E-ISSN : 2623-1700
P-ISSN : 1693-3672

Implementasi Arsitektur MVC Dalam Pengembangan Aplikasi Customer Relationship Portal

Pramudya Tamir Indra Permana[1], Andreas Nugroho Sihanato[2]
Universitas Pembangunan Nasional Veteran Jawa Timur
ipramudya0@gmail.com

Abstrak

Portal pelanggan (*customer relationship portal*) merupakan sebuah wadah bagi pelanggan yang dibangun dengan tujuan agar mempermudah mereka dalam memperoleh informasi terkait bisnis yang diusung oleh perusahaan dengan berbasis teknologi informasi. *Website* sebagai salah satu bentuk implementasi dari perkembangan teknologi yang dapat memungkinkan pengguna dalam bertukar informasi secara cepat dan mudah. Dengan memanfaatkan perkembangan teknologi yang begitu pesat, proses pengembangan aplikasi *web* portal pelanggan pada penelitian ini menggunakan arsitektur MVC yang terdiri atas *model*, *view*, dan *controller*. Arsitektur MVC dipilih karena memiliki tingkat fleksibilitas yang cukup tinggi dalam membagi logika pemrograman ke dalam beberapa bagian sesuai dengan perannya masing-masing, serta memberikan kemudahan bagi pengembang aplikasi dalam melakukan pemeliharaan aplikasi.

Kata kunci: .Portal Pelanggan, Website, Teknologi Informasi, MVC

Abstract

The customer portal is a forum for customers that was built with the aim of making it easier for them to obtain related information used by companies based on information technology. Website as a form of implementation of technological developments that can allow users to exchange information quickly and easily. By taking advantage of the rapid development of technology, the customer web portal application development process in this study uses an MVC architecture consisting of a model, display, and controller. The MVC architecture was chosen because it has a fairly high flexibility in dividing programming logic into several parts according to each part, as well as providing convenience for application developers in performing application maintenance.

Keywords : *Customer Relationship Portal, Website, Information Technology, MVC*

PENDAHULUAN

Evolusi e-bisnis menyebabkan gejolak di banyak bidang. *Website* menjadi inisiatif dalam menciptakan jangkauan yang lebih luas, yang mengarah ke lonjakan permintaan layanan pelanggan yang sangat drastis (Gow, D & Hills, B n.d). *Website* dapat digunakan untuk menghilangkan hambatan geografis serta mempermudah pelanggan dalam membandingkan penawaran secara kompetitif (Gow, D & Hills, B n.d). Namun, seperti bisnis tradisional, e-bisnis bergantung pada seberapa mudah dan cepat pelanggan dapat berhasil menemukan dan membeli produk yang mereka inginkan secara online dengan berkomunikasi secara interaktif dengan situs *web* (Liang n.d.).

Website merupakan sebuah aplikasi yang berisikan dokumen-dokumen berupa teks, gambar, suara, maupun video yang di dalamnya menggunakan protokol HTTP (*hypertext transfer protocol*) dan untuk mengakses menggunakan perangkat lunak *browser* (Wibisono & Susanto 2015). Oleh karena itu, *website* dianggap sebagai bentuk implementasi modern dari teknologi informasi karena pada dasarnya website digunakan sebagai tempat untuk bisa saling bertukar informasi secara fleksibel dengan berbasis *browser*.

Teknologi informasi sangat penting untuk keberhasilan bisnis apa pun. Informasi yang dikumpulkan dalam bisnis sama berharganya dengan sumber daya seperti modal atau sumber daya manusia (Salehi et al. 2012). Informasi yang diberikan oleh website merupakan faktor penting untuk menarik pelanggan dan membuat hubungan antara mereka dan produk atau layanan perusahaan (Salehi et al. 2012). Informasi dapat mencakup tren pasar, keuntungan pelanggan, serta kinerja pembelian pelanggan. Peningkatan informasi pada *website* juga dapat berpotensi menghasilkan pelanggan yang lebih berpengetahuan dalam mengoperasikan layanan yang diberikan oleh perusahaan.

