Penerapan MVC dalam Pengembangan Sistem *Point of Sale* (Studi Kasus TPOS PT. Java Signa Intermedia)

Irfan Zainul Abidin
Program Studi Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
16523136@students.uii.ac.id

Hanson Prihantoro Putro
Program Studi Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
hanson@uii.ac.id

Abstrak—TPOS (Toko *Point of Sale*) merupakan sebuah aplikasi untuk mengelola penjualan produk pada tiap titik toko di berbagai wilayah di Indonesia melalui situs web. Tujuan dikembangkan sistem tersebut untuk mempermudah manajemen barang, transaksi penjualan, dan pembuatan laporan penjualan. Ketidaksamaan aturan penulisan kode program antar *programmer* menyebabkan proses integrasi kode program sulit dilakukan, terutama jika suatu sistem yang dikembangkan memiliki tingkat kompleksitas yang cukup tinggi. Untuk mengatasi masalah ini diperlukan suatu konsep yang baku beserta pemetaan bagian bagian kelas pada program yang jelas agar proses integrasi program lebih mudah dilakukan. Solusi dari masalah ini adalah penerapan konsep MVC (*Model View Controller*). Konsep MVC membagi program pada sistem TPOS menjadi tiga kelas utama yaitu *model, view,* dan *controller*. *Model* bertugas untuk menyediakan, memanipulasi dan mengorganisasikan data dari basis data TPOS sesuai dengan perintah dari *controller*. *View* bertugas untuk menampilkan informasi kepada pengguna sesuai arahan dari *controller. Controller* berfungsi untuk mengatur tugas yang harus dilakukan *model* dan *view*. Pembagian kelas ini berguna untuk mempermudah proses integrasi kode program. Terdapat 14 kelas *model,* 18 kelas *controller,* dan 14 *folder view* inti pada sistem TPOS. Tiap *folder view inti* memiliki beberapa *file* yang berisikan halaman *view*. Makalah ini bertujuan untuk menjelaskan penerapan konsep MVC pada sistem TPOS. Hasil dari penerapan konsep MVC pada sistem TPOS berupa dokumentasi kode program yang terstruktur berdasar kelas *model, view, dan controller.*

Kata Kunci— ***Point of Sale, MVC, e-commerce.***

I. PENDAHULUAN

TPOS adalah aplikasi web untuk mengelola penjualan produk pada tiap titik toko di berbagai wilayah di Indonesia. Dalam bahasa Inggris konsep ini disebut Sistem *point of sale*. Secara teori POS (*point of sale)* merupakan tempat di mana pelanggan melakukan pembayaran untuk barang atau jasa, dan di mana pajak penjualan dapat dibayarkan. Hal itu bisa dilakukan di tempat toko fisik, di mana terminal POS dan sistem digunakan untuk memproses pembayaran kartu, atau titik penjualan virtual, seperti komputer atau perangkat elektronik seluler [1]. TPOS diharapkan dapat membantu transaksi penjualan dan perhitungan stok barang. Para admin toko dapat melakukan penjualan dan perhitungan stok barang melalui TPOS dari gawai mereka. Manfaat lain yang diharapkan adalah kemudahan admin untuk mengelola toko di berbagai titik wilayah di Indonesia melalui satu aplikasi, mulai dari data penjualan hingga stok produk.

TPOS merupakan suatu sistem yang cukup komplek dilihat dari banyaknya kebutuhan sistem. Proses integrasi kode program menjadi hambatan saat pengembangan sedang berlangsung. Hal ini karena berbagai macam penyebab, beberapa diantaranya adalah perbedaan aturan penulisan kode program setiap *programmer* berbeda dan pembagian kelas yang kurang jelas, seperti penggabungan beberapa kelas yang saharusnya tidak dilakukan. Masalah ini akan berdampak pada sulitnya manajemen program saat dan setelah program dikembangkan. Untuk mengatasi masalah ini, diperlukan suatu aturan penulisan program yang baku beserta pemetaan kelas yang jelas.

