JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 1, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i1.8790

PERANCANGAN SERVER APLIKASI SEDERHANA GUNA MENINGKATKAN EFEKTIVITAS PENGELOLAAN DATA

Mohammad Agil[1*], Imron Sodikin[2], Ulil Absor[3] , A. Hamdani [4]

[1,]Program Studi Sistem Informasi Fakultas Sains dan Teknologi, Universitas Ibrahimy,Situbondo
[2]Program Studi Sistem Informasi Fakultas Sains dan Teknologi, Universitas Ibrahimy,Situbondo

Keywords:
Server aplikasi, pengelolaan data, perancangan sistem, arsitektur tiga tingkat, CRUD, RESTful API

Corespondent Email:
mohagil040@gmail.com

Abstrak. Penelitian ini menyajikan perancangan dan implementasi konsep dari sebuah server aplikasi sederhana yang bertujuan untuk mendukung proses dasar pengelolaan data, khususnya di lingkungan pendidikan atau instansi skala kecil. Kebutuhan akan sistem informasi yang terstruktur menuntut pemahaman yang jelas mengenai peran server aplikasi sebagai penghubung antara permintaan klien dengan penyimpanan data. Banyak penelitian terdahulu cenderung berfokus pada implementasi dan konfigurasi tingkat lanjut yang kompleks, sehingga menyulitkan pemula untuk memahami konsep dasar. Oleh karena itu, penelitian ini bertujuan untuk mengisi keselarasan tersebut dengan menyediakan rancangan server aplikasi yang konseptual, ringan, dan mudah diterapkan. Metodologi penelitian mengadopsi pendekatan rekayasa sistem, yang meliputi tahapan analisis kebutuhan fungsional dan non-fungsional, perancangan arsitektur sistem three-tier (klien, server aplikasi, server basis data), pemodelan alur data menggunakan Entity-Relationship Diagram (ERD), dan perancangan Application Programming Interface (API) berbasis RESTful untuk operasi CRUD (Create, Read, Update, Delete). Hasil perancangan menunjukkan bahwa arsitektur yang diusulkan mampu mendukung aktivitas pengelolaan data secara efektif tanpa memerlukan implementasi yang kompleks. Sebagai studi kasus, dirancang sebuah sistem pengelolaan data siswa sederhana dengan antarmuka pengguna dasar. Model konsepsi dan implementasi parsial ini diharapkan dapat menjadi referensi awal yang komprehensif bagi pelajar, pengembang pemula, maupun instansi kecil yang ingin memahami dan membangun fondasi sistem informasi berbasis server yang efektif dan efisien.

Abstract. *his study presents the design and implementation of the concept of a simple application server intended to support basic data management processes, particularly in educational environments or small-scale institutions. The growing need for structured information systems requires a clear understanding of the role of an application server as an intermediary between client requests and data storage. Many previous studies tend to focus on complex, advanced-level implementations and configurations, which often make it difficult for beginners to grasp the fundamental concepts. Therefore, this study aims to address this gap by providing a conceptual, lightweight, and easy-to-implement application server design.The research methodology adopts a systems engineering approach, which includes the stages of functional and non-functional requirements analysis, three-tier system architecture design (client, application server, and database server), data flow modeling using an Entity-Relationship Diagram (ERD), and the design of a RESTful Application Programming Interface (API) for CRUD (Create, Read, Update, Delete) operations.The design results indicate that the proposed*

architecture is capable of effectively supporting data management activities without requiring complex implementation. As a case study, a simple student data management system with a basic user interface is designed. This conceptual model and partial implementation are expected to serve as a comprehensive introductory reference for students, beginner developers, and small institutions seeking to understand and build an effective and efficient server-based information system foundation.

1. PENDAHULUAN

Di era digitalisasi, kemampuan untuk mengelola data secara sistematis, cepat, dan akurat telah menjadi fondasi utama bagi kemajuan suatu organisasi, baik itu lembaga pemerintahan, swasta, maupun institusi pendidikan. Proses manajemen data yang manual, misalnya menggunakan lembar kerja spreadsheet atau bahkan arsip kertas, rentan terhadap berbagai masalah seperti kesalahan input data, redundansi informasi, keamanan yang terbatas, dan proses akses yang tidak efisien. Untuk mengatasi tantangan ini, diperlukan sebuah sistem informasi yang terstruktur dan andal

Pada inti dari setiap sistem informasi berbasis web modern terdapat server aplikasi (server aplikasi). Server aplikasi berfungsi sebagai middleware atau perantara krusial yang bertanggung jawab atas pengiriman logika bisnis. Ia menerima permintaan dari klien (browser pengguna), memproses permintaan tersebut, berinteraksi dengan basis data untuk mengambil atau menyimpan informasi, dan kemudian mengirimkan kembali respon yang sesuai kepada pengguna [1]. Tanpa adanya server aplikasi yang dirancang dengan baik, komunikasi antara antarmuka pengguna dan lapisan penyimpanan data akan menjadi kacau dan tidak efisien.