Berbagai macam perusahaan telah mengadopsi istilah portal atau *customer relationship* sebagai jargonnya, dengan harapan meningkatkan perhatian pasar (Gow, D & Hills, B n.d). Di sisi lain, perusahaan perlu fokus pada kualitas pengalaman pelanggan secara keseluruhan (Bradshaw & Brash 2001). Penggunaan utama portal pelanggan adalah sebagai otak dan sistem saraf yang memungkinkan *website* bekerja bersama secara konsisten, melalui semua saluran komunikasi pelanggan sesuai dengan aturan bisnis yang telah ditetapkan (Gow, D & Hills, B n.d). Portal pelanggan menghubungkan fungsi layanan mandiri sebuah bisnis di suatu tempat yang dipersonalisasi sehingga dapat ditampilkan melalui situs web hingga aplikasi seluler milik pengguna. Ketika portal pelanggan terintegrasi dengan sistem lain, maka dapat memungkinkan pelanggan untuk melihat dan membuat perubahan pada informasi akun milik mereka, seperti mengubah alamat pengiriman produk atau membatalkan pembelian suatu produk.

Untuk mengakomodasi kebutuhan pelanggan, perusahaan perlu mengembangkan sebuah aplikasi portal pelanggan berbasis web yang terintegrasi sistem sebagai sarana pemberian informasi mengenai produk yang akan ditawarkan. Ketika mengembangan *website* tentu perlu adanya strategi dan *design pattern* dalam menentukan sebuah rancangan yang efisien. Mengadopsi *design pattern* saat merancang aplikasi *web* dapat mendorong penggunaan kembali (*reusability*) dan konsistensi aplikasi web (Thung et al. 2010). Pemilihan *design pattern* yang salah akan membuat pengembangan aplikasi web menjadi lebih kompleks dan sulit untuk dipertahankan.

Design pattern mengacu pada solusi yang dapat digunakan kembali atau berulang dengan tujuan untuk memecahkan permasalahan yang timbul selama proses pengembangan aplikasi. Setiap *design pattern* memberikan solusi utama untuk masalah tertentu (Vora 2009). Pada umumnya, para pengembang *website* menggunakan *style* mereka sendiri atau kerangka kerja (*framework*) yang telah dibesut oleh perusahaan ketika proses pengembangan *website* (Vora 2009). Pola pengembangan aplikasi yang bersifat *reusable* cocok untuk pengembangan sistem apa pun yang memiliki fungsi atau permasalahan yang serupa. Pendekatan ini menyediakan cara yang nyaman dalam menggunakan kembali baris kode yang berorientasi obyek di antara proyek-proyek selain mempromosikan penggunakan kembali arsitektur dan konsistensi (Horchers 2001). Salah satu bentuk *design pattern* dalam pengembangan *website* adalah arsitektur *Model-View-Controller* (MVC). Pada pendekatan arsitektur MVC, sebuah sistem dibagi menjadi tiga buah komponen, yaitu *model* berperan sebagai area logika, *view* yang menyediakan antarmuka pengguna, dan *controller* berperan dalam mengelola perubahan pada tampilan (Sunardi & Suharjito 2019). Arsitektur MVC memungkinkan proses pengembangan *website* menjadi lebih modular dengan membagi kode program ke beberapa layer, sehingga kompleksitas dalam perancangan *website* dapat berkurang.

Pola arsitektur MVC selain dipilih untuk meningkatkan fleksibilitas dalam merancang *website*, arsitektur MVC juga dinilai cocok dalam pengembangan aplikasi portal pelanggan karena secara singkat logika bisnis pada portal pelanggan dikelompokan menjadi beberapa modul / komponen bagian berdasarkan fungsinya. Setiap modul bertanggung jawab dalam melakukan manipulasi data, menampilkan data dalam bentuk antarmuka pengguna, dan melakukan kontrol dalam perangkat lunak. Sebuah modul yang didefinisikan dapat mewarisi fungsionalitas dari modul-modul yang telah dibuat sebelumnya.