TPOS dikembangkan dengan konsep MVC dan menggunakan *framework* Codeigniter. Bahasa pemograman *back-end* yang digunakan adalah PHP dan MySQL sebagai basis data. Penggunaan konsep MVC ditujukan untuk mempermudah *developer* melakukan pengembangan secara modular. Konsep MVC memisahkan bagian *view, controller* dan *model* sehingga bagian aplikasi dapat dibedakan dengan jelas. Proses manajemen sistem TPOS menjadi lebih mudah karena dokumentasi kode program yang jelas dan terstruktur.

Beberapa makalah telah membahas penerapan konsep mvc untuk mengembangkan sitem web, contohnya adalah Pembangunan Sistem Informasi *Point of Sales* Terintegrasi Dalam Lingkup Rumah Makan Beserta Cabangnya (Studi Kasus: RM. Pecel Pincuk Bu Tinuk) [2]. Makalah tersebut menjelaskan proses pengembangan sistem POS untuk digitalisasi proses konvensional yang sudah ada. Konsep MVC digunakan setelah semua kebutuhan sistem diketahui. Setelah melalui pengujian *whitebox* dan *blackbox*, web dinyatakan telah memenuhi semua kebutuhan sistem. Salah satu kesimpulan dari makalah tersebut adalah penggunaan *framework* Codeigniter dengan konsep MVC memudahkan proses implementasi kode program.

Makalah lain membahas tentang penerapan arsitektur *model view controller* (mvc) dalam rancang bangun sistem kuis *online* adaptif [3]. Penggunaan konsep MVC digunakan untuk memudahkan pengembangan sistem. Konsep MVC digunakan untuk meminimalisir perubahan bagian kode program lain jika satu bagian kode program diubah. Hasil dari rancang bangun pada makalah ini adalah sebuah sistem penilaian siswa berdasarkan kemampuan, pengetahuan dan pilihan dari masing-masing siswa secara online. Kesimpulan dari makalah ini adalah dengan penggunaan konsep MVC, kompleksitas dari kode dalam perangkat lunak dapat

dikurangi secara signifikan, dengan demikian, meningkatkan fleksibilitas dan modularitas sistem perangkat lunak.

Fokus pembahasan karya ilmiah ini adalah penerapan MVC dalam pengembangan sistem *point of sale* pada aplikasi TPOS. Manajemen konsep MVC bergantung pada skala kebutuhan program. Jika terdapat penambahan fitur maka fungsi program yang berkaitan dapat ditambahakan pada kelas yang berkaitan. Hal yang sama akan dilakukan jika terdapat pengurangan atau penghapusan fitur, bedanya berupa penghapusan atau pengeditan fungsi program pada kelas terkait. Evaluasi dilakukan saat dan setelah sistem dikembangkan. Jika tidak terjadi ketidaksesuaian *output*, maka perlu dilakukan pengecekan variabel atau parameter pada masing masing fungsi program yang berkaitan. Pada bab selanjutnya akan dibahas keterkaitan antara *model, view,* dan *controller* pada aplikasi TPOS. Setelah dikembangkan, diharapkan aplikasi ini dapat membantu *client* (pedagang) untuk mengelola toko.

II. KAJIAN PUSTAKA

A. MVC (Model View Controller)

MVC adalah arsitektur pengembangan aplikasi berbasis web, membagi tiga bagian yang saling terhubung. Bagian tersebut berupa: *model, view, dan controller.* Hal ini dilakukan untuk memisahkan representasi informasi internal dari cara informasi disajikan dan diterima oleh pengguna [4]. *Model* bertugas untuk menyediakan, memanipulasi dan mengorganisasikan data dari basis data sesuai dengan perintah dari *controller*. *View* bertugas untuk menampilkan informasi kepada pengguna sesuai arahan dari *controller. Controller* berfungsi untuk mengatur tugas yang harus dilakukan *model* dan *view* mana yang harus ditampilkan berdasarkan permintaan dari user.

Beberapa sistem *point of sale* telah dikembangkan menggunakan konsep MVC menggunakan *framework* Laravel, salah satu contohnya adalah Aplikasi *Point of Sale* menggunakan *framework* Laravel [5]. Sistem tersebut dikembangkan menggunakan konsep MVC untuk sebuah toko aksesoris gawai. Sistem ini dikembangkan untuk digitalisasi proses yang sudah ada, yaitu manajemen transaksi penjualan dan perhitungan stok aksesoris. Hal ini karena proses sebelumnya dilakukan secara manual sementara toko terus berkembang dan transaksi yang dilakukan semakin banyak.

