakukan ekspor terhadap file .yaml yang dibuat di tahap sebelumnya meliputi file *deployment*, file *service*, dan file ingress. Pada tahap selanjutnya, dilakukan integrasi dengan *service* lain yaitu dengan *service* aplikasi *web* dan *service* penyimpanan objek MinIO.



GAMBAR 2
Blok Diagram Pengerjaan

B. Proses Pengerjaan



GAMBAR 3
Flowchart Proyek Akhir

Pada tahap pertama, dilakukan studi literatur untuk mencari referensi serta melakukan analisis untuk mengidentifikasi kebutuhan berbagai *services* dari produk Coofis Verse. Tahapan ini melibatkan pengumpulan, peninjauan, dan analisis berbagai sumber informasi yang relevan dengan topik penelitian.

Tahap kedua, dilakukan analisis kebutuhan perangkat yang bertujuan untuk mengidentifikasi semua kebutuhan teknis dan non-teknis yang diperlukan untuk mengembangkan dan menjalankan produk aplikasi dengan sukses.

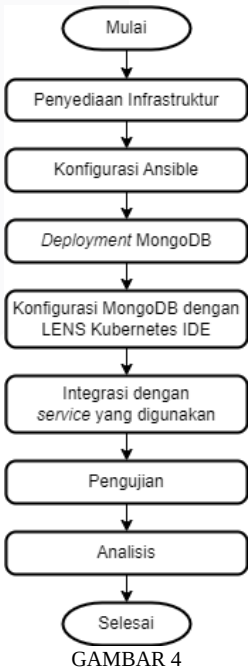
Tahap ketiga, dilakukan penyediaan infrastruktur yang mencakup pengaturan dan penyiapan semua komponen teknis yang diperlukan untuk mendukung pengembangan, pengujian, dan penerapan aplikasi. Ini meliputi penyiapan *server* fisik atau *virtual*, instalasi sistem operasi Linux, dan konfigurasi jaringan.

Tahap keempat, dilakukan konfigurasi Ansible yang digunakan untuk mengatur dan mengelola infrastruktur aplikasi secara otomatis. Pada tahap ini, proses yang dilakukan adalah menulis *playbook* Ansible, yang merupakan file konfigurasi dalam format YAML, untuk mendefinisikan tugas-tugas yang harus dijalankan pada *server* target. *Playbook* ini mencakup perintah untuk menginstal perangkat lunak, mengatur konfigurasi jaringan, mengelola pengguna, dan menerapkan aplikasi. Apabila Ansible *Playbook* berhasil dijalankan maka akan dilanjutkan dengan tahap *Deployment* untuk masing-masing *services*, sedangkan jika tidak berhasil maka kembali ke tahap konfigurasi untuk memeriksa Ansible *Playbook*

Tahap kelima, dilakukan proses dimana LENS, sebuah *Integrated Development Environment* (IDE) khusus untuk Kubernetes, digunakan untuk mengelola dan memantau kluster Kubernetes dengan lebih efisien. Setelah LENS terhubung ke kluster Kubernetes, antarmuka grafisnya dapat dimanfaatkan untuk memantau berbagai aspek kluster, seperti *node*, *pod*, *services*, dan *deployment*s. LENS menyediakan visualisasi yang intuitif dari sumber daya Kubernetes, memungkinkan pengguna untuk melihat status, metrik kinerja, dan log aplikasi secara real-time. Jika *services* tidak berhasil dikonfigurasi makan akan kembali ke tahap *Deployment* untuk memeriksa proses integrasi masing-masing *services*.

Tahap keenam, yaitu implementasi *services* dalam pembuatan aplikasi adalah proses dimana layanan-layanan inti yang mendukung fungsionalitas aplikasi dirancang, dikembangkan, dan diintegrasikan ke dalam sistem. Pada tahap ini, dilakukan penyusunan komponen *backend* yang menyediakan berbagai layanan, seperti autentikasi pengguna, pengelolaan data, penyimpanan objek, serta layanan API *(Application Programming Interface)* yang memungkinkan interaksi antara *frontend* dan *backend* aplikasi.

C. *Deployment*



GAMBAR 4
Flowchart Deployment

Pada tahap pertama, dilakukan penyediaan infrastruktur yang meliputi beberapa hal seperti penyiapan *server* fisik atau memastikan *server virtual* machine sudah berjalan, instalasi

sistem operasi Linux, instalasi *tools* otomatisasi Ansible dan konfigurasi jaringan.

Tahap kedua, dilakukan konfigurasi pada Ansible. Konfigurasi ini meliputi beberapa *command* yang berisikan perintah untuk melakukan instalasi pada *tools* yang nantinya akan digunakan dalam pengerjaan proyek akhir MongoDB yang kemudian ansible akan dijalankan pada proses *operations* menggunakan ansible *playbook*.

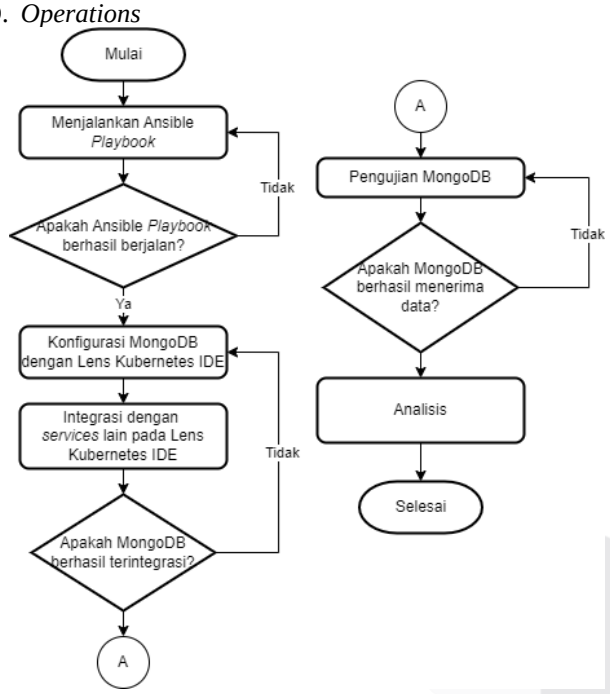
Tahap ketiga, dilakukan *deployment* terhadap MongoDB. Hal yang dilakukan pada langkah ini yaitu mempersiapkan source code untuk di ekspor, serta IP Address yang akan dikonfigurasikan pada Lens Kubernetes.

Tahap keempat, dilakukan konfigurasi MongoDB pada *tools* Lens Kubernetes IDE dengan melakukan ekspor microk8s config yang telah disiapkan pada langkah sebelumnya.

Tahap kelima, dilakukan integrasi terhadap *service* MongoDB sebagai platform yang digunakan untuk membangun basis data berbasis dokumen terdistribusi.

Tahap keenam dan ketujuh, adalah melakukan pengujian dan analisis dengan fokus khusus pada pengujian kinerja MongoDB.

D. *Operations*



GAMBAR 5
Flowchart Operations

Pada tahap pertama, melakukan running terhadap ansible yang telah dikonfigurasi pada tahap *deployment*. Running ansible, dilakukan dengan menggunakan *command* "ansible-*playbook* -I <file inventory> <nama .yaml yang akan dijalankan>.

Tahap kedua, memastikan terlebih dahulu apakah ansible yang telah dijalankan sudah berhasil atau belum. Jika pada tahap ini belum berhasil menjalankan ansible, maka perlu dilakukan konfigurasi ulang pada proses *deployment* serta melakukan running ulang ansible.

Tahap ketiga, melakukan *git clone repositiry*. Pada tahap ini, terdapat beberapa *repositiry* yang perlu dilakukan cloning yaitu *repositiry* front end serta back end.

Tahap keempat dan kelima, melakukan *build image frontend*, *backend*, *notes*, *portal*, *reimburse* menggunakan Docker. Dalam proses *build image* memiliki range waktu yang berbeda, tergantung pada konfigurasi Dockerfile yang dibuat. Jika proses nya berhasil maka, selanjutnya yang dilakukan adalah melakukan push *image*.

Tahap keenam dan ketujuh, melakukan push *image*. *Image* yang telah dibuat, selanjutnya akan disimpan pada *image registry*. Pada pengerjaan proyek akhir ini, *image registry* yang digunakan yaitu Docker Hub. Jika proses push *image* gagal, maka perlu dilakukan konfigurasi ulang pada proses push *image*-nya.

Tahap kedelapan, dilakukan konfigurasi pada *tools* Lens Kubernetes IDE dengan dengan membutuhkan file .yaml seperti *deployment*, *service*, ingress, *endpoints* untuk membuat container.

Tahap kesembilan, melakukan integrasi dengan *service* yang lain pada Lens Kubernetes. Tahap ini dilakukan create resource menggunakan file .yaml yang telah disiapkan pada tahap sebelumnya.Setelah dilakukan integrasi, selanjutnya adalah harus memastikan dahulu apakah progress integrasinya berhasil atau tidak.

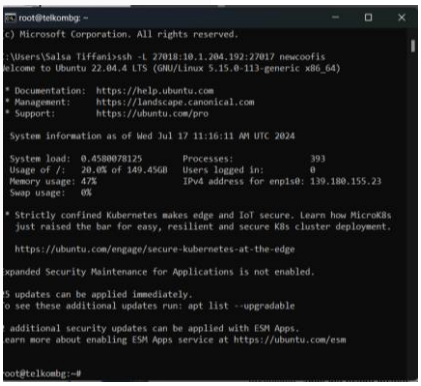
Tahap kesepuluh dan kesebelas melakukan pengujian dan analisis dengan fokus khusus pada pengujian kinerja MongoDB.

IV. HASIL DAN PEMBAHASAN

Pada tahap ini membahas mengenai hasil perancangan dan pengujian terhadap basis data MongoDB yang digunakan dalam implementasi untuk pengembangan produk Coofis Verse di PT ARM Solusi.

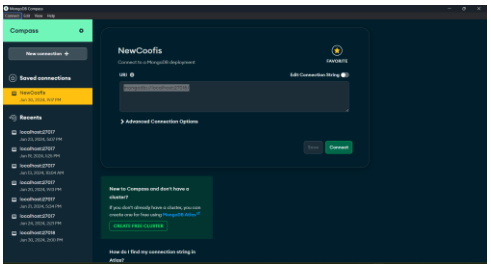
A. Hasil Perancangan MongoDB

Pertama membuat *tunnel* SSH yang mengarah ke port local 27018 yang diteruskan ke port 27017 di 10.1.204.192 di *server* newcoofis.



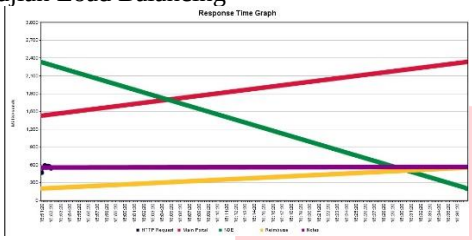
GAMBAR 6
Hasil Perancangan MongoDB

Menunjukan tampilan MongoDB Compass dengan *command* "Mongodb:*localhost*:27018/" yang digunakan untuk menghubungkan ke instans MongoDB yang berjalan di *localhost* pada port 27018.



GAMBAR 7
Tampilan MongoDB Compass

B. Pengujian Load Balancing



GAMBAR 8
Pengujiam Load Balancing

- Performa Terbaik : NDE memiliki *repon time* rata-rata terendah (210 ms) dan variabilitas terendah (27.37 ms), menunjukkan stabilitas dan kinerja yang konsisten.
- Performa Terburuk : Main Portal memiliki waktu respon rata-rata tertinggi (2666 ms) dan variabilitas yang signifikan (796.63 ms), menunjukkan yang besar dalam kinerjanya.
- Throughput dan Transfer Data : Meskipun Main Portal memiliki throughput data tertinggi, total throughput dari semua label mencapai 12.92157 permintaan per detik, menunjukkan kinerja sistem yang baik secara keseluruhan.

B. Pengujian Fungsionalitas

TABEL 1
Pengujiam Fungsionalitas

Skema Pengujian	Tujuan Pengujian	Hasil
Instalasi	Memastikan MongoDB terinstal dengan benar pada sistem	Instalasi MongoDB pada sistem berhasil dan berfungsi dengan baik. Layanan MongoDB berjalan dengan lancar, dan pengujian fungsionalitas dasar menunjukkan bahwa basis data dapat dibuat, data dapat dimasukkan, dan dokumen dapat dibaca dengan sukses.
Pengujian CRUD *(Create, Read, Update, Delete)*	Memastikan bahwa operasi dasar pada basis data berfungsi dengan benar.	*Create* pada basis data dapat disimpan dengan benar. *Read* pada basis data yang disimpan dapat diambil kembali dengan benar. *Update* pada basis data data yang ada dapat diperbarui tanpa kesalahan. *Delete* pada basis data yang tidak diperlukan dapat dihapus dengan benar.
Pengujian *Indeks*	Memastikan bahwa *indeks* telah diterapkan dengan benar dan meningkatkan kinerja kueri.	*Indeks* menunjukkan bahwa *indeks* yang diterapkan telah berhasil meningkatkan kinerja kueri dan menjaga integritas data, meskipun ada sedikit peningkatan beban pada operasi penulisan.

V. KESIMPULAN

Berdasarkan hasil perancangan, pengujian dan analisa basis data berbasis dokumen terdistribusi menggunakan MongoDB pada produk Coofis Verse maka dapat diambil beberapa kesimpulan sebagai berikut :

1. MongoDB terbukti efektif dalam mengelola data terdistribusi untuk aplikasi Coofis Verse, memberikan kinerja yang baik dalam menyimpan dan mengakses dokumen-dokumen terkait.
2. MongoDB berhasil digunakan sebagai solusi basis data untuk mendukung kebutuhan dokumen terdistribusi pada Coofis Verse.
3. Perancangan sistem yang menggunakan MongoDB sebagai basis datanya memberikan fleksibilitas dan kecepatan dalam pengelolaan data, mendukung aplikasi Coofis Verse dalam memberikan pengalaman pengguna yang optimal.

REFERENSI

[1] D. Radya Briantama and N. D. Hendrawan, "'Bimasakti' Aplikasi Persuratan Digital Berbasis Web Untuk Manajemen Dokumen Dengan Metode Addie," vol. 6, pp. 72–83, 2023.

[2] K. F. Ribawanto and D. Pramono, "Pengembangan Sistem Nota Dinas Elektronik dengan Tanda Tangan Elektronik Studi Kasus PT Andal Rancang Multi Solusi (Arm Solusi)," vol. 6, no. 8, pp. 3751–3760, 2022.

[3] V. B. Ramu, "Performance Impact of Microservices Architecture," *Rev. Contemp. Sci. Acad. Stud.*, vol. 3, no. 6, 2023, doi: 10.55454/rcsas.3.06.2023.010.

[4] A. Sinambela, E. Ernawati, and F. F. Coastera, "Implementasi Arsitektur Microservices Pada Rancang Bangun Aplikasi Marketplace Berbasis Web (Studi Kasus : Pasar Tradisional Modern Kota Bengkulu)," *Rekursif J. Inform.*, vol. 9, no. 1, pp. 1–13, 2021, doi: 10.33369/rekursif.v9i1.14929.

[5] A. H. Suyanto, "Basis Data Dan Dbms," vol. 1, pp. 1–5, 2015.

[6] Y. Y. Putra, O. Purwaningrum, and R. H. Winata, "Perbandingan Performa Respon Waktu Kueri Mysql, Postgresql, Dan Mongodb," *J. Sist. Inf. dan Bisnis Cerdas*, vol. 15, no. 1, pp. 39–48, 2022, doi: 10.33005/sibc.v15i1.7.

[7] A. M. Wibyantoro and A. T. Asmoro, "Perbandingan Basis Data SQL (relational) dengan NoSQL (no-relational)," *Academia.Edu*, pp. 1–10.

[8] M. Fowler and P. J. Sadalage, *NoSQL Distilled*, vol. 1. 2015.

[9] A. Nordeen, "MongoDB in 24 Hours." p. 70, 2020.