METODE

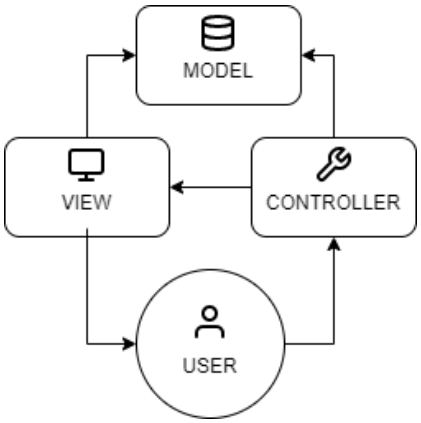
Bagian ini berisi teori-teori pembahasan mengenai metode yang digunakan dalam

E-ISSN : 2623-1700
P-ISSN : 1693-3672

pengembangan perangkat lunak berbasis web pada penelitian kali ini.

A. Model-View-Controller

Pola arsitektur MVC diperkenalkan sebagai salah satu solusi dalam permasalahan pengembangan aplikasi web yang sebelumnya dinilai memiliki skalabilitas buruk dan tingkat pemeliharaan yang rendah (GuangChun, Lu & Hanhong 2003). Dengan mengimplementasikan pola arsitektur MVC, penerapan logika sistem dapat dipisahkan dengan cara penyajian data(GuangChun, Lu & Hanhong 2003).



Gambar 1. Alur Arsitektur MVC

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 (Veit & Herrmann 2003).

B. Rapid Application Development

Istilah *Rapid Application Development* (RAD) pertama kali dicetuskan oleh James Martin pada buku miliknya yang berjudul "Rapid Application Development". Di dalam bukunya, Martin menjelaskan bahwa RAD merupakan sebuah siklus pengembangan sistem (*development life cycle*) yang dirancang untuk memberikan pengembangan yang jauh lebih cepat disertai dengan hasil berkualitas tinggi daripada dicapai menggunakan *development life cycle* tradisional. RAD dirancang untuk mengambil keuntungan maksimal dari pengembangan perangkat lunak yang telah berkembang baru-baru ini (Martin 1991). Karakteristik dari metode RAD mencakup kemampuan perencanaan, pemodelan data, penulisan kode, serta pengujian dan identifikasi kesalahan (Agarwal et al. 2000). Cakupan-cakupan tersebut dapat digali lebih lanjut, sehingga menghasilkan beberapa siklus pengembangan yaitu perencanaan kebutuhan, desain pengguna, dan konstruksi (Agarwal et al. 2000).

Proses pengembangan *website* dengan menggunakan RAD akan menjadi lebih mudah untuk diimplementasikan karena proses pengembangan akan berfokus pada setiap kebutuhan (*development requirement*) (Daud et al. 2010). Penerapan metodologi ini juga memungkinkan pengguna untuk menjadi bagian dari proses pengembangan sistem karena umpan balik atau *feedback* yang diberikan oleh pengguna akan dipertimbangkan setiap kali metode RAD diimplementasi (Daud et al. 2010). Keterlibatan pengguna pada saat proses pengembangan aplikasi akan sangat membantu dalam meningkatkan kepuasan pengguna karena terjadi lebih banyak komunikasi saat pengembangan aplikasi serta pengguna dapat melihat kemajuan dari proses pengembangan.

HASIL DAN PEMBAHASAN

A. Analisis Kebutuhan

Portal pelanggan atau *Customer Relationship Portal* dibangun untuk memudahkan pelanggan dalam mendapatkan informasi seputar produk bisnis yang ditawarkan oleh perusahaan, sehingga perusahaan dapat dengan mudah menjangkau kebutuhan pelanggan dengan maksimal. Pada penelitian ini, produk bisnis yang ditawarkan berupa makanan dan minuman. Informasi terkait produk dapat dicari berdasarkan