B. Sistem POS (Point of Sale)

Sistem POS (*Point of Sale*) merupakan sebuah sistem untuk menyinkronkan dan mengintegrasikan data reservasi, data pesanan, data e-commerce, data kartu hadiah, atau data poin loyalitas yang berada di perangkat POS dengan perangkat situs web pedagang dan menyinkronkan data yang berada di basis data situs web ke perangkat POS terkait [6]. Sistem POS digunakan untuk mengintegrasikan sistem reservasi, pemesanan, dan sistem *merchant e-commerce* lainya yang disediakan melalui situs web. Sitem POS mencakup perangkat POS, *server* web POS, Lapisan basis data POS, aplikasi situs web POS.

Adapun publikasi ilmiah yang berjudul *Web Based Point of Sale System* atau disingkat WPOS berupa sistem berbasis web yang memungkinkan manajemen laporan toko secara jarak jauh, dan memungkinkan pelanggan melakukan penjadwalan atau penjadwalan ulang waktu pengiriman [7]. Server menyediakan semua data dan informasi penting yang dibutuhkan melalui web browser pelanggan. dalam sistem ini, server antar toko dapat berkomunikasi satu sama lain dengan *mainframe* kantor pusat. WPOS dapat diimplementasikan sebagai rangkaian terintegrasi untuk kolaborasi antar lokasi toko.

III. METODOLOGI

Pengembangan sistem TPOS yang dirancang dengan konsep MVC ini menggunakan metode *waterfall*. Berikut merupakan penjelasan secara rinci tahapan pengembangan aplikasi menggunakan metode *waterfall.*

- *Requirements analysis:* Tahap ini merupakan proses menganalisis kebutuhan dan batasan sistem TPOS yang akan dikembangkan. Informasi didapat dari diskusi dengan *client*. Setelah informasi terkumpul, dilakukan proses analisis kebutuhan sistem. Analisis kebutuhan sistem TPOS dikumpulkan pada tahap ini. Tabel analisis kebutuhan sistem (*use case*) TPOS dapat dilihat pada Tabel I. Berkaitan dengan konsep MVC, pemetaan kelas *model, view, dan controller* akan dilakukan setelah semua kebutuhan sistem TPOS diketahui.
- *System Design:* Tahap selanjutnya setelah kebutuhan sistem diketahui adalah mendesain sistem. Proses ini berfokus pada rancangan antarmuka, fungsi program, prosedur program, algoritma program, dan arsitektur yang akan digunakan pada sistem TPOS. Pada tahap ini, kebutuhan sistem akan diterjemahkan menjadi fungsi atau prosedur program. Pemetaan konsep MVC mulai dilakukan di tahap ini, setiap kelas *model, view,* dan *controller* memiliki fungsi atau prosedur program yang sesuai dengan kelasnya masing-masing.
- *Implementation:* Langkah selanjutnya yaitu proses mengubah rancangan yang telah ditentukan menjadi kode program. Pada proses ini, dilakukan penulisan semua kode program yang telah dirancang termasuk fungsi atau prosedur program pada masing-masing kelas *model, view,* dan *controller.*
- *Integration and Testing:* Pada tahap ini modul-modul digabungkan, dan dilakukan pengetesan apakah fungsi program sudah berjalan sesuai parameter yang ditentukan. Namun terkadang *unit testing* juga perlu dilakukan untuk memeriksa apakah fungsi atau prosedur program dapat berjalan sesuai parameter.
- *Operation and Maintenance:* Tahap ini merupakan proses pemeliharaan program berupa perbaikan *bug* atau peningkatan sistem TPOS. Peningkatan sistem dapat menjadi kebutuhan baru.

Table I. TPOS *Use Case*

Id Kasus Uji	**Deskripsi *Use Case***
TUC-1	Sistem harus dapat melakukan registrasi via nomor telepon dan *email*.
TUC-2	Sistem harus dapat login menggunakan no telepon dan *email.*
TUC-3	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus toko.
TUC-4	Sistem harus dapat *login* berdasar hak akses yang dimiliki pengguna.
TUC-5	Sistem harus dapat *login* berdasar hak akses yang dimiliki pengguna.

