DOI: https://doi.org/10.70656/ijcse.v2i01.310

Analisis Integrasi Komponen Arsitektur MVC dalam Pengembangan Aplikasi Web

Diosi Putri Arlita[1], Intan Oktaviani Presia[1], Muhammad Tariq Pratama Buhar[1], Fassrah Putra Gunawan[1], Reyhan Maulana Ibrahim[1], Muhammad Sahlan Habibi[1], Muhammad Ripal Rabbani[1], Yudi Setiawan[2], Andang Wijanarko[2]
[1]Program Studi Informatika, [2]Program Studi Sistem Informasi
Fakultas Teknik, Univeristas Bengkulu
Corresponding author: diosiputriarlita06@gmail.com

Abstract*—Perkembangan teknologi informasi yang pesat mendorong kebutuhan akan sistem aplikasi yang terstruktur, mudah dikembangkan, dan mampu berintegrasi dengan sistem lain. Arsitektur Model-View-Controller (MVC) telah menjadi pendekatan populer dalam pengembangan perangkat lunak karena kemampuannya dalam memisahkan komponen logika bisnis, tampilan, dan kontrol alur data. Penelitian-penelitian menunjukkan bahwa penerapan MVC secara konsisten dapat meningkatkan efisiensi pengembangan, kemudahan pemeliharaan, serta fleksibilitas integrasi antar modul dan sistem yang berbeda. Selain itu, kombinasi MVC dengan pendekatan arsitektur lainnya seperti Service-Oriented Architecture (SOA) dan pemanfaatan RESTful API memperkuat kemampuan integrasi lintas platform dan sistem. Integrasi tersebut memungkinkan sistem informasi saling berkomunikasi dan bertukar data secara efektif, seperti yang terlihat pada implementasi di sektor pendidikan, pengelolaan data akademik, manajemen sampah, dan e-commerce. Dengan desain modular, MVC menjadi fondasi kuat dalam pengembangan aplikasi web modern yang tidak hanya fokus pada fungsionalitas, namun juga pada interoperabilitas antar sistem yang beragam.

Keywords*—Model-View-Controller, integrasi sistem, arsitektur perangkat lunak, SOA, RESTful API.

I. PENDAHULUAN

Perkembangan teknologi yang pesat khususnya dalam bidang teknologi informasi dan komunikasi mendorong kebutuhan yang kuat dalam perkembangan aplikasi yang canggih namun tetap mudah digunakan. Pola arsitektur yang banyak digunakan dalam pengembangan aplikasi, terutama dalam *website* adalah MVC (Model, *View, Controller*). MVC terbagi dalam tiga komponen utama yaitu Model yang mengelola data serta logika bisnis, *View* yang menampilkan dan mengelola tampilan antarmuka pengguna, serta *Controller* yang berfungsi sebagai jembatan penghubung antara Model dan *View* [1]. Penerapan MVC membuat pengembang dapat memisahkan tanggung jawab dari masing-masing komponen sehingga mempermudah dalam proses pengembangan serta pemeliaraan aplikasi[2]. Adanya pemisahan ini, membuat perubahan pada satu bagian tidak langsung memengaruhi bagian lainnya, sehingga dapat memperkecil risiko dari kerusakan sistem secara keseluruhan.

Penerapan konsep MVC dalam sistem informasi manajemen data dapat meningkatkan kerja pengelolaan data, hal ini dikarenkan adanya pembagian yang jelas antar komponen. Integrasi dan kesinabungan yang baik antara Model, *View*, dan *Controller* sangat penting agar terciptanya konsistensi data dan responsivitas aplikasi. Namun, dalam penerapannya mengintegrasikan ketiga komponen ini bukanlah hal yang mudah, terutama dalam mengelola interaksi yang kompleks antara proses bisnis dan antarmuka pengguna. Pemahaman yang mendalam tentang bagaimana mekanisme MVC dapat saling terintegrasi dengan baik sangat dibutuhkan agar aplikasi dapat berfungsi dengan baik, optimal serta dapat dengan mudah dikembangkan untuk kedepannya. Permana dan Sihanto [3] juga menegaskan dalam implementasi MVC untuk pengembangan aplikasi portal customer relationship harus dirancang dengan pendekatan integrasi yang sistematis agar menjamin kelancaran komunkasi antar komponen dan juga kemudahan dalam pengembangan fitur baru nantinya.

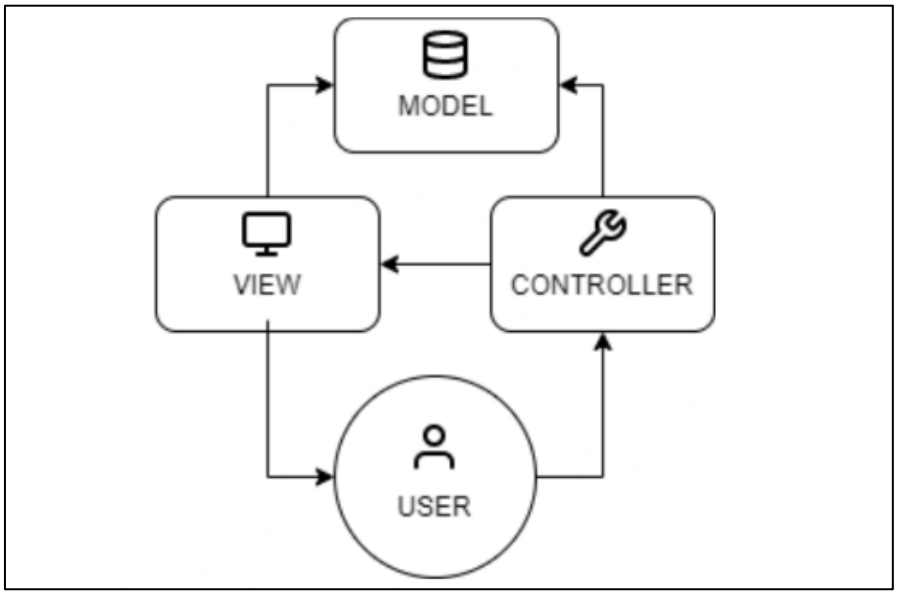
Penerapan dalam mengintegrasikan MVC yang baik telah diterapkan dalam beberapa sistem informasi seperti sistem akademik berbasis web [4], aplikasi pengelolaan bank sampah[5], dan sistem perpustakaan [6], yang dimana semuanya menunjukkan bahwa integrasi MVC yang baik menjadi faktor utama dan sangat penting dalam pengembangan aplikasi lintas domain. Dengan melihat berbagai tantangan tersebut, penting bagi setiap pengembang serta peneliti untuk memahami setiap proses integrasi secara komprehensif dan baik. Hal ini dilakukan agar arsitektur MVC dapat saling terintegrasi dengan baik antar satu komponen dengan komponen lainnnya. Maka dari itu, artikel ini akan membahas terkait mekenisme integrasi Model, *View*, dan *Controller* agar dapat salling terintegrasi dengan baik dan optimal, guna mendukung pengembangan aplikasi yang terstruktur, mudah dikelola, serta dapat dikembangkan sesuai kebutuhan.

II. METHODOLOGI

A. Konsep Dasar Arsitektur MVC

Arsitektur Model-View-Controller (MVC) adalah pola desain perangkat lunak yang dirancang untuk memisahkan logika aplikasi menjadi tiga bagian utama: Model yang menangani manajemen data dan logika bisnis, View yang berperan sebagai antarmuka pengguna, dan Controller yang berfungsi sebagai penghubung serta pengatur alur data dari input pengguna menuju model dan view. Konsep ini pertama kali diperkenalkan oleh Trygve Reenskaug pada tahun 1979 saat bekerja di laboratorium riset Xerox PARC dengan pengembangan bahasa pemrograman Smalltalk[1].

Seiring waktu, arsitektur MVC telah mengalami evolusi dan diadopsi secara luas dalam berbagai teknologi pengembangan aplikasi, khususnya dalam sistem berbasis web dan desktop. Pendekatan ini terbukti sangat bermanfaat karena memiliki karakteristik seperti modularitas, reusability, dan maintainability. Modularitas berarti bahwa masing-masing komponen dapat dikembangkan, diuji, dan diperbaiki secara terpisah tanpa memengaruhi komponen lainnya. Reusability memungkinkan pengembang untuk menggunakan kembali komponen yang sama di berbagai bagian aplikasi atau bahkan dalam proyek yang berbeda. Sedangkan maintainability memudahkan proses pemeliharaan sistem di masa depan karena kode lebih terorganisir dan terbagi berdasarkan tanggung jawabnya masing-masing.



Gambar 1. Alur Arsitektur MVC

Berbagai framework modern telah mengadopsi arsitektur ini secara default, seperti Laravel, Ruby on Rails, Django, Spring MVC, hingga ASP.NET MVC. Hal ini menunjukkan bahwa konsep MVC tetap relevan dan fleksibel untuk berbagai bahasa dan platform pemrograman. Kombinasi antara arsitektur yang rapi dan dukungan dari komunitas pengembang menjadikan MVC sebagai fondasi penting dalam pengembangan aplikasi kontemporer, terutama ketika aplikasi tersebut dituntut untuk bersifat interaktif, modular, dan terintegrasi secara luas dengan sistem lain.

Arsitektur MVC membagi interaktif sistemnya ke dalam tiga buah komponen yang memiliki spesialisasi perannya masing-masing. Tiga buah komponen tersebut yaitu model, view, dan controller (lihat Gambar 1). Model pada arsitektur MVC berupa kelas yang berasal dari objek tertentu. Model adalah abstraksi dari entitas objek yang secara spesifik tidak memiliki tanggung jawab tentang Graphical User Interface (GUI). Representasi dari model sebagai elemen GUI disebut View. View dapat dilihat sebagai pembungkus di sekitar Model yang mampu menampilkan data yang terenkapsulasi di dalam Model. Setiap View memiliki pengontrol yang saling berhubungan satu sama lain. Controller bertanggung jawab atas semua tindakan dalam mengatur hubungan antara Model dengan View. Sebuah Model dapat direpresentasikan atas beberapa View yang berbeda. Hal tersebut dapat terjadi karena sebuah View dapat merujuk pada data yang sama. Enkapsulasi GUI secara spesifik dijelaskan di dalam sebuah View dan tiap-tiap view diatur dan dikelola oleh Controller dengan mengacu pada Model yang memiliki data terkait[10].

Penerapan MVC juga mendukung interoperabilitas sistem, terutama ketika digabungkan dengan pendekatan arsitektur layanan seperti Service-Oriented Architecture (SOA). Dengan cara ini, setiap modul aplikasi dapat difungsikan sebagai layanan mandiri, memungkinkan sistem saling berkomunikasi dengan baik, bahkan pada arsitektur besar dan kompleks [10][11].

B. Penerapan Arsitektur MVC dalam Pengembangan Sistem

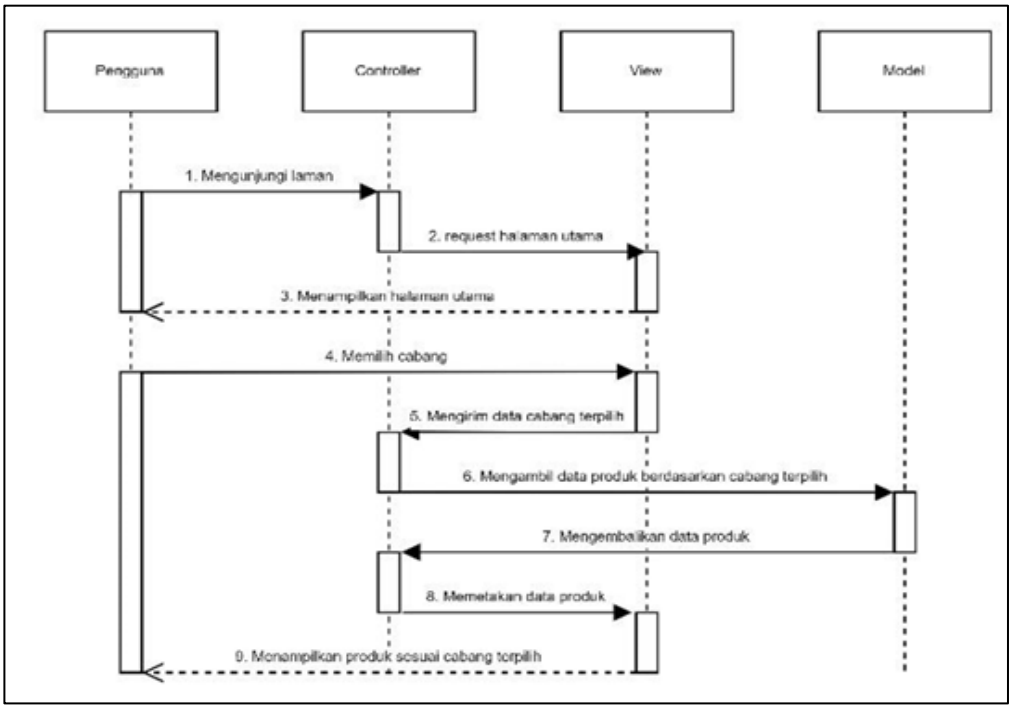
Berbagai studi menunjukkan bahwa arsitektur MVC telah banyak diterapkan dalam pengembangan sistem informasi untuk berbagai kebutuhan, baik di bidang pendidikan, bisnis, hingga lingkungan. Setiap implementasi menunjukkan kelebihan MVC dalam hal modularitas dan pemisahan tugas.

Portal Customer Relationship Management (CRM) Permana dan Sihanato [3] mengembangkan portal CRM menggunakan MVC untuk menangani interaksi pelanggan secara sistematis. Model menyimpan data pelanggan, View menyajikan informasi layanan, dan Controller mengatur proses komunikasi antara keduanya. Arsitektur ini memungkinkan personalisasi layanan tanpa mengubah keseluruhan sistem. Dalam konteks lain, Ramadan [7] mengintegrasikan MVC dalam sistem informasi manajemen

data terintegrasi pada institusi pendidikan tinggi, yang membantu mengelola data operasional secara konsisten.

Sistem Informasi Pengelolaan Bank Sampah Munir [4][5] menerapkan arsitektur MVC untuk pengembangan aplikasi bank sampah berbasis web. Sistem ini memisahkan proses pengumpulan sampah, perhitungan poin, dan penukaran barang dalam modul-modul yang berbeda. Implementasi ini mendukung efisiensi dan pengelolaan data yang lebih terintegrasi.

Sistem Toko Online Alip et al. [9] membangun website toko online menggunakan arsitektur MVC. Penerapan ini memungkinkan pengelolaan produk, transaksi, dan data pengguna dilakukan dalam modul terpisah. MVC memberikan kemudahan dalam pengembangan fitur baru tanpa mengganggu fungsionalitas utama.



Gambar 2. Sequence Diagram Portal Customer Relationship Management (CRM)

Gambar 3. Implementasi MVC dalam struktur direktori Yii2 Framework pada Sistem Informasi Pengelolaan Bank Sampah Munir (2018)

Selain implementasi di atas, Gunawan et al. [8] melakukan analisis performa MVC dalam framework Java Server Faces dan menunjukkan bahwa MVC, meskipun memiliki overhead pada awalnya, jauh lebih unggul dalam jangka panjang dari segi organisasi kode dan maintainability.

C. Perbandingan MVC dengan Pendekatan Arsitektur Lain

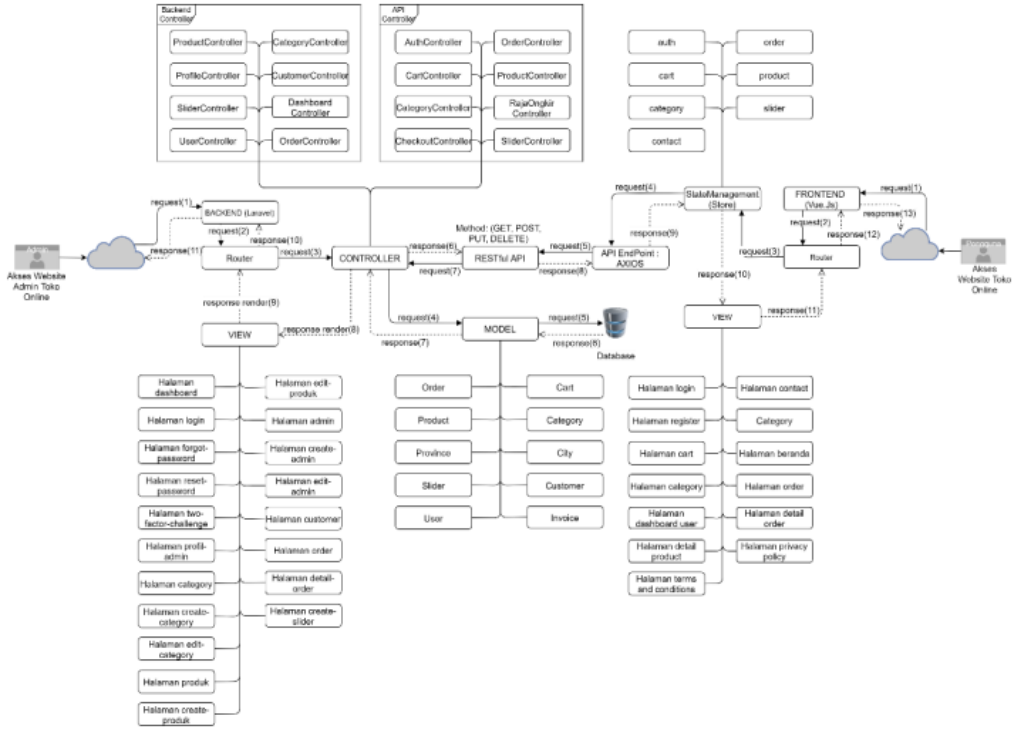
Model View Controller memiliki banyak keunggulan, namun bukan satu-satunya pendekatan yang digunakan dalam pengembangan perangkat lunak. Model tradisional seperti Waterfall mengandalkan tahapan yang linier, mulai dari analisis kebutuhan hingga implementasi dan pemeliharaan [12]. Model ini masih digunakan, tetapi kurang fleksibel untuk sistem yang memerlukan perubahan cepat dan bertahap.

Waterfall model adalah model pengembangan perangkat lunak tradisional yang bersifat sekuensial. Setiap tahapan seperti analisis kebutuhan, desain sistem, implementasi, pengujian, hingga pemeliharaan dilakukan secara berurutan. Hal ini berbeda dengan pendekatan MVC yang lebih modular dan mendukung pemisahan tanggung jawab sejak awal pengembangan.

SOA merupakan pendekatan arsitektur berbasis layanan, di mana setiap komponen sistem dikemas sebagai layanan mandiri yang dapat diakses melalui protokol komunikasi standar seperti RESTful API. Pendekatan ini berfokus pada

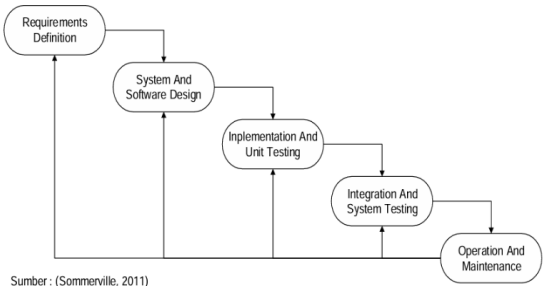
DOI: https://doi.org/10.70656/ijcse.v2i01.310

interoperabilitas antar sistem, dan cocok untuk sistem berskala besar atau terdistribusi.

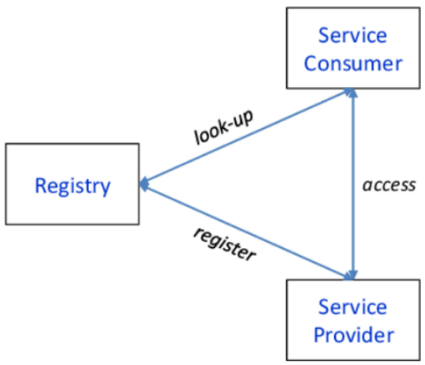


Gambar 2. Implementasi MVC dalam struktur direktori Yii2 Framework pada Sistem Informasi Pengelolaan Bank Sampah Munir (2018)

Kemudian RAD adalah pendekatan pengembangan sistem yang menekankan pada kecepatan dan iterasi dalam siklus pengembangan. Fokus RAD ada pada keterlibatan pengguna secara langsung dan pengembangan prototipe yang cepat untuk memvalidasi kebutuhan sistem.



Gambar 5. Waterfall Model



Gambar 6. Diagram Arsitektur SOA

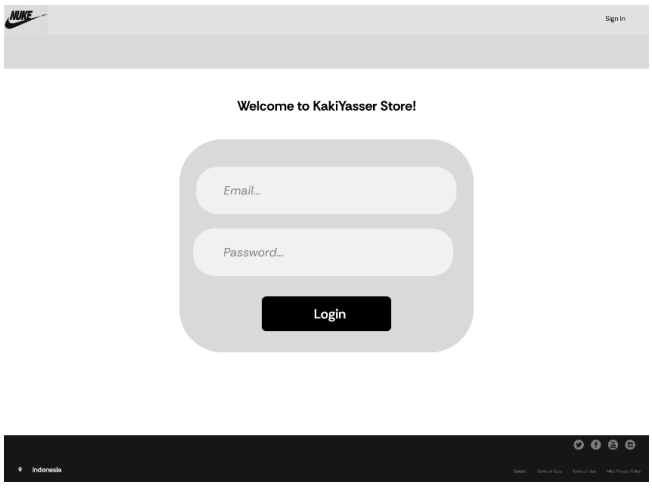
III. HASIL DAN PEMBAHASAN

Berdasarkan dari rancangan arsiktektur MVC (Model, *View, Controller),* sistem dikembangkan dengan memisahkan tugas pada masing-masing komponen yang bertujan untuk meningkatkan modularitas dan juga kemudahan dalam pengembangan serta pemeliharaan. Pada bagian ini akan dipaparkan hasil dari implementasi dan bagaimana tiap komponen saling terintegrasi untuk mendukung fungsi utama dari sistem, mulai dari proses autentikasi dari pengguna, pengelolaan transaksi, hingga pengaturan data oleh admin.

Sebagai bagian dari implemetasi tersebut, sistem ini mampu menampilkan deskripsi produk secara detail (spesifikasi bahan, ukuran, foto multi-angle) dengan tata letak yang responsif untuk memastikan kemudahan navigasi. Konten website diperkaya elemen dinamis seperti video demonstrasi produk. Pengguna umum dapat menjelajahi katalog tanpa batasan dan aktivasi akun wajib dilakukan sebelum memasuki tahap pemesanan untuk memvalidasi identitas pembeli. Integrasi fitur pembaruan status real-time yang mengirimkan alert ke admin via dashboard atau email begitu pembayaran sukses diverifikasi, mengurangi risiko human error. Keamanan platform dioptimalkan melalui kombinasi autentikasi dua faktor, enkripsi SSL, dan audit berkala untuk melindungi informasi sensitif pengguna dan transaksi.

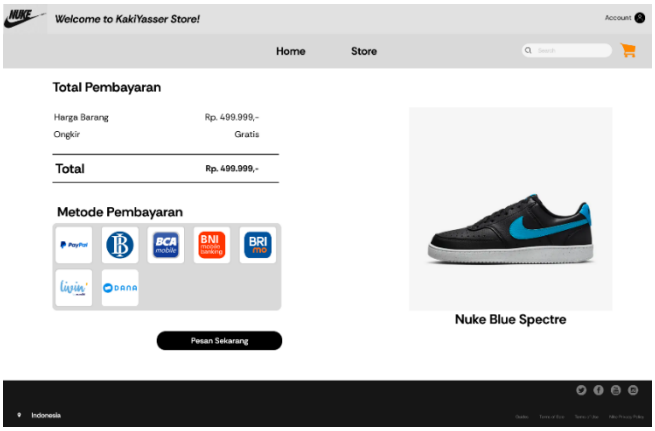
DOI: https://doi.org/10.70656/ijcse.v2i01.310

Pada halaman proses autentifikasi atau login model memproses validasi pengguna dan mengembalikan hasil berhasil atau gagal ke Controller. View menampilkan form login kolom Email, Password, dan tombol Login serta feedback error misalnya “Email atau password salah”. Setelah user mengklik Login, View mengirimkan request HTTP POST ke Controller. Kemudian Controller mengatur input POST dan menjalankan validasi awal. Setelah itu Controller memanggil Model untuk memeriksa kredensialpengguna. Jika validasi berhasil, Controller akanmenampilkan View berupa “Dashbaorad”, jika gagal Controller akan menampilkan View “Login” beserta pesan error. Model berfokus pada operasi data dan logika autentikasi tanpa mengetahui apapun tentang tampilan form. Hasil validasi, berhasil ataupun gagal akan dikembalikan ke Controller dalam bentuk objek.



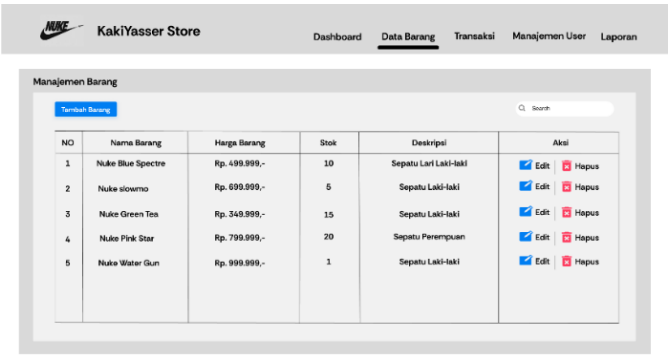
Gambar 7. Login Page

Dalam proses transaksi model menangani dalam pembuatan record order, pemanggilan API eksternal bank untuk pembayaran, serta pengelolaan transaksi database, yang dimana jika salah satu proses insert ataupun update gagal, seluruh transaksi dibatalkan dengan tujuan untuk menjaga konsistensi data. Pada tampilan transaksi terdapat tampilan rincian dari total pembayaran dan metode pembayaran. Menampilkan rincian harga barang (Rp. 499.999,-), ongkir gratis dan total.



Gambar 8. Transaksi

Daftar ikon metode pembayaran dan tombol pesan sekarang akan memicu request checkout. Saat admin/penjual (atau user) menekan pesan sekarang, controller mengatur data cart produk, qty dan pilihan metode pembayaran dikirim via POST ke endpoint, mengecek validasi dan menentukan respons, jika sukses maka redirect ke halaman konfirmasi, Jika gagal render ulang View checkout dengan pesan error.



Gambar 9. Tampilan Admin

Model juga mengelola operasi CRUD dan pencarian data yang ada pada modul admin, yang memungkinkan adanya pengeloaan data barang secara langsung. Dalam modul admin, view menyediakan tombol tambah barang yang memunculkan modal/form input. Tombol edit dan hapus pada tiap baris dan pencarian untuk memfilter daftar barang secara real-time. Controller mengatur pada tambah barang dan edit serta hapus pada barang, sesuai permintaan dari View.

Mekanisme integrasi MVC yang baik diwujudkan melalui komunikasi yang jelas dan terstruktur antar setiap komponen. Controller berperan sebagai pengatur alur data dan logika, yang mengubungkan pemintaaan dari View ke Model begitu juga sebaliknya, sehingga setiap perubahan data dapat ditangani langsung dengan tepat. Model yang bertanggung jawab pada pengeloahan data tidak terpengaruh oleh perubahan tampilan, sementara View fokus terhadap presentasi dan interaksi pengguna tanpa harus mengelola data secara langsung. Dengan pemisahan tugas yang jelas antar Model, View, dan Controller serta alur komunikasi yang terstruktur, aplikasi mampu berjalan secara efisien dan mudah untuk dilakukan pengembangan. Integrasi dari ketiga komponen ini dapat memmungkinkan proses autentikasi, pengelolaan transaksis, dan administrasi data dapat berjalan dengan optimal tanpa adanya gangguan, yang menjadi inti dari arsitektur MVC yang saling terintegrasi. Selain itu, hasil implementasi mnunjukkan penggunaan arsitektur MVC meningkatkan kemampuan aplikasi dalam mengelola setiap data, seperti pada fitur pembaruan status pembayaran yang memerikan notifikasi langsung kepada admin, Hal ini tidak hanya mengurangi risiko tingkat gangguan, tetapi juga memepercepat pemantauan dan juga pengambilan keputusan. Keamanan dari aplikasipun dapat tejaga dengan baik melalui penerapan autentikasi, yang menunjukkan MVC tidak hanya berfokus pada fungsionalitas, tetapi juga memberikan dasar yang kuat dalam pengembangan aplikasi.

KESIMPULAN

Berdasarkan hasil dan pembahasan yang telah dipaparkan, dapat disimpulkan bahwa integrasi yang efektif antara komponen Model, View, dan Controller dalam arsitektur MVC berperan sangat penting dalam pengembangan aplikasi. Integrasi arsitekstur MVC dapat terwujud secara efektif melalui pembagian tugas yang jelas dan komunikasi yang

terstruktur antar tiap komponen. Model berperan dalam mengelola data dan logika bisnis secara independen dari tampilan, View berfokus pada presentasi dan interaksi pengguna, sedangkan Controller berperan sebagai penghubung yang mengatur alur data dan logika antara View dan Model. Implementasi integrasi MVC yang baik meningkatkan modularitas, responsivitas, serta keamanan dari aplikasi. Integrasi yang baik ini tidak hanya menjamin kelancaran proses autentikasi, pengelolaan transaksi, tetapi juga meningkatkan efiensi pengelolaan data. Dengan mekanisme tersebut aplikasi menjadi lebih mudah dikembangkan, dipelihara, dan juga dapat beradaptasi dengan kebutuhan yang terus berkembang.

DAFTAR PUSTAKA

[1] Wijaya, K., & Christian, A. (2019). Implementasi Metode Model View Controller (MVC) Dalam Rancang Bangun Website SMK Yayasan Bakti Prabumulih. Jurnal Khatulistiwa Informatika, 21(1), 95-102.

[2] Makarim, I. F., et al. (2024). Penerapan Arsitektur MVC pada Website Pengumpul Tugas Menggunakan PHP. Prosiding Seminar Nasional Informatika Bela Negara, 4, 270–278.

[3] Permana, P. T. I., & Sihanato, A. N. (2024). Implementasi Arsitektur MVC Dalam Pengembangan Aplikasi Customer Relationship Portal. Jurnal Teknologi Informasi, 10(1), 50–57.

[4] Munir, S. (2016). Perancangan Sistem Informasi Akademik Berbasis Web Menggunakan Framework MVC Pada STT Terpadu Nurul Fikri. Jurnal Informatika Terpadu, 2(1).

[5] Munir, S. (2018). Implementasi Arsitektur Aplikasi MVC Pada Perancangan Aplikasi Bank Sampah Berbasis Web. Jurnal Teknologi Terpadu, 4(2).

[6] Wiharto, Y., & Irawan, A. (2017). Perancangan Sistem Perpustakaan Menggunakan Model View Controller (MVC) Dengan Metode OMT Pada SMP Negeri 44 Palembang. Jurnal Teknologi Informasi, 12(1).

[7] Ramadan, R. (2020). Penerapan Konsep Model View Controller Pada Sistem Informasi Manajemen Data Terintegrasi. Information Management For Educators and Professionals, 5(1), 45–54.

[8] Gunawan, G., Lawi, A., & Adnan, A. (2016). Analisis Arsitektur Aplikasi Web Menggunakan Model View Controller (MVC) pada Framework Java Server Faces. Scientific Journal of Informatics, 3(1), 55–67.

[9] Alip, A., et al. (2021). Implementasi Arsitektur Model View Controller pada Website Toko Online. Jurnal Bumigora Information Technology (BITe), 3(2), 135–150.

[10] Warkim, W., & Sensuse, D. I. (2017). Model Integrasi Sistem dengan Pendekatan Metode SOA dan MVC pada Pusat Penelitian LIPI. Jurnal Teknik Informatika dan Sistem Informasi, 3(1).

[11] Issarny, V., et al. (2016). Revisiting SOA for the IoT: A Middleware Perspective. Proceedings of the International Conference on Service-Oriented Computing (ICSOC), 3–17.

[12] Sommerville, I. (2011). Software Engineering (9th ed.). Massachusetts: Addison-Wesley.