DOI: https://doi.org/10.70656/ijcse.v2i01.298

Alasan Penggunaan Arsitektur MVC dalam Implementasi CRUD

Dinda Krisnauli Pakpahan[1], Fathiyya Nafisah[1], Hesi Desta Lestari[1], Fidia Dewi Wulandari Batu Bara[1], Rahma Hidayati Fitrah[1], Robi Septian Subhan[1], Aditya Saputra[1], Bavio Robia Rahmadan[1], Yudi Setiawan[2], Wili Novrian[2]
[1]Program Studi Informatika, Universitas Bengkulu, Indonesia
[2]Program Studi Sistem Informasi, Universitas Bengkulu, Indonesia
Corresponding author: dindakrisnauli04@gmail.com

Abstract*—Arsitektur Model-View-Controller (MVC) telah menjadi arsitektur perangkat lunak yang populer dalam pengembangan aplikasi berbasis web dan mobile, khususnya dalam implementasi operasi CRUD (Create, Read, Update, Delete). MVC membagi aplikasi menjadi tiga komponen utama: Model, View, dan Controller, yang masing-masing memiliki peran dan tanggung jawab yang terpisah. Pemisahan ini tidak hanya meningkatkan efisiensi dan keamanan, tetapi juga memperkuat modularitas sistem. Dengan struktur yang jelas, pengembang dapat bekerja pada bagian yang berbeda secara paralel tanpa saling mengganggu, serta mempermudah proses debugging dan pengujian. Model menangani pengelolaan data dan interaksi dengan database, View mengatur tampilan antarmuka pengguna, sementara Controller bertanggung jawab untuk mengelola alur aplikasi. Selain itu, MVC mendukung prinsip reusability, memungkinkan komponen Model digunakan kembali dalam berbagai bagian aplikasi atau proyek lain. Keuntungan lain yang signifikan dari MVC adalah kemudahan dalam pemeliharaan jangka panjang, karena setiap komponen dapat dimodifikasi atau diperbarui secara terpisah. Berdasarkan berbagai penelitian, arsitektur MVC terbukti efektif dalam mempercepat proses pengembangan, meningkatkan keamanan, dan mempermudah manajemen data, khususnya dalam aplikasi berbasis CRUD. Oleh karena itu, penerapan arsitektur MVC sangat dianjurkan dalam pengembangan aplikasi berbasis database maupun sistem informasi berskala besar, yang memerlukan pengelolaan data yang terstruktur, efisien, dan aman.

Kata kunci: MVC, CRUD, Reusability

I. PENDAHULUAN

Salah satu perkembangan paling signifikan dalam bidang teknologi informasi adalah kemajuan pesat dalam pengembangan aplikasi web. Web, sebagai platform layanan informasi berbasis internet, telah menjadi salah satu teknologi yang paling diminati di seluruh dunia. Aplikasi web tidak hanya memudahkan akses informasi, tetapi juga menjadi sarana penting untuk komunikasi, bisnis, hiburan, dan pendidikan. Keunggulannya yang terletak pada kemampuan untuk mengakses informasi secara global selama perangkat terhubung ke internet menjadikan web sebagai alat yang sangat fleksibel dan efektif dalam penyebaran data dan interaksi pengguna. Namun, dalam pengembangan aplikasi web, sering kali timbul tantangan terkait pengelolaan data, tampilan antarmuka, serta interaksi antar komponen sistem. Untuk itu, penting bagi pengembang untuk memilih pendekatan yang dapat membantu memisahkan logika aplikasi, mempermudah pengelolaan dan pemeliharaan, serta meningkatkan efisiensi pengembangan. Salah satu pendekatan yang banyak digunakan untuk menangani tantangan tersebut adalah Model-View-Controller (MVC).

Model-View-Controller (MVC) adalah pola desain perangkat lunak yang membagi aplikasi menjadi tiga komponen utama: Model, View, dan Controller. Model berfungsi untuk mengelola data dan logika aplikasi, View bertanggung jawab untuk menampilkan antarmuka pengguna, sedangkan Controller bertindak sebagai penghubung yang menerima input dari pengguna dan memperbarui model atau tampilan sesuai kebutuhan. Pembagian ini memungkinkan setiap komponen untuk berfungsi secara terpisah namun tetap berinteraksi satu sama lain dengan cara yang terstruktur.

Dalam konteks implementasi CRUD (Create, Read, Update, Delete), pola MVC sangat relevan karena memudahkan pengelolaan proses manipulasi data. Dengan MVC, proses pengambilan data (Read), penambahan data (Create), pembaruan data (Update), dan penghapusan data (Delete) dapat diorganisir dengan jelas di dalam Model, sementara tampilan atau antarmuka yang dilihat oleh pengguna dapat dipisahkan ke dalam View. Controller bertugas untuk mengatur bagaimana data dikirimkan dan diproses antara kedua komponen tersebut.

Selain itu, penggunaan MVC dalam pengembangan aplikasi web juga memberikan beberapa keuntungan signifikan, antara lain memudahkan pemeliharaan dan pengembangan aplikasi, mengurangi duplikasi kode, serta memungkinkan pengujian yang lebih efisien. Sebab, setiap bagian dalam aplikasi dapat dikelola secara terpisah tanpa saling mempengaruhi. Misalnya, perubahan pada tampilan (View) tidak akan mempengaruhi logika aplikasi (Model), begitu juga sebaliknya. Dengan adanya pemisahan tanggung jawab yang jelas, tim pengembang dapat bekerja lebih efisien dan lebih cepat dalam menyelesaikan proyek.

Secara keseluruhan, penggunaan arsitektur MVC dalam implementasi CRUD tidak hanya memudahkan dalam pengelolaan aplikasi, tetapi juga meningkatkan kualitas pengembangan perangkat lunak, mempercepat debugging dan pemeliharaan, serta memungkinkan pengembang untuk lebih fokus pada masing-masing aspek fungsionalitas aplikasi. Oleh karena itu, pemahaman dan penerapan pola MVC menjadi langkah penting dalam menciptakan aplikasi web yang efisien, mudah dikelola, dan dapat berkembang dengan cepat.

II. TINJAUAN PUSTAKA

Model-View-Controller (MVC) merupakan pola arsitektur perangkat lunak yang membagi aplikasi menjadi tiga komponen utama yang masing-masing memiliki tanggung jawab yang terpisah. Komponen pertama adalah Model, yang berfungsi untuk mengelola data dan logika bisnis aplikasi, termasuk interaksi dengan database. Komponen

kedua adalah View, yang bertanggung jawab untuk menyajikan antarmuka pengguna (UI) dan menampilkan data kepada pengguna. Komponen ketiga adalah Controller, yang mengelola input pengguna dan alur logika aplikasi, menghubungkan antara Model dan View. Pemisahan ini bertujuan untuk memisahkan logika aplikasi dari tampilan, yang pada gilirannya memudahkan pengembangan, pengujian, dan pemeliharaan aplikasi, serta memungkinkan tim pengembang untuk bekerja secara lebih efisien dan paralel.

Menurut penelitian yang dilakukan oleh Dimas Putra et al. (2016), penerapan arsitektur MVC dalam sistem informasi berbasis web sangat membantu dalam memisahkan struktur kode antara data dan tampilan. Dengan pemisahan ini, sistem menjadi lebih fleksibel dan lebih mudah dalam melacak kesalahan (tracing errors) pada berbagai bagian aplikasi. Dalam penelitian tersebut, penggunaan framework CodeIgniter yang berbasis MVC mampu menangani proses pengajuan proposal secara efisien, berkat struktur aplikasi yang modular dan rapi. Penerapan MVC juga memberikan keuntungan dalam hal pemeliharaan sistem, karena pengelolaan kode lebih terorganisir dan setiap komponen dapat dimodifikasi atau diperbarui secara terpisah tanpa mempengaruhi bagian lainnya.

Selain itu, Ali Ikhwan dan Arris Fahrian (2022) dalam penelitiannya tentang penerapan MVC pada sistem informasi penggajian karyawan menunjukkan bahwa arsitektur ini sangat efektif dalam mempermudah pengelolaan data serta menjaga keamanan dan keteraturan aplikasi, terutama dalam proses CRUD (Create, Read, Update, Delete) yang digunakan untuk mengelola data pegawai dan penggajian. Mereka menemukan bahwa MVC memisahkan secara jelas logika bisnis dan tampilan, memungkinkan proses pengelolaan data yang lebih terstruktur dan aman.

Penelitian oleh Syahrul Mubarok et al. (2022) juga memberikan bukti bahwa penerapan MVC pada sistem informasi kawasan agrowisata memberikan hasil yang positif. Dalam sistem ini, MVC mempermudah pengembangan berbagai fitur penting seperti transaksi, pengelolaan data produk dan wisata, serta integrasi antara pengguna sistem, seperti BUMDes dan UMKM Desa. MVC tidak hanya meningkatkan efisiensi pengkodean (coding) tetapi juga mendukung kolaborasi antara berbagai pihak yang terlibat dalam pengelolaan data real-time. Keuntungan lain yang ditemukan adalah kemampuan untuk mempercepat proses pengembangan aplikasi, mengingat struktur yang modular dan terorganisir dengan baik memungkinkan pengembangan dan pembaruan dilakukan secara lebih cepat dan efisien.

Dari hasil penelitian-penelitian tersebut, dapat disimpulkan bahwa arsitektur MVC sangat ideal digunakan dalam implementasi CRUD karena komponen Model langsung berinteraksi dengan database untuk melakukan operasi Create, Read, Update, dan Delete. Controller bertugas mengelola permintaan dari pengguna dan meneruskannya ke Model yang sesuai, sedangkan View hanya menyajikan hasil CRUD kepada pengguna tanpa mencampur kode logika. Penerapan MVC dalam pengembangan aplikasi berbasis web secara signifikan meningkatkan efisiensi pengembangan, kemudahan dalam pemeliharaan, serta memastikan keamanan data yang lebih baik. Terutama dalam aplikasi yang melibatkan banyak entitas dan proses transaksi, penerapan MVC terbukti memberikan solusi yang efisien dan mudah diadaptasi untuk pengelolaan data secara terstruktur dan aman.

III. HASIL DAN PEMBAHASAN

A. Model-View-Controller (MVC)

Model-View-Controller (MVC) adalah sebuah arsitektur perangkat lunak yang memisahkan aplikasi ke dalam tiga komponen utama, yaitu model, view, dan controller. Dalam pengembangan aplikasi, bagian yang paling sering mengalami perubahan adalah antarmuka pengguna (user interface). Perubahan pada user interface sering kali berhubungan dengan logika bisnis aplikasi, yang menyebabkan perubahan pada antarmuka dapat mempengaruhi logika bisnis tersebut. Untuk mengatasi masalah ini, banyak pengembang memilih untuk membagi pengembangan sistem ke dalam tiga bagian terpisah: Model, View, dan Controller.

1. Model

Model merupakan representasi langsung dari struktur database, mencakup desain tabel hingga relasi antar tabel yang membentuk basis data aplikasi. Peran utama Model adalah mengelola seluruh aspek data, mulai dari pengambilan data (retrieval), penyimpanan data baru (insertion), manipulasi data, hingga validasi agar data yang diproses tetap konsisten dan sesuai aturan bisnis yang berlaku. Dengan Model, proses interaksi aplikasi dengan database menjadi terorganisir dan terpisah dari logika tampilan, sehingga memudahkan pengelolaan data secara efisien dan terstruktur.

2. View

View bertugas untuk menampilkan data yang diberikan oleh Model dalam bentuk antarmuka pengguna yang mudah dipahami. Selain sebagai media penyajian data, View juga berfungsi sebagai penghubung antara aktivitas pengguna dengan sistem, dengan cara meneruskan setiap interaksi atau input dari pengguna kepada Controller. Dengan mengelompokkan seluruh kode tampilan dan presentasi pada satu tempat, View mempermudah pengembang untuk melakukan perubahan visual atau desain antarmuka tanpa harus mengubah logika bisnis yang ada, sehingga menjaga kestabilan aplikasi dan konsistensi data yang ditampilkan.

3. Controller

Controller berperan sebagai pengatur perilaku aplikasi dengan menerjemahkan tindakan pengguna menjadi instruksi yang dapat dipahami oleh Model. Controller menjadi jembatan utama antara View dan Model, mengelola setiap interaksi yang terjadi pada antarmuka pengguna serta meresponnya dengan mengubah data melalui Model atau memperbarui tampilan melalui View. Dalam Controller terdapat berbagai metode yang dirancang untuk menangani alur aplikasi dan menanggapi input pengguna secara dinamis, menjadikannya pusat logika pengendalian yang mengatur bagaimana aplikasi merespon setiap aksi dalam sistem.

B. CRUD

Operasi CRUD sendiri merupakan fondasi utama dalam pengelolaan data pada aplikasi berbasis web. Fitur CRUD seperti Create, Read, Update dan Delete. memungkinkan pengguna untuk melakukan operasi dasar terhadap data secara terstruktur dan efisien. Berikut ini adalah pengembangan dari

materi yang telah ada dengan penambahan detail untuk setiap tahap dalam proses CRUD:

1. Create (Membuat): Operasi ini digunakan untuk menambahkan data baru ke dalam sistem. Pada aplikasi berbasis web, ini sering kali melibatkan pengisian formulir dengan data yang diperlukan, seperti nama, alamat email, atau informasi lainnya. Setelah formulir diisi, data tersebut disimpan dalam database untuk digunakan lebih lanjut. Fungsi Create memungkinkan sistem untuk berkembang dengan menambah data baru sesuai kebutuhan pengguna.
2. Read (Membaca): Operasi ini memungkinkan pengguna untuk melihat atau mengakses data yang telah disimpan dalam database. Data yang diambil dapat berupa daftar item (seperti daftar user atau produk) atau detail informasi tertentu dari sebuah entri. Fungsi ini adalah salah satu yang paling sering digunakan, karena memberikan pengguna akses ke informasi yang relevan, tanpa mengubah atau menghapusnya.
3. Update (Memperbarui): Update memungkinkan pengguna untuk memperbarui atau mengubah data yang sudah ada dalam sistem. Ini termasuk perubahan informasi yang sudah terdaftar, seperti mengedit nama, alamat email, atau data lainnya. Fitur ini sangat penting untuk menjaga agar data tetap relevan dan akurat sesuai dengan perubahan yang terjadi di dunia nyata.
4. Delete (Menghapus): operasi yang digunakan untuk menghapus data yang sudah tidak dibutuhkan lagi. Pengguna dengan hak akses yang sesuai dapat menghapus entri dari database untuk mengelola data yang tersimpan. Fitur ini penting untuk menjaga kebersihan dan efisiensi sistem, meskipun biasanya disertai dengan konfirmasi untuk menghindari penghapusan data secara tidak sengaja.

C. Relevansi MVC Terhadap Operasi Crud

Berdasarkan hasil analisis dan pembahasan dari beberapa studi terdahulu, dapat disimpulkan bahwa arsitektur Model-View-Controller (MVC) memiliki relevansi yang sangat kuat dan strategis dalam implementasi operasi CRUD (Create, Read, Update, Delete). Arsitektur ini menjadi solusi ideal dalam pembangunan sistem informasi modern karena kemampuannya memisahkan komponen logika bisnis, antarmuka pengguna, dan alur kendali secara terstruktur. Pemisahan peran antara Model, View, dan Controller memberikan fleksibilitas tinggi dalam proses pengembangan perangkat lunak. Hal ini memungkinkan tim pengembang untuk mengerjakan bagian tampilan dan logika secara paralel tanpa saling mengganggu, serta mempercepat proses debugging dan pengujian. Dengan struktur ini, perubahan pada salah satu komponen (misalnya antarmuka pengguna) tidak akan berdampak langsung pada komponen lainnya (seperti database atau logika aplikasi), sehingga sistem menjadi lebih modular dan maintainable.

Selain itu, implementasi CRUD dalam konteks MVC menjadi lebih rapi karena seluruh proses pengelolaan data dilakukan di dalam Model, sementara alur interaksi pengguna dikendalikan oleh Controller, dan hasilnya disajikan oleh View. Ini menjadikan proses CRUD tidak hanya efisien, tetapi juga lebih aman, karena akses ke data dilakukan melalui mekanisme yang terkendali dan tervalidasi.

Dukungan dari berbagai penelitian memperkuat kesimpulan ini. Studi oleh Ali Ikhwan dan Arris Fahrian (2022) menunjukkan bahwa sistem penggajian berbasis MVC mampu menangani proses CRUD secara efisien dan aman. Penelitian lain oleh Makarim et al. (2024) membuktikan bahwa arsitektur MVC mempercepat pengembangan dan mempermudah manajemen data tugas mahasiswa. Sementara itu, Susetyo et al. (2018) memperlihatkan bagaimana struktur modular MVC (dalam bentuk HMVC) mendukung sistem informasi berskala besar dengan kebutuhan CRUD yang kompleks. Dari sisi akademis dan teknis, MVC bukan hanya cocok digunakan untuk membangun aplikasi berbasis CRUD, tetapi juga layak dijadikan sebagai fondasi utama dalam merancang sistem informasi berbasis web dan mobile secara umum. Kemampuan MVC untuk menyederhanakan proses pengembangan, mendukung pembagian kerja tim, serta menjaga keamanan dan keteraturan data menjadikannya arsitektur yang relevan, efisien, dan berkelanjutan. Terdapat beberapa keuntungan MVC dalam CRUD, yaitu:

D. Reusability dan Modularitas Komponen

Salah satu kekuatan utama MVC terletak pada kemampuannya dalam memisahkan tugas secara jelas ke dalam tiga komponen utama: Model, View, dan Controller. Komponen Model bertanggung jawab terhadap logika data dan komunikasi dengan database, View mengatur tampilan antarmuka pengguna, dan Controller mengelola alur aplikasi dan interaksi antar komponen. Pemisahan ini memungkinkan masing-masing bagian dikembangkan, diuji, dan dimodifikasi secara mandiri tanpa memengaruhi komponen lainnya. Selain itu, komponen Model sangat mendukung prinsip reusability karena dapat digunakan kembali dalam berbagai bagian aplikasi atau bahkan di proyek lain. Struktur ini juga memungkinkan tim pengembang untuk bekerja secara paralel pada bagian yang berbeda, sehingga meningkatkan efisiensi dan produktivitas kerja.

E. Memudahkan Debugging dan Testing

Pemisahan komponen dalam MVC sangat mendukung proses debugging dan pengujian (testing). Karena setiap komponen berdiri sendiri, pengujian dapat dilakukan secara terpisah untuk masing-masing modul. Isolasi semacam ini memudahkan deteksi dan perbaikan bug secara spesifik. Struktur ini juga mendukung praktik Test-Driven Development (TDD), yaitu metode pengembangan di mana pengujian ditulis terlebih dahulu sebelum kode utama dikembangkan.

F. Konsistensi Data dan Validasi

Dalam konteks keamanan dan kualitas data, MVC menawarkan mekanisme validasi yang konsisten dan terpusat. Aturan validasi, seperti format input, panjang data, nilai minimum atau maksimum, serta kolom wajib, dapat didefinisikan langsung di dalam Model. Dengan demikian, setiap operasi CRUD akan mengikuti aturan yang sama, menjaga konsistensi data di seluruh aplikasi. Penerapan validasi terpusat ini juga mendukung prinsip Don't Repeat Yourself (DRY), sehingga meminimalisir kesalahan pengolahan data.

G. Efektivitas Modularitas, Pemeliharaan, dan Keamanan

Penggunaan arsitektur MVC juga memberikan keunggulan dalam hal modularitas sistem, kemudahan pemeliharaan, dan peningkatan keamanan. Penelitian oleh

Makarim et al. (2024) menunjukkan bahwa pengembangan aplikasi manajemen tugas dengan MVC memungkinkan penanganan fitur CRUD lebih cepat dan efisien. Ali Ikhwan & Arris Fahrian (2022) dalam sistem penggajian karyawan menunjukkan bahwa penambahan fitur baru dapat dilakukan tanpa memengaruhi komponen lain. Susetyo et al. (2018) menambahkan bahwa struktur HMVC mampu mengatur hak akses dan CRUD lintas pengguna secara aman dan modular.

Untuk memberikan gambaran yang lebih jelas mengenai bagaimana arsitektur Model-View-Controller (MVC) digunakan dalam implementasi operasi CRUD (Create, Read, Update, Delete), berikut adalah studi kasus implementasi sistem manajemen tugas berbasis web. Sistem ini memungkinkan pengguna untuk mengelola tugas mereka, yang mencakup penambahan tugas baru, melihat daftar tugas, memperbarui status tugas, dan menghapus tugas yang sudah selesai atau tidak relevan.

1. Model bertanggung jawab untuk mengelola data dalam aplikasi, yang dalam konteks ini adalah data terkait tugas yang akan dikelola. Model ini berfungsi untuk berinteraksi dengan database, melakukan query, dan mengelola data yang terkait dengan tugas.

```php
class TaskModel {
    private $db;

    public function __construct($db) {
        $this->db = $db;
    }

    public function createTask($title, $description) {
        $sql = "INSERT INTO tasks (title, description) VALUES (:title, :description)";
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':title', $title);
        $stmt->bindParam(':description', $description);
        $stmt->execute();
    }

    public function getAllTasks() {
        $sql = "SELECT * FROM tasks";
        $stmt = $this->db->prepare($sql);
        $stmt->execute();
        return $stmt->fetchAll();
    }

    public function updateTask($taskId, $title, $description) {
        $sql = "UPDATE tasks SET title = :title, description = :description WHERE id = :id";
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':title', $title);
        $stmt->bindParam(':description', $description);
        $stmt->bindParam(':id', $taskId);
        $stmt->execute();
    }

    public function deleteTask($taskId) {
        $sql = "DELETE FROM tasks WHERE id = :id";
        $stmt = $this->db->prepare($sql);
        $stmt->bindParam(':id', $taskId);
        $stmt->execute();
    }
}
```

Gambar 1. Contoh Model Tugas (PHP)

2. View bertanggung jawab untuk menampilkan antarmuka pengguna (UI) dan menerima input dari pengguna. View ini akan menampilkan data yang dikendalikan oleh Controller dan memberikan formulir atau tampilan untuk melakukan operasi CRUD.

```php
<!DOCTYPE html>
<html>
<head>
    <title>Task Management</title>
</head>
<body>
    <h1>Task Management</h1>
    <form method="post" action="index.php?action=create">
        <input type="text" name="title" placeholder="Task Title" required>
        <textarea name="description" placeholder="Task Description" required></textarea>
        <button type="submit">Add Task</button>
    </form>

    <h2>All Tasks</h2>
    <ul>
        <?php foreach ($tasks as $task) { ?>
            <li>
                <?php echo $task['title']; ?> - <?php echo $task['description']; ?>
                <a href="index.php?action=edit&id=<?php echo $task['id']; ?>">Edit</a> |
                <a href="index.php?action=delete&id=<?php echo $task['id']; ?>">Delete</a>
            </li>
        <?php } ?>
    </ul>
</body>
</html>
```

Gambar 2. Contoh Tampilan Tugas (HTML)

3. Controller bertanggung jawab untuk menerima input dari pengguna, mengelola interaksi antara Model dan View, serta menentukan apa yang harus dilakukan berdasarkan aksi pengguna (misalnya, menambah, mengedit, menghapus tugas).

```php
class TaskController {
    private $taskModel;

    public function __construct($taskModel) {
        $this->taskModel = $taskModel;
    }

    public function create() {
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $title = $_POST['title'];
            $description = $_POST['description'];
            $this->taskModel->createTask($title, $description);
            header("Location: index.php");
        }
    }

    public function read() {
        $tasks = $this->taskModel->getAllTasks();
        include 'view/tasks.php';  // Memanggil tampilan untuk menampilkan tugas
    }

    public function edit() {
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
            $taskId = $_POST['id'];
            $title = $_POST['title'];
            $description = $_POST['description'];
            $this->taskModel->updateTask($taskId, $title, $description);
            header("Location: index.php");
        } else {
            $taskId = $_GET['id'];
            $task = $this->taskModel->getTaskById($taskId);
            include 'view/edit_task.php'; // Tampilan untuk mengedit tugas
        }
    }

    public function delete() {
        $taskId = $_GET['id'];
        $this->taskModel->deleteTask($taskId);
        header("Location: index.php");
    }
}
```

Gambar 3. Contoh Controller Tugas (PHP)

Alur kerja arsitektur MVC dalam sistem manajemen tugas berbasis operasi CRUD berjalan dengan cara yang sangat terstruktur. Pada tahap Create, pengguna memasukkan data tugas baru melalui antarmuka View, yang kemudian dikirimkan ke Controller. Controller bertugas untuk

memproses data yang dimasukkan oleh pengguna, dan setelah itu, data tersebut disimpan dalam database melalui Model. Pada tahap Read, Controller mengambil data tugas yang ada dari Model dan kemudian menampilkannya di View, sehingga pengguna dapat melihat daftar tugas yang tersimpan. Untuk tahap Update, ketika pengguna memilih tugas yang ingin diedit, Controller mengambil data tugas yang ada, memperbarui informasi sesuai permintaan pengguna melalui Model, dan mengarahkan pengguna kembali ke daftar tugas yang telah diperbarui. Terakhir, pada tahap Delete, ketika pengguna memutuskan untuk menghapus suatu tugas, Controller menerima permintaan tersebut, kemudian menghapus data tugas yang dimaksud dari database melalui Model. Proses ini memastikan bahwa setiap operasi dilakukan secara terpisah dan efisien, dengan pemisahan yang jelas antara logika aplikasi, tampilan pengguna, dan pengelolaan data.

KESIMPULAN

Berdasarkan pembahasan yang telah dilakukan, dapat disimpulkan bahwa arsitektur Model-View-Controller (MVC) sangat relevan dan efektif dalam implementasi operasi CRUD (Create, Read, Update, Delete) pada pengembangan aplikasi berbasis web. Pembagian aplikasi menjadi tiga komponen utama—Model, View, dan Controller—membantu memisahkan tanggung jawab masing-masing bagian, sehingga meningkatkan efisiensi, keamanan, dan modularitas sistem. Pemisahan ini memungkinkan pengembang untuk bekerja secara paralel tanpa saling mengganggu, mempercepat proses debugging dan pengujian, serta mempermudah pemeliharaan aplikasi dalam jangka panjang.

Melalui implementasi MVC, proses pengelolaan data dalam aplikasi dapat dilakukan dengan lebih terstruktur, di mana Model bertanggung jawab untuk interaksi dengan database, View mengatur tampilan antarmuka pengguna, dan Controller mengelola alur aplikasi berdasarkan input pengguna. Keuntungan lain dari penerapan MVC adalah konsistensi dalam pengelolaan data, kemudahan dalam memperbarui fitur atau komponen tanpa mempengaruhi bagian lain, serta keamanan yang lebih terjamin melalui pemisahan logika bisnis dan tampilan.

Selain itu, penelitian dan studi kasus yang diulas memperlihatkan bahwa penggunaan MVC dalam sistem berbasis CRUD tidak hanya meningkatkan efisiensi pengembangan, tetapi juga mempermudah pengelolaan data yang melibatkan banyak entitas dan proses transaksi. Dengan demikian, penerapan arsitektur MVC sangat dianjurkan dalam pengembangan aplikasi berbasis database dan sistem informasi berskala besar, yang membutuhkan pengelolaan data yang terstruktur, aman, dan mudah dipelihara.

DAFTAR PUSTAKA

[1] S. M. N. Muhammad, F. A. Mauladi, R. Kurniawan, dan R. Sanjaya, "Sistem Informasi Kawasan Agrowisata menggunakan Konsep Model View Control berbasis Web," Edumatic: Jurnal Pendidikan Informatika, vol. 6, no. 1, pp. 88–97, Jun. 2022. doi: 10.29408/edumatic.v6i1.5422

[2] A. Ikhwan dan A. Fahrian, "Sistem Informasi Penggajian Karyawan Pada Basnul Coffee Berbasis Web," Impression JTI, vol. 1, no. 2, pp. 30–35, 2022

[3] D. Putra, Azhar, dan A. Fata, "Rancang Bangun Aplikasi Pengajuan Proposal Judul Tugas Akhir Berbasis Web dan SMS Gateway Menggunakan Konsep Model View Control," Jurnal Infomedia, vol. 1, no. 2, pp. 17–22, Des. 2016. ISSN: 2527-9858I. S. Jacobs and C. P. Bean, "Fine particles, thin films and

[4] Y. F. A. W. Firdaus dan W. Maharani, "Analisis Performansi Framework PRADO dan CakePHP pada Aplikasi Web AJAX," dalam Seminar Nasional Aplikasi Teknologi Informasi (SNATI), Yogyakarta, 21 Jun. 2008. ISSN: 1907-5022, pp. C-47–C-48

[5] M. A. B. W. Anuar, "New Aspect of Code Automation for Web Application Framework," Ph.D. dissertation, Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Feb. 2023

[6] I. F. Makarim et al., "Penerapan Arsitektur MVC pada Website Pengumpul Tugas Menggunakan PHP," dalam Seminar Nasional Informatika Bela Negara (SANTIKA), vol. 4, 2024, pp. 270–271. ISSN (Online): 2747-0563

[7] Falahah dan R. Puspita, "Perancangan Arsitektur Pemrograman Berbasis MVC (Studi Kasus Sistem Informasi Asset)," dalam Konferensi Nasional Teknologi Informasi dan Aplikasinya (KNTIA), Palembang, 13 Sep. 2014

[8] D.-P. Pop dan A. Altar, "Designing an MVC Model for Rapid Web Application Development," Procedia Engineering, vol. 69, pp. 1172–1179, 2014. doi: 10.1016/j.proeng.2014.03.106

[9] W. Mualim dan G. U. Putra, "Implementasi Framework MVC pada Sistem Informasi Akademik di STMIK YADIKA Bangil," Jurnal SPIRIT, vol. 9, no. 2, pp. 35–39, Nov. 2017. ISSN: 2085-3092

[10] R. Maulana and R. D. Wulandana, "Pengaruh Kualitas Layanan Terhadap Kepuasan dan Loyalitas Pelanggan IndiHome di Kota Mataram," *Jurnal Ekonomi dan Bisnis Jagaditha*, vol. 8, no. 1, pp. 49–57, 2021.