E-ISSN : 2623-1700
P-ISSN : 1693-3672

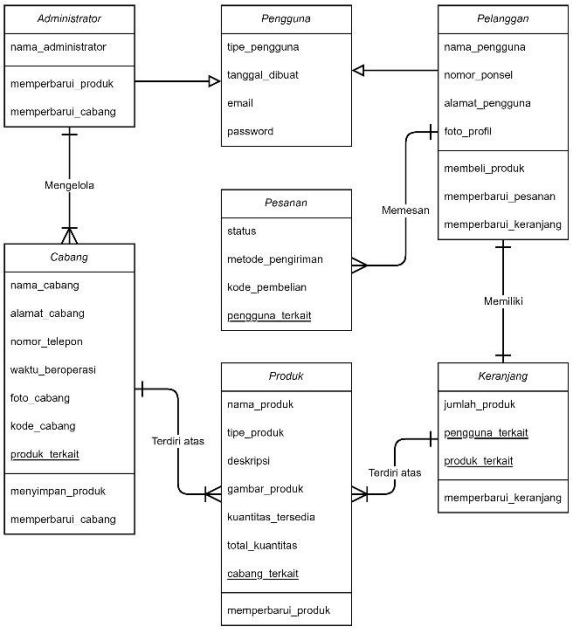
cabang perusahaan yang tersedia di laman portal pelanggan. Informasi dapat berupa harga produk, stok ketersediaan, dan komplemen.

Untuk meningkatkan efisiensi waktu dari proses pengembangan *website*, analisis kebutuhan pengguna diperlukan sebagai langkah awal dalam menentukan fungsionalitas fitur dari aplikasi yang akan dikembangkan. Berikut detail fungsionalitas dari aplikasi *Customer Relationship Portal*:

1) Fitur otentikasi pengguna. Proses otentikasi perlu dilakukan untuk mengidentifikasi identitas dan peran pengguna sebelum mengoperasikan aplikasi. Fitur otentikasi pada aplikasi Customer Relationship Portal berupa *Login* dan *Register*.
2) Fitur manajemen cabang. Fitur ini memungkinkan pengguna untuk melihat informasi mengenai suatu produk berdasarkan cabang yang mereka pilih. Setiap cabang memiliki produknya masing-masing dan tidak berkaitan satu sama lain.
3) Fitur keranjang belanja. Sebelum masuk ke bagian pembayaran, produk yang telah dipilih akan disimpan di keranjang belanja. Pada bagian keranjang, pengguna dapat mengubah informasi pesanan seperti menambah, mengurangi atau menghapus produk dari keranjang belanja.
4) Fitur profil. Bagian ini dibuat untuk memudahkan pengguna dalam mengelola data diri mereka secara fleksibel seperti penerapan beberapa alamat pengiriman, mengubah nama atau nomor ponsel, hingga mengganti foto profil.

B. Pemodelan Sistem

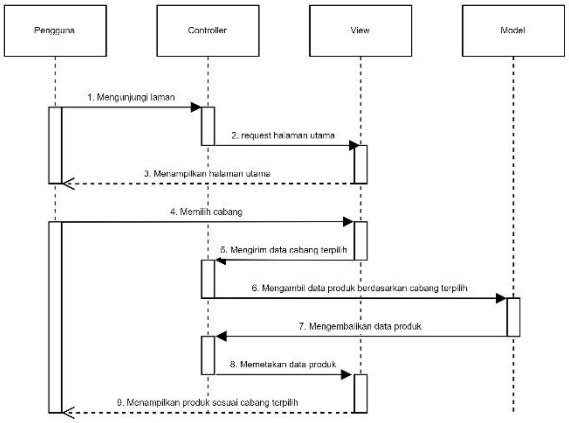
Pemodelan sistem dijabarkan menggunakan metode *Unified Modelling Language* (UML) dengan mengimplementasikan arsitektur MVC. UML mentransfer fitur-fitur pada proses analisis kebutuhan ke dalam bentuk visual agar mudah untuk dipahami. Proses pemodelan sistem dilakukan dengan memperhatikan umpan balik (*feedback*) positif yang diberikan pada saat proses analisis kebutuhan pengguna.



Gambar 2. *Class Diagram*

Pada class diagram yang ditunjukan oleh Gambar 2 menggambarkan bentuk data yang akan disimpan di dalam *database*. Setiap data direpresentasikan oleh sebuah *class* dimana setiap *class* terdapat beberapa atribut yang mendeskripsikan bentuk dari data itu sendiri. Setiap *class* dapat memiliki hubungan dengan *class* lainnya dan *class* yang terhubung dapat berbagi atribut satu sama lain. *Class* Pelanggan dan Administrator mewarisi atribut dari *class* Pengguna. *Class* Administrator memiliki relasi dengan *class* Cabang, sehingga setiap administrator dapat mengelola beberapa cabang. Seorang administrator dapat memperbarui data pada banyak cabang. *Class* Pelanggan memiliki relasi dengan class Pesanan dan class Keranjang. Seorang pengguna dapat memiliki banyak pesanan namun seorang pengguna hanya dapat memiliki satu buah keranjang. *Class* Keranjang memiliki relasi dengan class produk dimana setiap keranjang dapat menampung satu produk atau lebih. *Class* Produk memiliki relasi dengan *class* Cabang, sehingga setiap cabang dapat terdiri atas banyak produk.