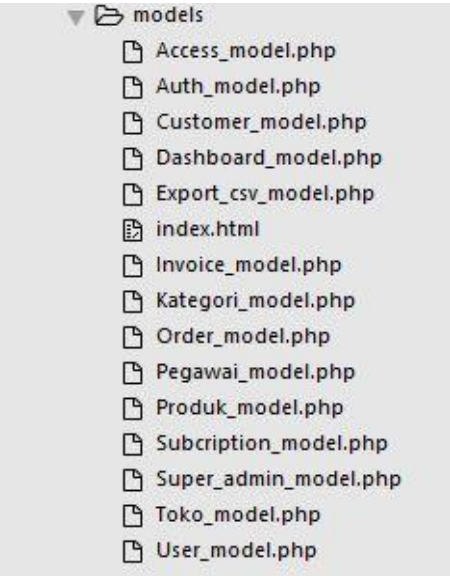
Id Kasus Uji	Deskripsi *Use Case*
TUC-6	Sistem harus dapat menampilkan jumlah pelanggan baru dalam seminggu, total pelanggan, total pegawai, grafik penjualan produk tertinggi, grafik penjualan kategori tertinggi, dan pengingat stok di halaman *dashboard*.
TUC-7	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus data pelanggan.
TUC-8	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus data produk.
TUC-9	Sistem harus dapat membagikan produk via *whatsapp*.
TUC-10	Sistem harus dapat menyimpan produk sebagai *draft.*
TUC-11	Sistem harus dapat mencari produk berdasar kode atau nama produk.
TUC-12	Sistem harus dapat memindahkan produk ke toko lain.
TUC-13	Sistem harus dapat menambah dan mengedit kategori dan sub kategori produk.
TUC-14	Sistem harus dapat melakukan proses pemesanan produk.
TUC-15	Sistem harus dapat menyimpan data pesanan.
TUC-16	Sistem harus dapat menampilkan, mencetak, memfilter berdasarkan tanggal, dan mencari data pesanan.
TUC-17	Sistem harus dapat mengubah status pesanan dan mengirimkan informasi status ke pelanggan melalui *email*.
TUC-18	Sistem harus dapat mencetak dokumen *invoice* dan mengirimkan dokumen *invoice* melalui *whatsapp* dan *email*.
TUC-19	Sistem harus dapat mengatur pesanan berupa metode pengiriman dan metode pembayaran produk.
TUC-20	Sistem harus dapat mengembalikan jumlah stok produk jika pesanan dikembalikan.
TUC-21	Sistem harus dapat menampilkan info grafis penjualan, pelanggan, dan lokasi.
TUC-22	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus data pegawai toko.
TUC-23	Sistem harus dapat memfilter pegawai toko berdasar hak akses.
TUC-24	Sistem harus dapat mengedit profil pengguna.
TUC-25	Sistem harus dapat mengubah menu akses.

IV. HASIL

Setelah diketahui kebutuhan sistem (*use case)* TPOS, dapat dilakukan pemetaan kelas *model, view,* dan *controller*. Berikut murapakan kelas MVC pada TPOS yang telah dipetakan berdasarkan analisis kebutuhan sistem pada tahap *requirements analysis*. Poin selanjutnya akan memaparkan beberapa tampilan beserta keterkaitan antara *model, view,* dan *controller* dari sistem TPOS.

A. MVC TPOS

TPOS dikembangkan menggunakan konsep MVC untuk memecah program menjadi tiga bagian utama. Pemetaan kelas MVC didapat dari tahap *system design*. Setelah dipetakan terdapat 14 kelas *model* yaitu Access_model, Auth_model, Customer_model, Dashboard_model, Export_csv_model, Invoice_model, Kategori_model, Order_model, Pegawai_model, Produk_model, Subcription_model, Super_admin_model, Toko_model, dan User_model. 14 kelas model tersebut dapat dilihat pada Gambar 1.