Berbagai penelitian sebelumnya telah banyak membahas pengembangan sistem informasi dan arsitektur server. Penelitian di bidang sistem integrasi, misalnya, telah menunjukkan pentingnya peran server sebagai perantara utama dalam pertukaran data menggunakan layanan seperti REST API [1]. Di sisi lain, kajian lain menekankan pada optimalisasi manajemen data dasar melalui server model yang dimaksudkan untuk meningkatkan efisiensi akses dan keamanan data [2]. Namun, sebagian besar penelitian ini cenderung berfokus pada implementasi tingkat lanjut, penggunaan kerangka kerja tertentu yang kompleks, atau konfigurasi server untuk lingkungan produksi berskala besar. Fokus ini sering kali mengabaikan kebutuhan para pemula—mahasiswa, pengembang perangkat lunak yang baru memulai, atau staf IT di instansi kecil—yang memerlukan pemahaman mendasar mengenai cara kerja server aplikasi sebelum melangkah ke konfigurasi yang lebih rumit.

State of the art dalam bidang ini menunjukkan bahwa terdapat uraian mengenai desain server aplikasi sederhana secara konseptual yang disertai dengan contoh implementasi yang mudah dipahami. Padahal, arsitektur server yang sederhana dan ringan sangat relevan untuk kebu*virtualisasi ringan* .

2. TINJAUAN PUSTAKA

Tinjauan pustaka pada penelitian ini membahas konsep-konsep fundamental yang menjadi landasan teoritis dalam perancangan dan implementasi server aplikasi sederhana.

1. 2.1 Server Aplikasi dan Arsitektur Tiga Tingkat

Server aplikasi adalah perangkat lunak *middleware* yang menyediakan lingkungan untuk menjalankan aplikasi, khususnya aplikasi berbasis web. Tugas utamanya adalah mengelola logika bisnis dari aplikasi. Ia bertindak sebagai perantara antara klien (biasanya browser web) dan sumber daya lain seperti basis data atau layanan eksternal [5]. Server aplikasi menangani berbagai tugas kompleks seperti manajemen sesi pengguna, validasi data, eksekusi transaksi, dan pengelolaan koneksi ke data dasar.

Untuk memastikan server aplikasi berjalan secara optimal, diperlukan arsitektur yang terstruktur. **Arsitektur three-tier** (tiga lapisan) adalah model arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga lapisan logistik dan fisik yang berbeda [6]. Model ini dirancang untuk meningkatkan skalabilitas,

ketidaknyamanan, dan pemeliharaan. Tiga lapisan tersebut adalah:

2. **Tingkat Presentasi (Lapisan Presentasi):** antarmuka pengguna (UI) yang berinteraksi langsung dengan pengguna. Dalam aplikasi web, lapisan ini dijalankan di browser klien.
3. **Tingkat Aplikasi (Lapisan Aplikasi/Logika):** Inti dari aplikasi, di mana seluruh logika bisnis berada. Lapisan ini dijalankan pada server aplikasi.
4. **Data Tier (Lapisan Data):** Bertanggung jawab untuk menyimpan, mengelola, dan mengakses data. Biasanya terdiri dari sistem manajemen basis data (DBMS).

Pemisahan ini memungkinkan pengembang untuk memperbarui satu lapisan tanpa mengubah lapisan lain secara signifikan.

5. 2.2 Operasi CRUD dan RESTful API

CRUD adalah singkatan dari empat operasi dasar yang paling umum digunakan dalam pengelolaan data persisten: Create , Read , Update , dan Delete [7]. Setiap sistem pengelolaan data yang kompleks pada dasarnya dibangun di atas kombinasi dari keempat operasi ini.

Untuk menghubungkan lapisan aplikasi dengan lapisan data, pendekatan modern yang paling populer adalah menggunakan RESTful API (Representational State Transfer). REST adalah gaya arsitektur untuk mendesain aplikasi terdistribusi. API yang mematuhinya disebut RESTful API. API ini menggunakan metode HTTP standar (GET untuk Read, POST untuk Create, PUT/PATCH untuk Update, dan DELETE untuk Delete) untuk melakukan operasi CRUD [1]. Penggunaan RESTful API memungkinkan komunikasi yang jelas, *stateless* (tidak menyimpan status sesi di sisi server), dan skalabel antara klien dan server.

6. 2.3 Node.js dan Express.js

Node.js adalah lingkungan runtime JavaScript sisi server yang bersifat *open-source* dan *cross-platform* . Node.js menjalankan mesin JavaScript V8 (mesin yang sama dengan Google Chrome) di luar browser, yang memungkinkan pengembang untuk menggunakan JavaScript untuk menulis *alat baris perintah* dan *skrip sisi server* [3]. Keunggulan utama Node.js adalah model I/O non-blocking dan berbasis *event-driven* , yang membuatnya sangat ringan dan efisien untuk aplikasi yang bersifat real-time dan data-intensif.

Express.js adalah kerangka kerja (*framework) aplikasi web* minimalis dan fleksibel untuk Node.js. Express menyediakan sekumpulan fitur yang kuat untuk aplikasi web dan mobile, mempermudah proses pembuatan API, routing, middleware, dan penanganan permintaan HTTP. Karena sifatnya yang minimalis, Express tidak memaksakan struktur tertentu, memberikan kebebasan bagi pengembang untuk merancang arsitektur aplikasi mereka sendiri, yang menjadikannya pilihan ideal untuk proyek-proyek sederhana dan bagi pemula [3].

3. METODE PENELITIAN

3.1 Rancangan Penelitian

Penelitian Penelitian ini menggunakan metode deskriptif kualitatif dengan pendekatan rekayasa sistem. Pendekatan ini dipilih karena tujuan utamanya adalah merancang, membangun, dan mendemonstrasikan sebuah prototipe sistem. Proses penelitian ini dibagi beberapa menjadi tahapan sistematis. [1]. [2].

3.1 Tahapan Perancangan dan Implementasi Sistem

Proses perancangan dan implementasi mengikuti model iteratif sederhana, dengan fokus pada penghasilan artefak desain yang jelas pada setiap tahap.

1. **Analisis Kebutuhan:**

Fungsional: Sistem harus memungkinkan admin untuk melakukan operasi CRUD penuh pada data master (data mahasiswa). Sistem harus memiliki mekanisme login sederhana untuk membedakan antara admin dan pengguna umum (read-only). [3].

Non-Fungsional: Sistem harus responsif, memiliki antarmuka yang intuitif, dan dapat berjalan pada server sumber daya yang minimal. [4].

2. **Perancangan Arsitektur Sistem:** Mengadopsi arsitektur three-tier dengan teknologi: Node.js/Express.js (Application Layer), MySQL (Data Layer), dan HTML/CSS/JS (Presentation Layer). [5].
3. Perancangan Basis Data: Membuat Entity-Relationship Diagram (ERD) untuk memodelkan entitas (, ,

`kategoriusersdata_mastercategories,) dan hubungannya.activity_logs. [6].

4. Peranan API: Dalam arsitektur three-tier yang diimplementasikan, Application Programming Interface (API) bukan sekadar komponen tambahan, melainkan **jantung dari lapisan**jantung dari lapisan aplikasi (Application Layer)** dan kunci dari efektivitas seluruh sistem. [7].
5. Implementasi Kode: Tahap implementasi kode adalah fase di mana rencana sistem diwujudkan dalam bentuk perangkat lunak yang dapat dilaksanakan. Proses ini dilakukan secara sistematis dengan mengikuti arsitektur three-tier yang telah ditetapkan, memastikan batasan tanggung jawab yang jelas antara lapisan data, logika aplikasi, dan presentasi. Kode ditulis dengan tekanan pada keterbacaan, modularitas, dan praktik keamanan dasar. [8].
6. Pengujian dan Integrasi: Setelah tahap implementasi kode selesai, langkah krusial berikutnya adalah melakukan pengujian dan integrasi. Tujuan dari tahap ini adalah untuk memastikan bahwa setiap komponen perangkat lunak, baik secara individu maupun sebagai sebuah sistem yang terintegrasi, berfungsi sesuai dengan yang diharapkan. Pengujian dilakukan secara sistematis untuk mengidentifikasi, melacak, dan memperbaiki *bug* atau ketidaksesuaian sebelum sistem siap digunakan. [9].

3.2 Penyebaran Lingkungan

Untuk

- Sistem Operasi: Ubuntu 22
- Server Web: T*proksi terbalik* .
- Lingkungan eksekusi: Node.
- Kerangka kerja: Express.js v4
- Data Dasar: MySQL

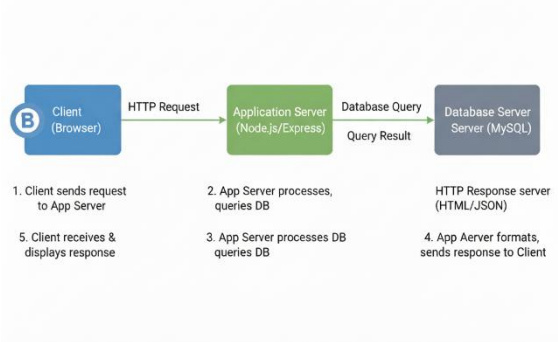
Penyebaran Proses:

1. Cloning repository Repositori kloning
2. Instalasi depen *npm install*).
3. File konfigurasi.envuntuk koneksi basis data dan
4. Menjalanka*alat migrasi*) atau m
5. Pertahanan aplikasi menggunakan atau lebih baik lagi, menggunakan pro *node app.jspm2 start app.js*) untuk menjaga. [10].

4. HASIL DAN PEMBAHASAN

4.1 Hasil Perancangan Arsitektur Sistem

Arsitektur three-tier yang dirancang diimplementasikan dengan teknologi yang telah dipilih.

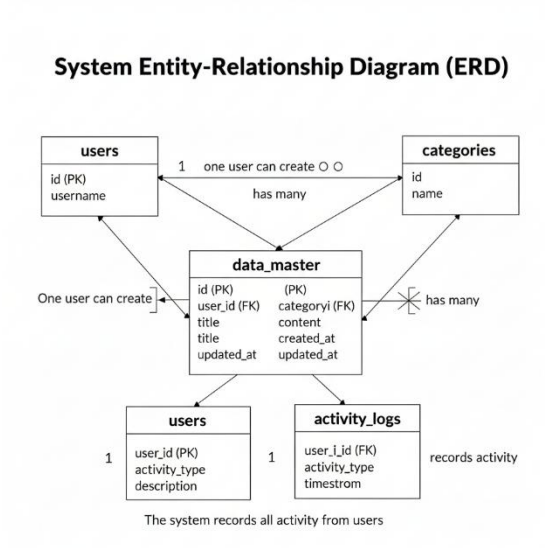


Gambar 1: Diagram Arsitektur Three-Tier yang Diimplementasikan

1. Klien (Browser) mengirimkan permintaanServer Aplikasi (Node.js/Express) .
2. Server Aplikasi menerima permintaan, memprosServer Basis Data (MySQL) .
3. Server Basis Data• query dan mengServer Aplikasi.
4. Server Aplikasi memformat data (biasanyaKlien .
5. Klienmenerima respons dan

Use Case Diagram

Use Berdasarkan analisis menunjukkan Entitas.



Gambar 2: Diagram Entitas-Relasi (ERD) Sistem

4.3 Hasil Implementasi API

API *berhasdata_master.*

Berkas: *routes/dataRoutes.js*

```
const express = require('express');
const router = express.Router();
const dataController = require('../controllers/dataController');
const authMiddleware = require('../middleware/auth'); // Middleware untuk autentikasi

// Semua route di bawah ini memerlukan autentikasi
router.use(authMiddleware);

// CREATE: POST /api/data
router.post('/', dataController.createData);

// READ: GET /api/data
router.get('/', dataController.getAllData);

// READ: GET /api/data/:id
router.get('/:id', dataController.getDataById);

// UPDATE: PUT /api/data/:id
router.put('/:id', dataController.updateData);

// DELETE: DELETE /api/data/:id
router.delete('/:id', dataController.deleteData);

module.exports = router;
```

Gambar 2: Berkas:routes/dataRoutes.js

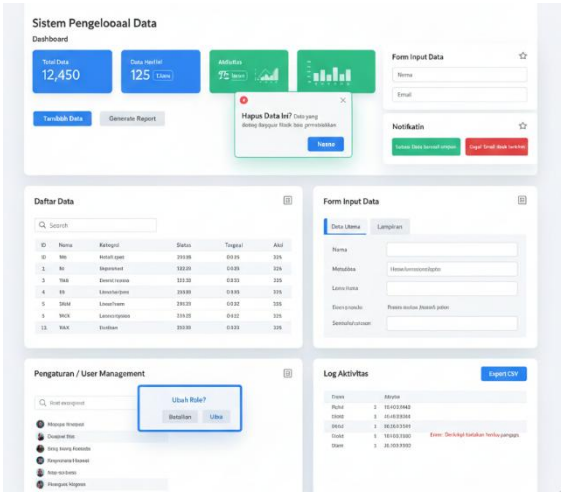
```
const db = require('../dbConnection'); // Modul koneksi database

exports.createData = (req, res) => {
    const { name, description, category_id } = req.body;
    const created_by = req.user.id; // ID user didapat dari middleware auth

    const sql = 'INSERT INTO data_master (name, description, category_id, created_by) VALUES (?, ?, ?, ?)';
    db.query(sql, [name, description, category_id, created_by], (err, result) => {
        if (err) {
            return res.status(500).json({ error: err.message });
        }
        res.status(201).json({ message: 'Data created successfully', dataId: result.insertId });
    });
};
```

Gambar 3: (Contoh fungsi)controllers/dataController.jscreateData

4.4 Hasil Prototipe Antarmuka Pengguna

Prototipe antarmuka pengguna (User Interface - UI) dikembangkan dengan filosofi desain yang berfokus pada pengguna (*user-centric*), tekanan pada kemudahan penggunaan (*usability*), kejernihan informasi, dan responsivitas. Tujuannya adalah untuk menciptakan antarmuka yang bercermin sehingga pengguna, bahkan yang memiliki kemampuan teknis terbatas, dapat dengan mudah memahami dan menjalankan fungsi-fungsi sistem. Antarmuka ini dibangun menggunakan teknologi dasar web: HTML5 untuk struktur, CSS3 dengan pendekatan *mobile-first* dan *flexbox/grid* untuk tata letak yang fleksibel, serta JavaScript vanilla untuk interaktivitas dinamis. Semua interaksi klien dengan server aplikasi dilakukan secara asinkron menggunakan API, yang memungkinkan pembaruan data tanpa perlu melakukan *reload* halaman penuh, sehingga memberikan pengalaman pengguna yang lebih cepat dan lebih lancar.*fetch.* [11].

Gambar 4 menunjukkan tampilan utama dari dashboard admin, yang merupakan kendali pusat untuk pengelolaan data.



Gambar 4: Dashboard Admin dengan Tabel Data Master

Secara rinci, antarmuka pengguna ini terdiri dari beberapa komponen dan alur kerja sebagai berikut:

1. Header dan Navigasi Bagian atas (header) dari antarmuka menampilkan judul aplikasi, informasi pengguna yang sedang login (misalnya, "Selamat datang, Admin"), dan tombol untuk keluar (*logout*). Desain header yang bersih memastikan identitas aplikasi dan status pengguna selalu terlihat jelas.

2. Tabel Data Master Komponen utama dari dashboard adalah tabel data yang menampilkan semua record dari tabel di basis data. Setiap baris mewakili satu entri data, dengan kolom-kolom yang disesuaikan untuk menampilkan informasi penting seperti:**data_master**

- ID: Pengenal unik untuk setiap data.

- Nama: Nama utama dari entitas data (misalnya, nama siswa).
- Kategori: Kategori tempat data tersebut dikumpulkan, diambil dari tabel .categories
- Tanggal Dibuat: Menunjukkan kapan data tersebut pertama kali dimasukkan ke sistem.
- Aksi: Kolom ini berisi tomEdit dan **Hapus Hapus .

**3. Alu3. Alur Kerja Membuat dan Mengedit Data (Create & Update) Untuk pria*dialog modal* .

- Membuat Data: Pengguna priadata_master, seNama(masukan t)Deskripsi(textarea), dan `KategKategori(dropdown yang d
- Mengedit Data: Ketika pengguna menekan tombol "EGET /api/data/:id. MembentukPOST /api/datatidakPUT /api/data/:iduntuk mengupdate). Jika ser

**44. Alur Kerja Menghapus Data (Delete) Aksi hapuconfirm()) akan muncul, menaDELETEakan dDELETE /api/data/:idJika berhasil, bar

5. Fitur Pencarian dan Filter Untuk

- Pencarian (Search): Sebuah masukan
- Filter Kategori: SeGET /api/data?category_id=2) untuk mengambil data yang sudah difilter, sehingga lebih efisien untuk dataset yang sangat besar. [12].

4.5 Pembahasan

Hasil dari perancangan dan implementasi ini tidak hanya menunjukkan keberhasilan teknis, tetapi juga memberikan wawasan mendalam mengenai penerapan praktis konsep sistem informasi dalam konteks yang sederhana. Pembahasan akan mengupas signifikansi dari setiap komponen yang dibangun, membahasnya dengan masalah awal, dan memancarkan esensinya.

4.5.1 Analisis Pemilihan Teknologi (Node.js dan Express.js)

Keputusan untuk menggunakan Node.js dan Express.js sebagai fondasi server aplikasi terbukti sangat tepat dan sejalan dengan tujuan penelitian, yaitu menciptakan solusi yang ringan dan mudah dipahami bagi pemula. Sifat *non-blocking I/O* dari Node.js memungkinkan server untuk menangani banyak koneksi secara bersamaan tanpa gangguan oleh operasi I/O yang lambat seperti query basis data. Ini secara implisit menjawab kebutuhan akan efisiensi sumber daya, yang sering menjadi masalah di instansi kecil dengan perangkat keras terbatas. Berbeda dengan framework berat lainnya yang memerlukan konfigurasi yang rumit (misalnya, dependency injection container di Java Spring atau struktur MVC yang kaku di beberapa framework PHP), Express.js memberikan harapan maksimal. Pengembang dapat membangun aplikasi dari nol, menambahkan hanya middleware yang diperlukan, dan memahami alur data secara transparan dari permintaan hingga respons. Hal ini secara drastis mengurangi kurva pembelajaran, membuat fokus tetap pada logika bisnis daripada pada kompleksitas framework itu sendiri, sebagaimana disarankan oleh Kim dan Lee mengenai efisiensi *virtualisasi ringan* [3].

4.5.2 Implikasi Arsitektur Tiga Tingkat

Penerapan arsitektur three-tier bukanlah sekadar pilihan teknis, melainkan sebuah keputusan strategi yang berdampak langsung pada kemudahan pemeliharaan dan pengembangan sistem di masa depan. Pemisahan yang jelas antara lapisan presentasi, logika, dan data secara praktis berarti:

- **Modularitas:** Perubahan pada salah satu lapisan tidak akan merusak lapisan lain secara sistemik. Misalnya, jika tim pengembang memutuskan untuk mengubah tampilan antarmuka pengguna dari situs web menjadi aplikasi mobile, mereka hanya perlu membangun kembali *Presentation Tier* baru yang berkomunikasi dengan *Application Tier* yang sudah ada. *Tingkat Aplikasi* dan *Tingkat Data* tidak perlu diubah.
- **Skalabilitas:** Setiap lapisan dapat diskalakan secara independen. Jika basis data menjadi bottleneck, data basis server dapat ditingkatkan (misalnya, dengan menambah RAM atau menggunakan *clustering database*) tanpa harus mengubah kode aplikasi di server.
- **Pembagian Tugas yang Jelas:** Dalam tim pengembangan, seorang *pengembang front-end* dapat fokus pada HTML, CSS, dan JavaScript, sementara *pengembang back-end* fokus

pada logika API dan interaksi data dasar. Ini meningkatkan efisiensi dan spesialisasi kerja. [13].

Arsitektur ini secara langsung mengatasi masalah sistem yang monolitik dan sulit dipertahankan, yang seringkali menjadi hasil dari pengembangan aplikasi yang terburu-buru tanpa perencanaan yang matang.

4.5.3 Dampak pada Efektivitas Pengelolaan Data

Prototipe yang dihasilkan secara langsung memberikan solusi atas berbagai masalah yang diidentifikasi pada proses pengelolaan data manual:

- **Mengurangi Kesalahan Input (Human Error):** Sistem mengimplementasikan validasi data di sisi server (dalam controller). Misalnya, field "nama" tidak boleh kosong, dan "email" harus mengikuti format email yang valid. Ini mencegah data tidak valid masuk ke data dasar, sesuatu yang sulit dihindari dalam penggunaan spreadsheet yang tidak memiliki aturan validasi yang ketat.
- **Menghilangkan Redundansi Data:** Dengan menggunakan basis data relasional yang ternormalisasi (seperti yang ditunjukkan pada ERD), informasi seperti "kategori" disimpan dalam satu tabel terpisah . Data master hanya menyimpan kategori ID ( ). Hal ini memastikan konsistensi data dan menghemat ruang penyimpanan, berbeda dengan spreadsheet di mana nama kategori mungkin ditulis ulang secara manual di setiap baris, yang berisiko menyebabkan inkonsistensi (misalnya, "Teknik Informatika" vs "Tek. Informatika").**categoriescategory_id**
- **Meningkatkan Keamanan dan Aksesibilitas:** Data tidak lagi tersebar di berbagai file spreadsheet yang mungkin disimpan di komputer pribadi. Semua data dimaksudkan di satu data berbasis server. Akses ke data dikontrol melalui API. Pengguna dengan peran 'user' hanya dapat membaca data (Read-only), sementara 'admin' memiliki hak penuh (CRUD). Laporan aktivitas ( ) menyediakan *jejak audit* yang lengkap, mencatat siapa, kapan, dan apa yang dilakukan terhadap data. Ini adalah peningkatan keamanan dan akuntabilitas yang mendasar dibandingkan dengan sistem manual.*activity_logs.* [6].

4.5.4 Keterbatasan dan Arah Pengembangan Masa Depan

Meskipun berhasil mencapai tujuan, prototipe ini memiliki batasan yang menjadi peluang berharga untuk penelitian dan pengembangan selanjutnya:

1. **Keamanan Tingkat Lanjut:** Sistem saat ini menggunakan autentikasi berbasis sesi sederhana. Untuk lingkungan produksi yang lebih aman, penerapan otentikasi berbasis token seperti JWT (JSON Web Token) atau integrasi dengan penyedia identitas pihak ketiga (OAuth 2.0) menjadi suatu keharusan.
2. **Pengujian Performa (Load Testing):** Sistem belum diuji dalam kondisi beban tinggi. Penelitian selanjutnya dapat melakukan pengukuran beban menggunakan alat seperti Apache JMeter atau Artillery untuk mengukur berapa banyak permintaan secara bersamaan yang dapat ditangani server sebelum waktu respons melambat.
3. **Deployment pada Cloud:** Penelitian ini hanya mendemonstrasikan deployment di server lokal. Langkah logis berikutnya adalah mengeksplorasi proses penerapan pada layanan cloud (seperti AWS EC2, Google Cloud Compute Engine, atau Heroku) untuk memahami tantangan dan manfaat dari skalabilitas dan ketersediaan tinggi yang ditawarkan oleh cloud.
4. **Fitur Pelaporan dan Analitik:** Saat ini, sistem hanya menampilkan data dalam bentuk tabel. Pengembangan fitur untuk membuat laporan (misalnya, laporan PDF) atau visualisasi data (grafik dan diagram) akan meningkatkan nilai aplikasi secara signifikan. [14].

5. KESIMPULAN

Berdasarkan keseluruhan penelitian yang telah dilakukan, mulai dari perancangan konsep hingga implementasi dan deployment prototipe, dapat disimpulkan secara tegas bahwa

penelitian ini telah berhasil membuktikan bahwa server aplikasi yang sederhana, ringan, dan efektif dapat dibangun menggunakan teknologi modern yang mudah diakses. Model yang dikembangkan memberikan jawaban komprehensif atas tantangan pengelolaan data di lingkungan skala kecil dan pendidikan, sekaligus berfungsi sebagai panduan pembelajaran yang praktis.

Penelitian ini menjawab secara langsung rumusan masalah yang dikemukakan di awal:

1. **Bagaimana merancang dan mengimplementasikan arsitektur server aplikasi yang sederhana?** Jawabannya adalah dengan mengadopsi arsitektur three-tier yang memisahkan tanggung jawab secara jelas, dan mengimplementasikannya menggunakan *stack* teknologi Node.js, Express.js, dan MySQL. Detail perancangan, mulai dari ERD hingga endpoint API, telah dipaparkan secara lengkap dalam makalah ini.
2. **Komponen-komponen apa saja yang diperlukan?** Komponen-komponen esensial tersebut meliputi: (a) Skema basis data relasional yang ternormalisasi untuk memastikan integritas data, (b) Kumpulan endpoint RESTful API yang mengatur operasi CRUD, (c) Lapisan logika bisnis (controller) yang memproses validasi dan aturan, (d) antarmuka pengguna (front-end) yang responsif dan komunikatif, serta (e) Laporan audit untuk pelacakan aktivitas.
3. **Bagaimana cara melakukan penerapan?** Proses deployment telah didemonstrasikan pada lingkungan server lokal Ubuntu, meliputi instalasi dependensi, konfigurasi variabel lingkungan, dan manajemen proses aplikasi, yang berfungsi sebagai panduan praktis bagi pemula.
4. **Betapa efektifnya prototipe yang dihasilkan?** Prototipe ini terbukti jauh lebih efektif daripada metode manual dengan secara signifikan mengurangi kesalahan input data melalui validasi, menghilangkan redundansi data melalui normalisasi data dasar, meningkatkan keamanan melalui kontrol akses ringkas, dan mempercepat akses informasi melalui antarmuka sepertinya yang responsif.

Kontribusi utama dari penelitian ini adalah dua kali lipat. Pertama, secara praktis, dihasilkan sebuah prototipe fungsional yang dapat langsung diadopsi atau dikembangkan lebih lanjut oleh instansi kecil. Kedua, secara teoretis dan edukatif, disajikan sebuah dokumentasi end-to-end yang lengkap dan terstruktur, mulai dari analisis kebutuhan hingga penerapan, yang dapat dijadikan bahan ajar atau referensi utama bagi pelajar dan praktisi pemula yang ingin mempelajari fondasi pengembangan sistem informasi berbasis server. [15].

Meskipun demikian, penelitian ini membuka pintu bagi pengembangan lebih lanjut di bidang keamanan siber, optimasi kinerja, dan pemanfaatan infrastruktur awan. Dengan landasan yang telah dibangun, penelitian selanjutnya dapat fokus pada aspek-aspek lanjutan tersebut untuk menciptakan sistem yang tidak hanya sederhana, tetapi juga tangguh dan siap menghadapi tantangan skala perusahaan.

berkelanjutan.

UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih kepada semua pihak yang telah memberikan dukungan dalam proses penyusunan penelitian ini, baik berupa bantuan pemikiran, arahan, motivasi, maupun fasilitas yang diberikan. Ucapan terima kasih juga disampaikan kepada dosen pembimbing yang telah memberikan panduan serta masukan yang sangat berarti, rekan-rekan yang turut membantu dalam pengumpulan referensi, serta lingkungan akademik yang menyediakan suasana kondusif untuk belajar dan meneliti. Seluruh kontribusi tersebut sangat membantu penulis dalam menyelesaikan penelitian ini dengan baik.

DAFTAR PUSTAKA

[1] D. Wijaya dan L. Fajar, "Implementasi REST API untuk integrasi sistem informasi berbasis web," *Jurnal Teknologi dan Sistem Informasi* , vol. 7, tidak. 1, hlm. 22–30, 2021.

[2] B. Santoso dan R. Sari, "Optimasi manajemen basis data menggunakan model server singkatnya," *Jurnal Informasi dan Komputasi* , vol. 9, tidak. 3, hlm.144–153, 2022.

[3] H. Kim dan S. Lee, "Virtualisasi ringan untuk penyebaran server skala kecil," *IEEE Access* , vol. 9, hlm. 55123–55134, 2021.
[4] B. Oetomo, "Konsep dasar sistem informasi dalam pengelolaan data modern," *Jurnal Sistem Informasi* , vol. 14, tidak. 2, hal.55–63, 2018.
[5] D. Wijaya dan L. Fajar, "Perancangan server aplikasi untuk pengelolaan logika bisnis web," *Jurnal Teknologi dan Sistem Informasi* , vol. 7, tidak. 1, hlm. 22–30, 2021.
[6] A. Rahman dan Y. Pratama, "Implementasi arsitektur three-tier pada sistem informasi terintegrasi," *Jurnal Rekayasa Teknologi Informasi* , vol. 5, tidak. 2, hal.101–109, 2020.
[7] B. Santoso dan R. Sari, "Penerapan model CRUD dalam pengembangan aplikasi berbasis web," *Jurnal Informasi dan Komputasi* , vol. 9, tidak. 3, hlm.144–153, 2022.
[8] DM Kroenke dan DJ Auer, *Pemrosesan Basis Data: Dasar-dasar, Desain, dan Implementasi* , edisi ke-14. Pearson, 2019.
[9] AS Tanenbaum dan M. Van Steen, *Sistem Terdistribusi: Prinsip dan Paradigma* , edisi ke-3. Prentice Hall, 2016.
[10] W. Stallings, *Sistem Operasi: Internal dan Prinsip Desain* , edisi ke-9. Pearson, 2018.
[11] RS Pressman, *Rekayasa Perangkat Lunak: Pendekatan Praktisi* . McGraw-Hill, 2014.
[12] L. Bass, P. Clements, dan R. Kazman, *Arsitektur Perangkat Lunak dalam Praktik* , edisi ke-3. Addison-Wesley, 2013.
[13] J. Heizer dan B. Render, *Manajemen Operasi* , edisi ke-11. Pearson, 2017.
[14] D. Oaks dan J. Smith, "Desain server yang efisien untuk aplikasi pemrosesan data ringan," *Jurnal Internasional Aplikasi Komputer* , vol. 182, no. 35, hlm. 1–10, 2019.
[15] A. Rahman dan F. Yuliani, "Model server aplikasi terpusat untuk meningkatkan efisiensi manajemen data," *Jurnal Penelitian Sistem Informasi* , vol. 12, no. 2, hlm. 55–64, 2020.