E-ISSN : 2623-1700
P-ISSN : 1693-3672



Gambar 3. *Sequence Diagram*

Selain *class diagram*, terdapat juga pemodelan sistem yang berperan sebagai skenario dalam penggunaan sebuah sistem, yaitu *sequence diagram*. Untuk mempermudah proses pengembangan aplikasi, *sequence diagram* diperlukan sebagai bentuk visualisasi dari langkah-langkah yang diperlukan dalam mengoperasikan suatu sistem. Pada Gambar 3 menunjukkan bentuk pemodelan *sequence diagram* dengan mengimplementasikan arsitektur MVC berupa proses pemilihan produk berdasarkan cabang yang dilakukan oleh pelanggan. Pada saat pelanggan mengunjungi *website*, *controller* membaca request yang dilakukan oleh pelanggan melalui *browser*. Kemudian *view* akan memberikan respon balik ke pelanggan berupa tampilan berdasarkan *request* yang diteruskan oleh *controller*. Pada skenario pemilihan produk berdasarkan cabang, *view* akan menerima *request* dari tombol yang dipilih oleh pelanggan dan diteruskan ke bagian *controller*. *Controller* melakukan komunikasi dengan model untuk mendapatkan data yang disimpan di dalam *database* dengan mengirimkan cabang sebagai parameter. *Model* akan mengembalikan data produk ke *controller* untuk dipetakan sesuai dengan cabang yang dipilih. Data hasil pemetaan kemudian akan disajikan ke pelanggan oleh *view*.

C. Implementasi Arsitektur MVC

Setiap fungsionalitas *website* portal pelanggan akan diimplementasi menggunakan arsitektur MVC untuk memudahkan proses pengembangan. Setiap fungsionalitas *website* memiliki sebuah komponen *model*, *view* dan *controller*. Berikut adalah penjabaran setiap fungsionalitas *website* ke dalam bentuk MVC.

1) *Model*

Pembuatan *model* merupakan langkah pertama yang perlu sebelum melakukan implementasi arsitektur MVC. *Model* merupakan representasi dari data yang akan disimpan di dalam *database*. Setiap model didefinisikan sebagai *class* yang bertanggung jawab dalam pengelolaan data berbasis *query*. Proses *query* berupa penambahan data, penghapusan data, pembuatan data baru, hingga pencarian data yang berlangsung di dalam *database*.

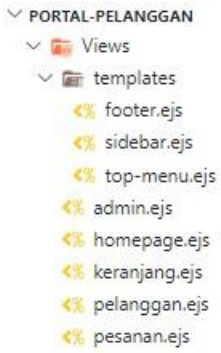
Tabel 1. Peran *Model* pada Aplikasi

Model	Keterangan
pengguna-model.js	Didefinisikan sebagai *model* dasar untuk tiap subjek (admin dan pengguna)
admin-model.js	Menangani proses *query* data admin. *Model* mewarisi fungsionalitas dari pengguna-model.js
pelanggan-model.js	Menangani proses *query* data pelanggan. *Model* mewarisi fungsionalitas dari pengguna-model.js
pesanan-model.js	Menangani proses *query* data pesanan
keranjang-model.js	Menangani proses *query* data keranjang
produk-model.js	Menangani proses *query* data produk
cabang-model.js	Menangani proses *query* data cabang

2) *View*

Pada bagian *view* berperan sebagai antarmuka komponen yang akan ditampilkan ke layar pengguna *website*. Secara umum setiap *view* memiliki templat yang diasosiasikan khusus untuk sebuah tampilan, seperti *top-menu*, *sidebar*, *footer*, dan semacamnya. Proses ini akan meningkatkan tingkat modularitas dari arsitektur MVC. Bagian *view* ditunjukkan pada Gambar 4.

E-ISSN : 2623-1700
P-ISSN : 1693-3672



Gambar 4. Struktur *File* Bagian *View*

Pada Gambar 4 menunjukan bahwa pada *folder views* terdapat sebuah *folder templates* berisikan file *footer*, *sidebar*, dan *top-menu*. *File* yang berada di dalam folder *templates* bersifat *reusable* (dapat digunakan kembali), sehingga memungkinkan setiap *view* memiliki templat di dalamnya. *Footer* berisi informasi umum yang sering dibutuhkan oleh pelanggan, seperti kontak perusahaan dan informasi perusahaan. *Homepage* merupakan tampilan utama dari *website* portal pelanggan yang berisikan informasi berbagai cabang dan produk.

3) *Controller*

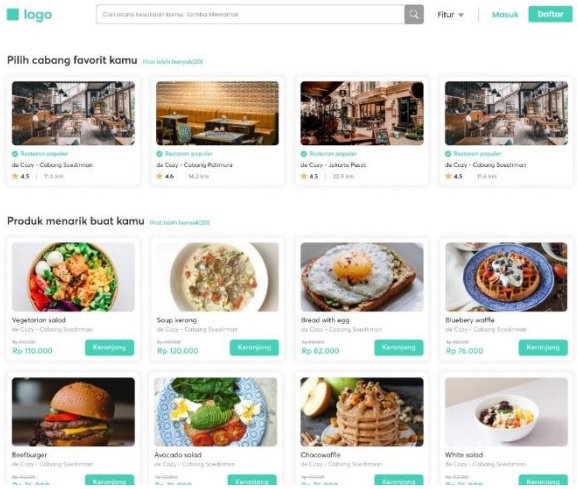
Controller bertanggung jawab dalam mengatur hubungan antara *model* dan *view*. *Controller* melakukan proses penyajian tampilan berdasarkan *request* dari pengguna, serta melakukan pemrosesan data yang dikirimkan oleh pengguna melalui *browser*. Bagian *controller* dijabarkan pada Tabel 2.

Tabel 2. Peran *Controller* pada Aplikasi

Controller	Keterangan
admin-controller.js	Berperan dalam mengelola data cabang dan memperbarui data produk
pelanggan-controller.js	Berperan dalam mengelola data pelanggan
pesanan-controller.js	Berperan dalam menangani proses pengiriman produk
keranjang-controller.js	Berperan dalam menangani proses penambahan produk ke dalam keranjang
produk-controller.js	Berperan dalam mengelola data produk
cabang-controller.js	Berperan dalam mengelola data cabang dan data produk yang terkait
auth-controller.js	Berperan dalam menangani proses validasi dan verifikasi pengguna (berupa *login* dan *register*)

D. Hasil Rancangan

Hasil rancangan aplikasi serta implementasi arsitektur MVC disajikan dalam bentuk antarmuka sistem. Ketika pelanggan mengunjungi *website* untuk pertama kali, antarmuka halaman utama akan disajikan. Halaman utama berisikan *top-bar*, daftar cabang restoran populer, dan daftar produk. Halaman utama ditunjukan pada Gambar 5.

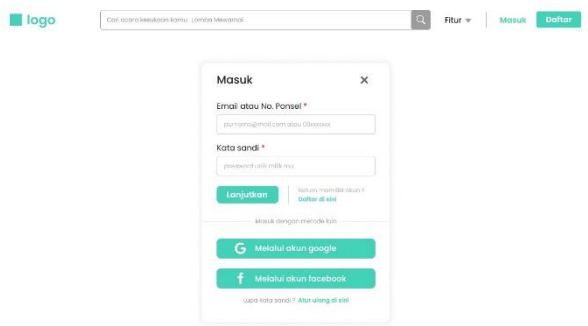


Gambar 5. Halaman Utama

Setiap produk yang ditampilkan pada Gambar 5 mengacu pada cabang yang telah terpilih. Pengguna dapat melihat informasi produk yang ingin mereka cari berdasarkan cabang favorit mereka. Setiap cabang memiliki produknya masing-masing dan produk pada setiap cabang akan berbeda dengan cabang lainnya. Hal ini dilakukan agar pengguna dapat dengan mudah membandingkan penawaran-penawaran yang diberikan masing-masing cabang secara kompetitif. Hal ini juga baik bagi perusahaan karena setiap cabang akan memiliki pelanggan yang setia terhadap cabang tersebut,

E-ISSN : 2623-1700
P-ISSN : 1693-3672

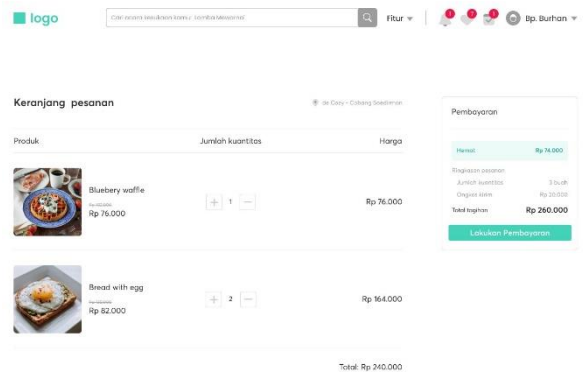
serta memungkinkan cabang perusahaan untuk membangun ikatan dengan pelanggan.



Gambar 6. Halaman *Login*

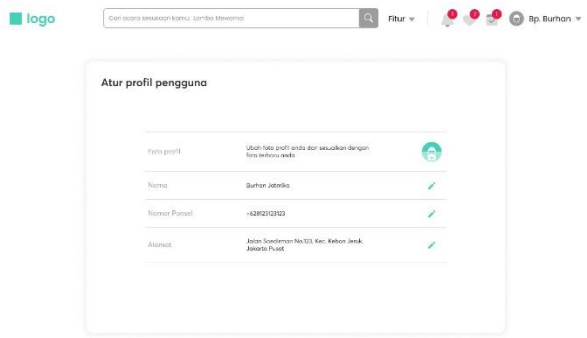
Sebelum proses pemesanan produk dapat dilakukan, pengguna wajib memasukkan data akun milik mereka terlebih dahulu pada halaman *login* yang ditunjukkan oleh Gambar 6, kemudian data tersebut akan diverifikasi dan divalidasi oleh sistem agar sistem dapat memberikan akses yang tepat kepada pelanggan. Apabila pelanggan belum memiliki akun, pelanggan perlu mendaftarkan data diri mereka terlebih dahulu dengan menekan tombol daftar pada bagian pojok kanan atas *top-bar* yang ditunjukan pada Gambar 6.

Setelah pengguna memasukan data mereka serta proses validasi berjalan lancar, maka pengguna dapat mulai melakukan pemesanan produk yang dibutuhkan berdasarkan cabang favorit mereka. Apabila mereka ingin menambahkan produk ke dalam keranjang, mereka cukup menekan tombol bertuliskan keranjang yang tertera pada setiap produk dan produk tersebut secara otomatis akan masuk ke dalam keranjang pesanan pelanggan. Antarmuka keranjang pesanan ditunjukkan pada Gambar 7.



Gambar 7. Halaman Keranjang Pesanan

Pada Gambar 7 menunjukkan data keranjang pesanan berisikan produk beserta kuantitas yang telah ditambahkan oleh pelanggan. Harga produk yang akan dipesan oleh pelanggan dijumlahkan berdasarkan kuantitas tiap-tiap produk yang telah ditambahkan. Pada halaman tersebut memungkinkan pelanggan untuk menambah dan mengurangi produk pesanan mereka. Mereka juga dapat melihat informasi total tagihan pesanan, ongkos kirim, serta total potongan harga pada bagian kanan halaman. Pada bagian tersebut juga terdapat tombol untuk melakukan pembayaran apabila pesanan telah dirasa sesuai.



Gambar 8. Halaman Profil

Pada Gambar 8 menampilkan halaman profil pengguna yang berisikan data diri mereka. Pengguna dapat mengubah data tersebut, mulai dari mengganti foto, nama, nomor ponsel, hingga menambahkan alamat baru.

KESIMPULAN DAN SARAN

Berdasarkan penelitian dan analisis yang telah dilakukan dapat disimpulkan bahwa pengembangan aplikasi portal pelanggan berbasis *web* dengan dibekali *design pattern* berupa arsitektur MVC dapat meminimalisir kompleksitas alur pengembangan aplikasi serta membuat struktur pengembangan menjadi jauh lebih jelas jika dibandingkan dengan arsitektur tradisional. Arsitektur MVC membagi logika pengembangan aplikasi ke dalam perannya masing-masing seperti struktur data aplikasi, kontrol data dan tampilan, serta cara mempresentasikan data. Tidak hanya itu, aplikasi yang telah dibangun menggunakan arsitektur MVC bersifat modular dan reusable, sehingga proses pemeliharaan aplikasi akan jauh lebih mudah.

E-ISSN : 2623-1700
P-ISSN : 1693-3672

Adapun saran yang diberikan oleh peneliti kepada para pengembang aplikasi *web* agar tetap melakukan eksplorasi terkait arsitektur-arsitektur baru, karena arsitektur MVC bukanlah salah satu kerangka yang terbaik dan masih banyak arsitektur lain yang mungkin cocok dengan penelitian yang memiliki skenario lebih kompleks.

DAFTAR PUSTAKA

[1] Gow, D & Hills, B n.d., *Customer Relationship Portals – Managing Customers in an E-Business World*, Mthink, diakses pada 29 Juli 2022, .

[2] Liang, Y. n.d., 'User-Website Interactive Communication Analysis in Web-Based Information System Development', final paper for EEE'06, University of Paisley

[3] Wibisono, G & Susanto, W. E. 2015, 'Perancangan Website Sebagai Media Informasi dan Promosi Batik Khas Kabupaten Kulonprogo', *Evolusi*, vol. 3, no. 2, pp. 1-6, DOI:10.31294/evolusi.v3i2.630

[4] Salehi, F., Abdollahbeigi, B., Langroudi, A. C., & Salehi, F 2012, 'The Impact of Website Information Convenience on E-commerce Success of Companies', *Procedia - Social and Behavioral Sciences*, vol. 57, pp. 381-387, DOI:10.1016/j.sbspro.2012.09.1201

[5] Bradshaw, D & Brash, C 2001, 'Managing customer relationships in the e-business world: how to personalise computer relationships for increased profitability', *International Journal of Retail & Distribution Management*, vol. 29, no. 12, pp. 520-530, DOI: 10.1108/09590550110696969

[6] Thung, P. L., Ng, C. J., Thung, S. J., & Sulaiman, S 2010, 'Improving a web application using design patterns: A case study', *Proceedings 2010 International Symposium on Information Technology - Visual Informatics*, ITSim'10, Universiti Sains Malaysia, Penang, Malaysia.

[7] Vora, P 2009, *Web Application Design Patterns*, Morgan Kaufmann, diakses pada 29 Juli 2022, .

[8] Horchers, J. O. 2001, 'A pattern approach to interaction design', *AI and Society*, DOI: 10.1007/bf01206115

[9] Sunardi, A & Suharjito 2019, 'MVC architecture: a comparative study between laravel framework and slim framework in freelancer project monitoring system web based', *4th International Conference on Computer Science and Computational Intelligence 2019 (ICCSCI)*, Bina Nusantara University, Jakarta, Indonesia, 12-13 September 2019.

[10] GuangChun, L, Lu, W & Hanhong, X 2003, 'A novel web application frame develop by MVC', *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 2, DOI: 10.1145/638750.638779

[11] Veit, M & Herrmann, S 2003, 'Model-View-Controller and object teams: a perfect match of paradigms', *2nd International Conference on Aspect-Oriented Software Development*

[12] Martin, J 1991, *Rapid Application Development*, Macmillan Publishing Co., Inc., USA

[13] Agarwal, R, Prasad, J, Tanniru, M & Lynch, J 2000, 'Risks of rapid application development', *Communication of the ACM*, vol. 43, no. 11es, pp. 1-es, DOI:10.1145/352515.352516

[14] Daud, N. M. N., Bakar, N. A. A. A., & Rusli, H. M 2010, 'Implementing rapid application development (RAD) methodology in developing practical training application system', *Proceedings 2010 International Symposium on Information Technology - System Development and Application and Knowledge Society*, ITSim'10, Kuala Lumpur, Malaysia, 15-17 Juni 2010.