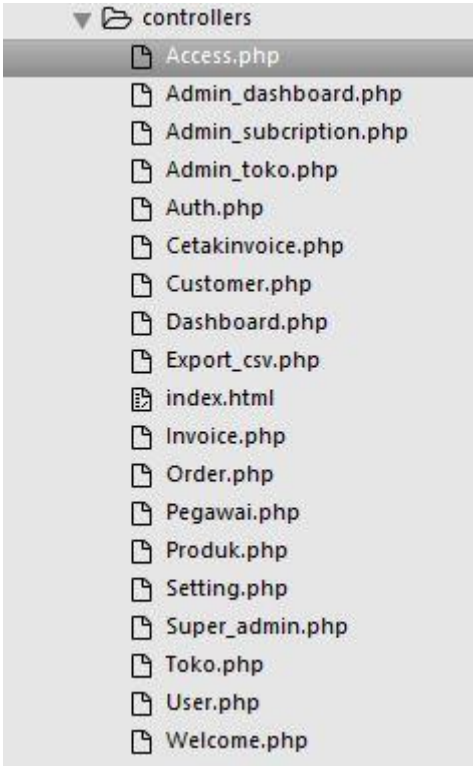


Gambar 1. Kelas *Model*

Terdapat 14 *folder view* inti, dimana setiap *folder* memiliki beberapa *file* tampilan antarmuka. 14 *folder* tersebut adalah admin, auth, customer, dashboard, errors, invoice, kategori, order, partial_sidebar, pegawai, produk, setting, toko, dan user. Gambar 2 menunjukan 14 *folder view* inti TPOS.

TPOS memiliki 18 kelas *controller*. 18 kelas *controller* tersebut adalah Access, Admin_dashboard, Admin_subcription, Admin_toko, Auth, Cetakinvoice, Customer, Dashboard, Export_csv, Invoice, Order, Pegawai, Produk, Setting, Super_admin, Toko, User, dan Welcome. Gambar 3 menunjukan 18 kelas *controller* TPOS.

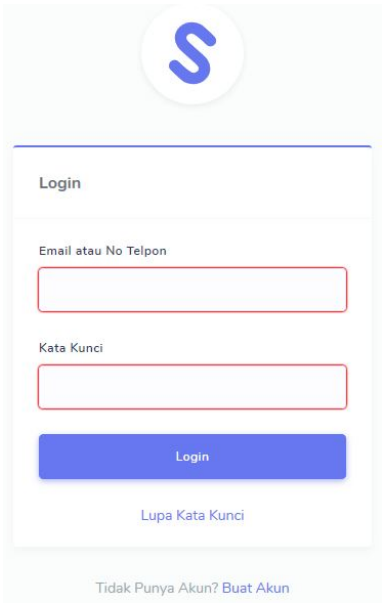
Gambar 2. *Folder View* inti



Gambar 3. Kelas *Controller*

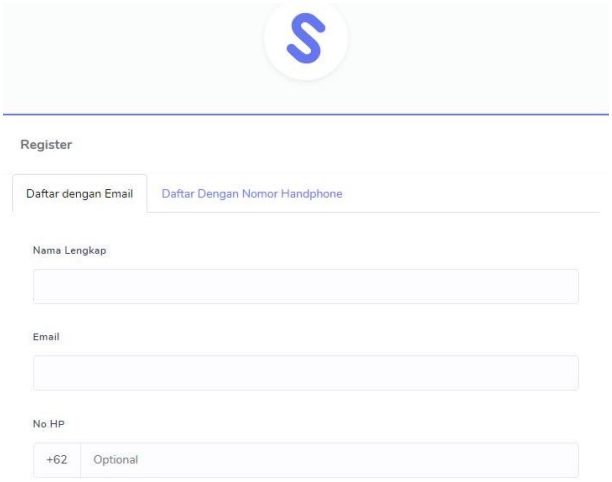
B. Bagian Login dan Registrasi

Folder view auth memuat halaman *login*, halaman registrasi via *email*, dan halaman registrasi via nomor telepon. Pada Halaman *login* terdapat form pengisian *email* atau nomor telepon dan password. Setelah berhasil login, pengguna akan diarahkan ke halaman *dashboard*. Gambar 4 menunjukan halaman *login*.

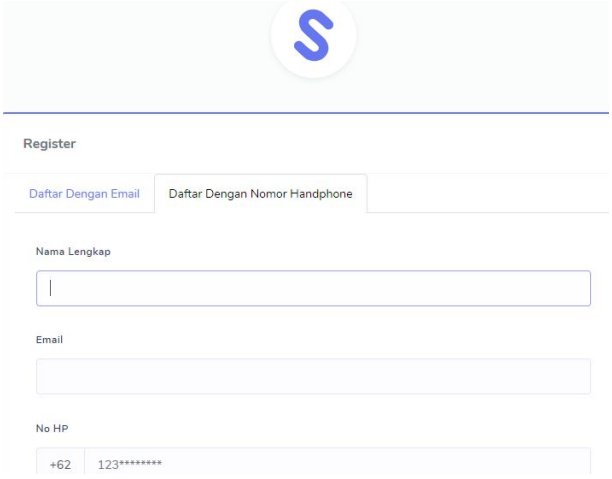


Gambar 4. Halaman *Login*

Terdapat 2 halaman registrasi yaitu halaman registrasi via *email* dan nomor telepon. Gambar 5 dan Gambar 6 merupakan tampilan halamn registrasi via *email* dan nomor telepon.

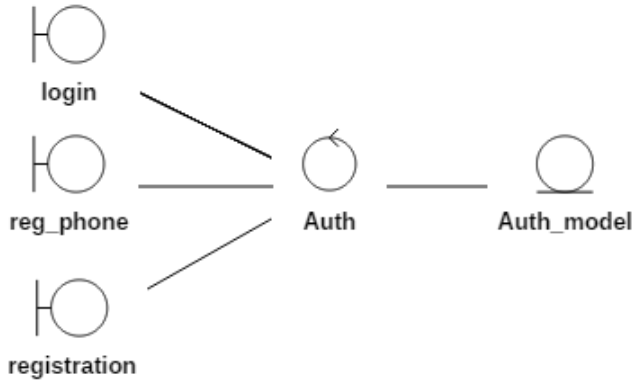


Gambar 5. Halaman Registrasi via *Email*



Gambar 6. Halaman Registrasi via Nomor Telepon

Halaman *login*, halaman registrasi via *email* dan halaman registrasi via nomor telepon diatur oleh *controller* Auth. *Controller* Auth menghubungkan ketiga halaman kelas *view* dengan *model* Auth_model. Skema MVC untuk kasus ini dapat dilihat pada Gambar 7.



Gambar 7. Skema MVC Halaman *Login* dan Registrasi

Terdapat 12 fungsi program pada *controller* Auth dan 4 fungsi program pada *model* Auth_model, setiap fungsi program pada *controller* Auth dapat memanggil satu atau lebih fungsi program di *model* Auth_model. Berikut merupakan salah satu contohnya. *View* login memiliki *form* login. Elemen *form* HTML memiliki atribut action, atribut action akan memanggil fungsi index di kelas *controller* Auth (lihat Gambar 8).

```
<form method="POST" action="<?php echo site_url('auth/') ?>" class="needs-validation">
```

Gambar 8. Atribut action Memanggil Fungsi index

Fungsi index akan memanggil fungsi _login di *controller* Auth jika validasi *form* benar. Fungsi _login membutuhkan data *email* dari *model* Auth_model untuk dilakukan pengecekan di langkah selanjutnya. Data *email* disimpan di variabel user dan user_phone (lihat Gambar 9).

```
private function _login()
{
    $role_pegawai = $this->db->get('hak_akses_pegawai')->result();
    $role_user = $this->db->get('role')->result();
    $email = trim($this->input->post('email'));
    if(substr($email, 0, 1)=='0'){
        // var_dump(substr($email, 0, 1));die;
        $email = trim('+62'.substr($email,1,15));
    }
    $password = $this->input->post('password');
    $user = $this->auth->getUserLogin(['user.email' => $email])->row_array();
    $user_phone = $this->auth->getUserLogin(['user.phone' => $email])->row_array();
```

Gambar 9. Variabel user dan Variabel user_phone

Variabel user dan variabel user_phone memanggil fungsi getUserLogin yang ada di *model* Auth_model disertai parameter variabel *email*. Gambar 10 menunjukan fungsi getUserLogin di *model* Auth_model.

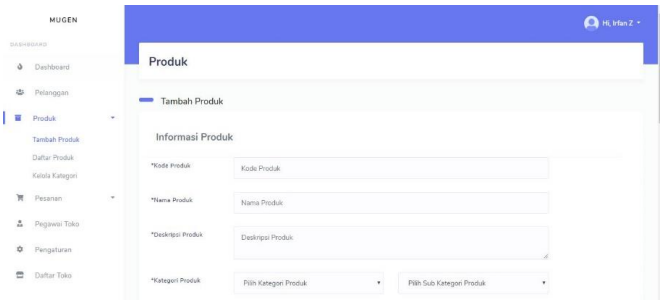
```
defined('BASEPATH') or exit('No direct script access allowed');
class Auth_model extends CI_Model
{

    public function getUserLogin($where)
    {
        $this->db->select('*');
        $this->db->from('user');
        $this->db->join('role', 'user.role_id=role.id_role');
        $this->db->where($where);
        return $this->db->get();
    }
```

Gambar 10. Fungsi getUserLogin di *Model* Auth_model

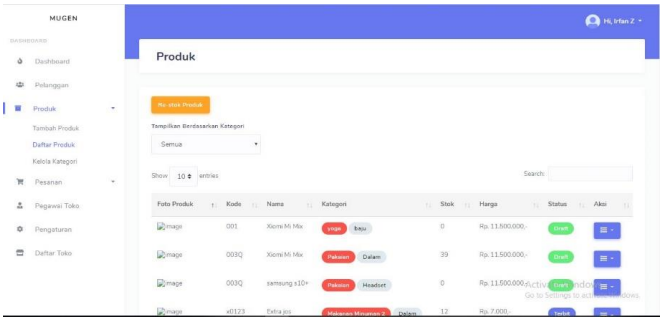
C. Bagian Produk

Folder view produk memuat 3 halaman tampilan utama yaitu tambah produk, daftar produk, dan kelola kategori. Halaman tambah produk berisi form yang berkaitan dengan atribut produk seperti nama, warna, ukuran, deskripsi, dan kategori. Halaman Daftar Produk menampilkan lis produk dari basis data. Pada halaman kelola kategori terdapat daftar kategori, sub kategori, fitur tambah, dan edit. Gambar 8, Gambar 9, Gambar 10 secara berurutan menunjukan halaman tambah produk, daftar produk, dan kelola kategori.

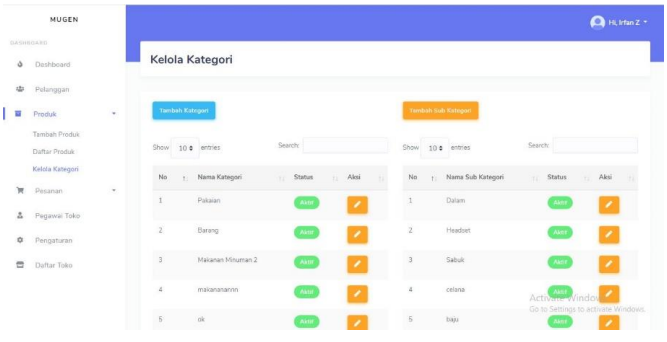
Halaman tambah produk, daftar produk, dan kelola kategori dikelola oleh *controller* Produk. Setiap tombol aksi di ketiga halaman tersebut mengacu pada fungsi di *controller* Produk. *Controller* Produk berhubungan dengan *model* Produk_model, Dashboard_model, dan Kategori_model. Terdapat sekitar 50 fungsi program pada *controller* Produk, 21 fungsi program di *model* Produk_model, 5 fungsi program di *model* Dashboard_model, dan 8 fungsi program pada *model* Kategori_model. Skema MVC untuk bagian produk ditunjukan oleh Gambar 11.



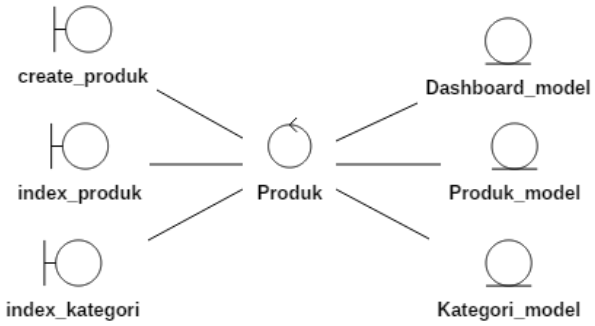
Gambar 11. Halaman Tambah Produk



Gambar 12. Halaman Daftar Produk



Gambar 13. Halaman Kelola Kategori



Gambar 14. Skema MVC Bagian Produk

Merujuk Gambar 11, berikut merupakan penjelasan keterkaitan *view* dan *controller* saat proses menambah produk. Halaman create_produk memiliki *form* dengan atribut action yang memanggil fungsi insert_produk di *controller* Produk (lihat Gambar 15).

```
<form action="<?php echo site_url('produk/insert_produk') ?>" method="post" enctype="multipart/form-data">
    <div class="section-body">
        <div class="section-title">Tambah Produk</div>
        <div class="section-lead"></div>
```

Gambar 15. Atribut action Memanggil Fungsi insert_produk

Fungsi insert_produk di *controller* Produk akan menyimpan nilai *input* sebagai *array* dari *form* create_produk. *Array* disimpan di variabel data kemudian variabel data akan memanggil fungsi insert untuk memasukan data ke basis data (lihat Gambar 16).

```
$data = [
    'kode_produk'=>$this->input->post('kode_produk'),
    'nama_produk'=>$this->input->post('nama_produk'),
    'harga_beli'=>$this->input->post('harga_beli'),
    'link'=>$this->input->post('link'),
    'harga_jual'=>'0',
    'ukuran' => $this->input->post('ukuran'),
    'berat_produk'=>'0',
    'stok_produk'=>$this->input->post('stok_produk'),
    'label_produk'=>'-',
    'deskripsi_produk'=>$this->input->post('deskripsi_produk'),
    'terbitkan_diweb'=>$terbitkan_diweb,
    'id_toko'=>$this->session->userdata('login_toko_id_pos'),
    'kategori_id'=>$this->input->post('kategori_produk'),
    'sub_kategori_id'=>$this->input->post('subkategori'),
    'bahan'=>$this->input->post('bahan'),
    'warna'=>$this->input->post('warna')
];

for ($i=1; $i <=5 ; $i++) {
    if (!empty($_FILES['image' . $i]['name'])) {
        $upload = $this->do_upload($i);
        $data['image' . $i] = $upload;

    } else {
        $data['image' . $i] = 'default.png';
    }
}
$this->db->insert('product',$data);
```

Gambar 16. Variabel data dan fungsi insert

Contoh salah satu kemudahan proses integrasi kode program adalah ketika *programmer* satu mengerjakan halaman tampilan create_produk, dan *programmer* dua mengerjakan seperangkat fungsi program di kelas *controller* Produk pada perangkat yang berbeda. Kedua *file* yang berisikan kode program tersebut dapat diintegrasikan dengan mudah dengan atau tanpa salah satu *tools versioning*. Hal yang berkebalikan akan dijumpai jika kedua *programmer* tersebut mengerjakan kedua tugas tersebut dalam satu *file* yang sama pada perangkat yang berbeda. Hal ini membuktikan bahwa pemetaan kelas MVC dapat mempermudah proses integrasi kode program TPOS.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Pemetaan kelas MVC sistem TPOS dilakukan pada tahap *System Design* setelah diketahui kebutuhan sistem (*use case*). Terdapat 14 kelas *model*, 18 kelas *controller*, dan 14 *folder view* inti. Tiap *folder view inti* memiliki beberapa *file* yang berisikan halaman *view*. Masing-masing halaman *view* dikelola oleh *controller*. *Controller* menghubungkan halaman *view* dengan kelas *model*. Hal ini sesuai dengan pemaparan pada bab iv. Pemetaan kelas MVC mempermudah proses integrasi kode program TPOS.

B. Saran

Penerapan MVC sistem *Point of Sale* dalam makalah ini masih dapat dikembangkan guna meningkatkan efisiensi proses integrasi kode program pada sistem. Beberapa saran pengembangan dari penerapan konsep MVC sebagai berikut :

- