Implementasi Hybrid Database Menggunakan MongoDB dan Milvus untuk Pengelolaan Metadata pada Sistem Deteksi Kemiripan Karya Citra Digital

1st Jihania Ananda Ramadhani
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
jihaniaananda@student.telkomuniversity.ac.id

2nd Iwan Iwut Tritoasmoro
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
iwaniwut@telkomuniversity.ac.id

3rd Eng Danu Dwi Sanjoyo
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
danudwj@telkomuniversity.ac.id

***Abstrak* — Proses pengelolaan data pada sistem deteksi kemiripan karya citra digital memerlukan mekanisme penyimpanan yang efisien serta komunikasi yang terintegrasi antara frontend dan backend. Penelitian ini bertujuan mengimplementasikan *hybrid database* menggunakan MongoDB sebagai penyimpanan metadata dan Milvus sebagai penyimpanan *embedding vector*, serta mengembangkan Metadata Service dan integrasi frontend–backend berbasis REST API. Backend dikembangkan menggunakan arsitektur *microservices* sehingga setiap layanan memiliki tanggung jawab yang terpisah dan dapat berkomunikasi melalui API Gateway. Metadata Service menyediakan fungsi *Create, Read, Update,* dan *Delete* (CRUD) untuk mengelola metadata karya digital, sedangkan integrasi *frontend–backend* dilakukan menggunakan pertukaran data berformat JSON melalui endpoint REST API. Hasil implementasi menunjukkan bahwa mekanisme penyimpanan metadata dan *embedding vector* dapat berjalan secara terintegrasi melalui penggunaan atribut milvus_id, sementara seluruh endpoint REST API berhasil mendukung komunikasi antara frontend dan backend sesuai dengan fungsinya. Pendekatan yang diusulkan menghasilkan sistem yang modular, terstruktur, dan mudah dikembangkan sehingga dapat mendukung pengelolaan data pada sistem deteksi kemiripan karya citra digital secara lebih efektif.**

***Kata kunci*— *Hybrid Database*, MongoDB, Milvus, REST API, *Metadata Service, Frontend-Backend Integration*.**

I. PENDAHULUAN

Perkembangan *Artificial Intelligence* (AI) telah mendorong pemanfaatan representasi data berbasis *embedding* pada berbagai aplikasi, seperti *content-based image retrieval*, sistem rekomendasi, dan deteksi kemiripan citra. Pada pendekatan tersebut, data tidak lagi direpresentasikan sebagai informasi tekstual atau piksel semata, tetapi diubah menjadi vektor berdimensi tinggi sehingga memungkinkan proses pencarian berdasarkan kesamaan semantik. Seiring meningkatnya volume data digital, pengelolaan dan pencarian *embedding vector* menjadi tantangan baru karena basis data konvensional belum dirancang untuk melakukan *similarity search* secara efisien pada data berdimensi tinggi. Kondisi ini mendorong berkembangnya *Vector Database Management System* (VDBMS) sebagai solusi yang dioptimalkan untuk penyimpanan, pengindeksan, dan pencarian data vektor [1] [2] [3].

Pada sistem deteksi kemiripan karya citra digital, data yang dikelola tidak hanya berupa *embedding vector*, tetapi juga metadata yang berisi informasi seperti judul karya, pencipta, kategori, deskripsi, dan status registrasi. Kedua jenis data tersebut memiliki karakteristik yang berbeda sehingga membutuhkan mekanisme penyimpanan yang berbeda pula. Basis data dokumen seperti MongoDB lebih sesuai digunakan untuk menyimpan metadata karena memiliki struktur yang fleksibel dan mudah dikembangkan, sedangkan data vektor membutuhkan *vector database* agar proses pencarian tetap memiliki latensi rendah meskipun jumlah data terus bertambah. Oleh karena itu, pendekatan *hybrid database*, yaitu mengombinasikan basis data dokumen dengan *vector database*, menjadi salah satu pendekatan yang semakin banyak digunakan pada aplikasi berbasis AI untuk memperoleh keseimbangan antara fleksibilitas pengelolaan data dan performa pencarian [2].

Milvus merupakan salah satu *vector database* yang dirancang khusus untuk mengelola data vektor dalam skala besar. Milvus mendukung berbagai metode pengindeksan *Approximate Nearest Neighbor* (ANN), pemrosesan data secara terdistribusi, serta menyediakan SDK dan *RESTful API* yang memudahkan integrasi dengan layanan backend maupun aplikasi berbasis web. Arsitektur tersebut memungkinkan proses pencarian kemiripan dilakukan secara lebih cepat dibandingkan penggunaan basis data umum untuk data vektor, sehingga Milvus banyak dimanfaatkan pada aplikasi pencarian citra, *retrieval-augmented generation*, maupun sistem rekomendasi [1].

Meskipun penelitian mengenai *vector database* berkembang dengan pesat, sebagian besar masih berfokus pada optimasi penyimpanan dan pencarian *embedding*. Pembahasan mengenai integrasi *vector database* dengan basis data metadata, implementasi layanan *Metadata Service* berbasis REST API, serta mekanisme sinkronisasi antara metadata dan *embedding vector* pada aplikasi deteksi kemiripan citra masih relatif terbatas. Padahal, pada implementasi sistem berbasis web, backend tidak hanya bertugas melakukan pencarian data vektor, tetapi juga menyediakan layanan *Create, Read, Update, Delete*

(CRUD), menjaga konsistensi hubungan antara metadata dan *embedding*, serta menjadi penghubung antara frontend dengan seluruh layanan yang tersedia. Arsitektur backend yang kurang terintegrasi dapat menyebabkan inkonsistensi data, redundansi informasi, serta meningkatkan kompleksitas pengelolaan sistem seiring bertambahnya jumlah data dan layanan [2].

Berdasarkan permasalahan tersebut, penelitian ini mengimplementasikan arsitektur backend pada sistem deteksi kemiripan karya citra digital menggunakan pendekatan *hybrid database* yang memanfaatkan Milvus sebagai penyimpanan *embedding vector* dan MongoDB sebagai penyimpanan metadata. Selain itu, penelitian ini mengembangkan *Metadata Service* berbasis REST API yang menyediakan layanan CRUD serta mengintegrasikan komunikasi antara frontend, backend, dan kedua basis data melalui identitas `milvus_id` sebagai penghubung antara metadata dan data vektor. Implementasi ini diharapkan mampu menghasilkan pengelolaan data yang lebih terstruktur, menjaga konsistensi hubungan antara metadata dan *embedding*, serta mendukung proses pencarian kemiripan citra secara efisien pada aplikasi berbasis web.

II. KAJIAN TEORI

Kajian teori pada penelitian ini membahas konsep dan teknologi yang digunakan sebagai dasar dalam pengembangan sistem. Pembahasan meliputi *Hybrid Database*, MongoDB, Milvus, *Metadata Service*, dan REST API sebagai landasan implementasi pengelolaan metadata serta integrasi frontend dan backend.

A. REST API

Representational State Transfer Application Programming Interface (REST API) merupakan gaya arsitektur layanan web yang memungkinkan komunikasi antara *client* dan *server* melalui protokol HTTP. REST API menggunakan metode seperti *GET, POST, PUT*, dan D*ELETE* untuk mengelola sumber daya (*resource*), serta umumnya memanfaatkan format JSON sebagai media pertukaran data. Karakteristik REST API yang bersifat ringan, *stateless*, dan mudah diintegrasikan menjadikannya sebagai salah satu pendekatan yang banyak digunakan dalam pengembangan aplikasi berbasis web dan *microservices* [4] [5].

B. MongoDB

MongoDB merupakan basis data NoSQL yang menyimpan data dalam bentuk dokumen BSON (*Binary JSON*). Pendekatan ini memberikan fleksibilitas dalam pengelolaan data karena tidak bergantung pada skema tabel yang tetap. MongoDB banyak digunakan pada aplikasi berbasis web untuk menyimpan data yang bersifat dinamis, termasuk metadata, serta mendukung implementasi layanan CRUD secara efisien melalui REST API. Pada penelitian ini, MongoDB digunakan untuk menyimpan metadata karya digital yang meliputi informasi judul, kategori, deskripsi, URL gambar, serta identitas `milvus_id` sebagai penghubung dengan data vektor [6].

C. Milvus

Milvus merupakan *Vector Database Management System* (VDBMS) yang dirancang untuk menyimpan, mengelola, dan melakukan pencarian terhadap data vektor berdimensi tinggi. Milvus mendukung berbagai algoritma *Approximate Nearest Neighbor* (ANN) sehingga mampu melakukan proses *similarity search* secara efisien pada data berskala besar. Selain itu, Milvus menyediakan SDK dan REST API sehingga mudah diintegrasikan dengan aplikasi berbasis Artificial Intelligence. Dalam penelitian ini, Milvus digunakan sebagai penyimpanan *embedding vector* yang digunakan pada proses pencarian kemiripan citra [7] [8].

D. *Hybrid Database*

Hybrid database merupakan pendekatan penyimpanan data yang mengombinasikan lebih dari satu jenis basis data sesuai karakteristik data yang dikelola. Pada aplikasi berbasis Artificial Intelligence, metadata dan *embedding vector* memiliki kebutuhan penyimpanan yang berbeda sehingga kombinasi antara basis data dokumen dan *vector database* menjadi solusi yang efektif. Pendekatan ini memungkinkan pengelolaan metadata yang fleksibel sekaligus mempertahankan performa pencarian vektor yang tinggi. Pada penelitian ini diterapkan MongoDB sebagai penyimpanan metadata dan Milvus sebagai penyimpanan *embedding vector* [8].

E. *Metadata Service*

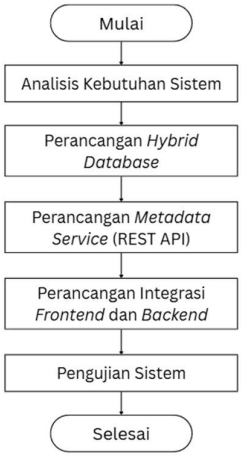
Metadata Service merupakan layanan backend yang menyediakan fungsi *Create, Read, Update,* dan *Delete* (CRUD) terhadap metadata karya digital melalui REST API. Layanan ini berperan sebagai penghubung antara frontend dengan basis data sehingga proses pengelolaan metadata dilakukan secara terpusat. Dengan pendekatan tersebut, integritas data dapat dipertahankan serta sinkronisasi antara metadata di MongoDB dan *embedding vector* di Milvus dapat dikelola secara konsisten.

III. METODE

Bab ini menjelaskan metode yang digunakan dalam penelitian sebagai acuan dalam pengembangan sistem. Metode penelitian meliputi tahapan penelitian, perancangan *hybrid database*, perancangan *Metadata Service*, perancangan integrasi antara frontend dan backend, serta metode pengujian untuk mengevaluasi fungsionalitas sistem. Setiap tahapan dirancang untuk memastikan sistem mampu mengelola metadata karya digital, menghubungkan data metadata dengan *embedding vector*, serta mendukung komunikasi antara frontend dan backend melalui REST API.

A. Tahapan Penelitian

Penelitian ini menggunakan metode implementasi sistem dengan pendekatan pengembangan backend berbasis *microservices*. Tahapan penelitian dimulai dengan analisis kebutuhan sistem berdasarkan proses bisnis pengelolaan data karya digital. Selanjutnya dilakukan perancangan arsitektur backend, implementasi *Metadata Service*, penerapan *hybrid database*, integrasi frontend dan backend menggunakan REST API, serta pengujian terhadap seluruh layanan yang dikembangkan. Tahapan penelitian ditunjukkan pada Gambar 1.



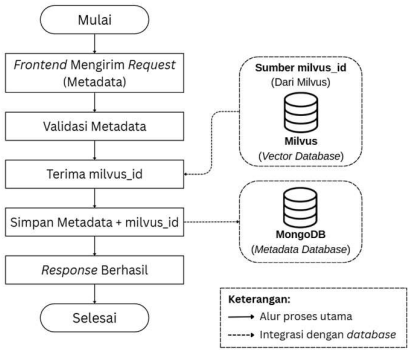
GAMBAR 1
Tahapan Penelitian

B. Perancangan *Hybrid Database*

Penelitian ini menerapkan pendekatan *hybrid database* dengan memanfaatkan MongoDB sebagai penyimpanan metadata dan Milvus sebagai penyimpanan *embedding vector*. Hubungan antara kedua basis data dilakukan menggunakan atribut `milvus_id` yang berfungsi sebagai referensi terhadap data vektor pada Milvus. Rancangan ini bertujuan memisahkan penyimpanan berdasarkan karakteristik data sehingga pengelolaan metadata dan pencarian kemiripan dapat dilakukan secara efisien.

C. Perancangan Metadata Service

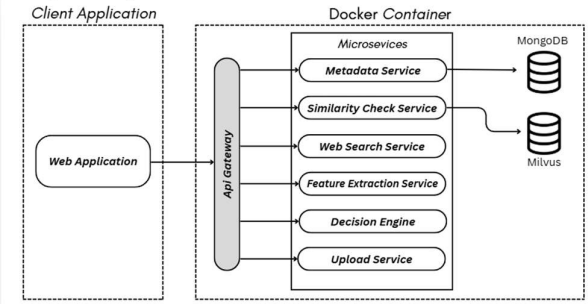
Metadata Service dikembangkan sebagai layanan backend yang menyediakan fungsi *Create, Read, Update*, dan *Delete* (CRUD) terhadap metadata karya digital melalui REST API. Setiap permintaan dari frontend akan divalidasi sebelum diproses oleh backend. Data metadata kemudian disimpan ke MongoDB, sedangkan `milvus_id` disimpan sebagai referensi terhadap *embedding vector* pada Milvus. Alur penyimpanan metadata ditunjukkan pada Gambar 2.



GAMBAR 2
Perancangan Metadata Service

D. Perancangan Integrasi Frontend dan Backend

Integrasi antara frontend dan backend dirancang menggunakan arsitektur *microservices* dengan komunikasi berbasis REST API. Seluruh permintaan (*request*) dari aplikasi frontend dikirim melalui protokol HTTP menuju *API Gateway* sebagai *entry point* sistem. *API Gateway* kemudian meneruskan permintaan ke layanan backend sesuai dengan endpoint yang diakses. Pendekatan ini memungkinkan setiap layanan backend bekerja secara independen serta mempermudah proses pengembangan dan pemeliharaan sistem. Adapun perancangan Integrasi *Frontend* dan *Backend* ditunjukkan pada Gambar 3.



GAMBAR 3
Perancangan Integrasi *Frontend* dan *Backend*

Berdasarkan Gambar 3, backend terdiri atas beberapa layanan, yaitu *Upload Service, Feature Extraction Service, Similarity Check Service, Web Search Service, Decision Engine,* dan *Metadata Service*. Setiap layanan memiliki fungsi yang berbeda dalam mendukung proses pemeriksaan kemiripan karya digital. Upload Service menerima citra dari *frontend, Feature Extraction Service* menghasilkan *embedding vector*, Similarity Check Service melakukan pencarian kemiripan terhadap data yang tersimpan di Milvus, *Web Search Service* melakukan pencarian referensi eksternal apabila diperlukan, *Decision Engine* menghasilkan keputusan berdasarkan hasil pemeriksaan, sedangkan Metadata Service bertanggung jawab mengelola metadata karya digital pada MongoDB.

Pada proses integrasi, *frontend* hanya berkomunikasi dengan API Gateway sehingga tidak mengakses layanan *backend* maupun basis data secara langsung. Setiap layanan *backend* saling berkomunikasi melalui REST API sesuai kebutuhan proses bisnis, kemudian hasil pengolahan dikembalikan kepada *frontend* dalam format JSON untuk ditampilkan kepada pengguna. Perancangan ini menghasilkan komunikasi yang terstruktur, modular, dan memudahkan pengembangan setiap layanan tanpa mengubah antarmuka *frontend.*

E. Pengujian Sistem

Pengujian dilakukan untuk memastikan seluruh layanan backend berjalan sesuai dengan rancangan. Pengujian meliputi fungsi endpoint REST API, operasi CRUD pada *Metadata Service*, penyimpanan metadata pada MongoDB, penyimpanan *embedding vector* pada Milvus, serta integrasi komunikasi antara *frontend* dan *backend.* Hasil pengujian digunakan untuk mengevaluasi keberhasilan implementasi sistem dalam mengelola metadata dan menjaga konsistensi hubungan antara MongoDB dan Milvus.

IV. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan implementasi backend berbasis *microservices* yang terintegrasi dengan MongoDB sebagai penyimpanan metadata dan Milvus sebagai penyimpanan *embedding vector*. Selain itu, dikembangkan

Metadata Service berbasis REST API untuk mengelola operasi *Create, Read, Update*, dan *Delete* (CRUD) terhadap metadata karya digital. Implementasi tersebut kemudian dievaluasi melalui pengujian fungsional terhadap layanan backend, penyimpanan basis data, dan komunikasi antara frontend dengan backend.

A. Implementasi Hybrid Database

Implementasi *hybrid database* pada sistem dilakukan dengan memanfaatkan dua jenis basis data yang memiliki fungsi berbeda, yaitu MongoDB sebagai penyimpanan metadata karya digital dan Milvus sebagai penyimpanan *embedding vector*. Pendekatan ini diterapkan untuk memisahkan penyimpanan data berdasarkan karakteristiknya sehingga pengelolaan metadata dan proses pencarian kemiripan citra dapat dilakukan secara lebih efisien.



GAMBAR 4
Implementasi Penyimpanan Metadata pada MongoDB

Gambar 4 menunjukkan implementasi penyimpanan metadata pada MongoDB. Metadata yang tersimpan telah memiliki atribut `milvus_id` yang nilainya identik dengan id pada Milvus. Hal ini menunjukkan bahwa hubungan antara metadata dan *embedding vector* berhasil dibentuk menggunakan satu identitas yang sama tanpa terjadi redundansi data.



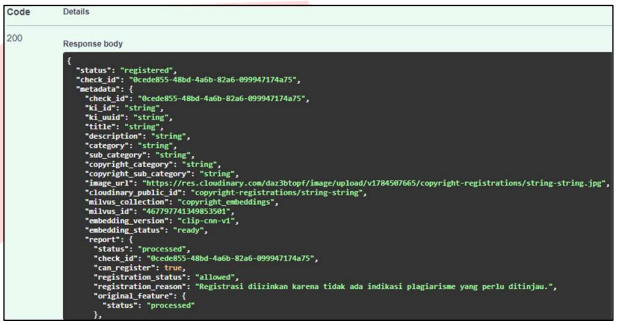
GAMBAR 5
Implementasi Penyimpanan *Embedding Vector* pada Milvus

Gambar 5 menunjukkan implementasi penyimpanan *embedding vector* pada Milvus. Setiap data *embedding vector* disimpan menggunakan id unik yang digunakan sebagai identitas data vektor. Nilai id tersebut kemudian direferensikan oleh atribut `milvus_id` pada MongoDB, sehingga sistem dapat menghubungkan metadata karya digital dengan data *embedding vector* yang sesuai. Pada contoh implementasi, nilai id pada Milvus sama dengan nilai `milvus_id` pada MongoDB, yang menunjukkan bahwa hubungan antara kedua basis data berhasil dibentuk melalui satu identitas yang sama.

Implementasi *hybrid database* yang dikembangkan berhasil memisahkan penyimpanan metadata dan *embedding vector* sesuai dengan karakteristik masing-masing basis data. MongoDB berperan dalam mendukung operasi CRUD terhadap metadata, sedangkan Milvus dioptimalkan untuk penyimpanan dan pencarian data vektor. Penggunaan `milvus_id` sebagai penghubung antara kedua basis data menjaga konsistensi hubungan data, mengurangi redundansi penyimpanan, serta mempermudah proses integrasi dengan layanan backend.

B. Implementasi Metadata Service

Metadata Service berhasil diimplementasikan sebagai layanan backend yang bertanggung jawab dalam pengelolaan metadata karya digital melalui mekanisme REST API. Layanan ini menerima data metadata dari frontend, melakukan validasi terhadap data yang dikirimkan, kemudian menyimpan metadata ke MongoDB beserta atribut `milvus_id` yang diperoleh dari Milvus. Setelah proses penyimpanan selesai, backend mengembalikan respons dalam format JSON sebagai konfirmasi bahwa proses registrasi metadata berhasil dilakukan.



GAMBAR 6
Implementasi *Endpoint* `POST /api/v1/register-metadata`

Berdasarkan Gambar 6, endpoint `POST /api/v1/register-metadata` berhasil menerima metadata karya digital yang dikirimkan dari frontend dan mengembalikan respons dengan status `registered`. Respons tersebut memuat informasi metadata, seperti `metadata_id`, `check_id`, dan `milvus_id`, yang menunjukkan bahwa metadata telah berhasil disimpan dan dihubungkan dengan data *embedding vector* pada Milvus. Selain itu, respons juga menampilkan `registration_status` dan `embedding_status` sebagai informasi bahwa proses registrasi metadata telah berhasil diproses oleh sistem.

Hasil implementasi ini menunjukkan bahwa *Metadata Service* mampu mengelola proses registrasi metadata secara terstruktur, menjaga keterhubungan antara MongoDB dan Milvus melalui atribut `milvus_id`, serta menyediakan respons dalam format JSON yang dapat digunakan oleh frontend untuk proses selanjutnya.

C. Implementasi Integrasi *Frontend* dan *Backend*

Integrasi antara *frontend* dan *backend* berhasil diimplementasikan menggunakan REST API sebagai mekanisme komunikasi antar komponen sistem. Frontend mengirimkan *request* menggunakan protokol HTTP menuju *endpoint* yang tersedia pada *backend*, kemudian *backend* memproses permintaan tersebut dan mengembalikan *response* dalam format JSON**.** Pendekatan ini memungkinkan *frontend* dan *backend* berkomunikasi tanpa melakukan akses langsung ke basis data sehingga meningkatkan modularitas dan mempermudah proses pengembangan sistem.

Pada implementasinya, integrasi mencakup proses registrasi metadata, pengambilan data metadata, pembaruan metadata, penghapusan metadata, serta proses persetujuan

hasil pemeriksaan karya. Seluruh fungsi tersebut diakses melalui *endpoint* REST API yang ditunjukkan pada Tabel 1.

TABEL 1
Endpoint REST API pada Integrasi *Frontend* dan *Backend*

Method	*Endpoint*	Fungsi
Get	`/health`	Memeriksa status layanan backend.
POST	`/api/v1/upload`	Mengirim citra digital dari frontend ke backend untuk diproses.
POST	`/api/v1/register-metadata`	Menyimpan metadata karya digital ke MongoDB beserta referensi milvus_id.
POST	`/api/v1/review-check/{check_id}/approve`	Memperbarui stastus hasil pemeriksaan karya menjadi *approved.*
GET	`/api/v1/metadata`	Menampilkan seluruh metadata karya digital.
GET	`/api/v1/metadata/{metadata_id}`	Menampilkan metadata berdasarkan ID.
PUT	`/api/v1/metadata/{metadata_id}`	Memperbarui metadata karya digital.
DELETE	`/api/v1/metadata/{metadata_id}`	Menghapus metadata karya digital.

D. Hasil Pengujian Sistem

Pengujian dilakukan untuk memastikan setiap fungsi backend berjalan sesuai dengan kebutuhan sistem. Pengujian meliputi penyimpanan metadata, integrasi MongoDB dan Milvus, operasi CRUD, serta komunikasi antara frontend dan backend.

TABEL 2
Hasil Pengujian Sistem

Pengujian	Kriteria Keberhasilan	Hasil
Penyimpanan Metadata	Metadata berhasil disimpan pada MongoDB	Berhasil
Integrasi MongoDB-Milvus	milvus_id pada MongoDB sesuai dengan data di Milvus	Berhasil
Operasi CRUD	*Create, Read, Update,* dan *Delete* berjalan sesuai fungsi	Berhasil
Integrasi Frontend-Banckend	Frontend menerima respons JSON dari backend	Berhasil

Berdasarkan hasil pengujian, seluruh layanan backend berhasil berjalan sesuai dengan rancangan sistem. Metadata tersimpan dengan baik pada MongoDB, sedangkan hubungan dengan data *embedding vector* di Milvus tetap terjaga melalui penggunaan `milvus_id`. Selain itu, seluruh operasi CRUD dapat dilakukan tanpa kendala dan komunikasi antara frontend dengan backend berlangsung secara konsisten melalui REST API.

E. Pengembangan Penelitian Selanjutnya

Berdasarkan hasil implementasi yang telah dilakukan, sistem berhasil menerapkan *hybrid database* menggunakan MongoDB dan Milvus serta mengintegrasikan frontend dan backend melalui REST API. Meskipun demikian, masih terdapat beberapa aspek yang dapat dikembangkan pada penelitian selanjutnya. Dari sisi keamanan, sistem dapat ditingkatkan dengan menerapkan mekanisme autentikasi dan otorisasi pada setiap endpoint REST API untuk membatasi akses terhadap layanan backend. Selain itu, penerapan *caching* menggunakan Redis dapat dipertimbangkan untuk meningkatkan performa akses metadata yang sering digunakan sehingga waktu respons sistem menjadi lebih cepat. Pengembangan juga dapat dilakukan dengan mengevaluasi penggunaan *vector database* lain sebagai pembanding Milvus untuk menganalisis performa penyimpanan, pencarian, dan skalabilitas. Di samping itu, mekanisme sinkronisasi transaksi antara MongoDB dan Milvus dapat dikembangkan agar konsistensi data tetap terjaga apabila terjadi kegagalan pada salah satu proses penyimpanan.

V. KESIMPULAN

Penelitian ini berhasil mengimplementasikan backend pada sistem deteksi kemiripan karya citra digital dengan menerapkan pendekatan *hybrid database* yang memanfaatkan MongoDB sebagai penyimpanan metadata dan Milvus sebagai penyimpanan *embedding vector*. Implementasi Metadata Service berhasil menyediakan layanan CRUD terhadap metadata karya digital melalui REST API, sehingga proses pengelolaan data dapat dilakukan secara terstruktur dan konsisten. Selain itu, integrasi antara frontend dan backend berhasil diimplementasikan melalui mekanisme REST API yang memungkinkan pertukaran data menggunakan format JSON secara efektif. Hasil pengujian menunjukkan bahwa seluruh endpoint yang dikembangkan dapat berfungsi sesuai dengan kebutuhan sistem, proses penyimpanan metadata dan referensi *embedding* berjalan dengan baik, serta komunikasi antara frontend dan backend berlangsung secara stabil. Dengan demikian, implementasi yang dikembangkan mampu mendukung pengelolaan data karya digital secara terintegrasi, modular, dan mudah dikembangkan untuk mendukung kebutuhan sistem deteksi kemiripan citra digital.

REFERENSI

[1] J. Wang *dkk.*, “Milvus: A Purpose-Built Vector Data Management System,” dalam *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Association for Computing Machinery, 2021, hlm. 2614–2627. doi: 10.1145/3448016.3457550.

[2] J. J. Pan, J. Wang, dan G. Li, “Survey of Vector Database Management Systems,” Okt 2023, [Daring]. Tersedia pada: http://arxiv.org/abs/2310.14021

[3] T. Taipalus, “Vector database management systems: Fundamental concepts, use-cases, and current challenges,” *Cogn. Syst. Res.*, vol. 85, Jun 2024, doi: 10.1016/j.cogsys.2024.101216.

[4] S. U. Meshram, "Evolution of Modern Web Services – REST API with its Architecture and Design," *International Journal of Research in Engineering, Science and Management*, vol. 4, hlm. 83–86, Jul 2021.

[5] F. Tanveer, F. Iradat, W. Iqbal, dan A. Ahmad, "Towards Secure APIs: A Survey on RESTful API Vulnerability Detection," 2025, *Tech Science Press*. doi: 10.32604/cmc.2025.067536.

[6] E. Shkodra, E. Jajaga, dan M. Shala, "Development and Performance Analysis of RESTful APIs in Core and Node.js using MongoDB Database", doi: 10.5220/0010621200003058.

[7] "Milvus Architecture Overview | Milvus Documentation." Diakses: 19 Juli 2026. [Daring]. Tersedia pada: https://milvus.io/docs/architecture_overview.md

[8] L. Ma *dkk.*, "A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge," Mar 2023, [Daring]. Tersedia pada: http://arxiv.org/abs/2310.11703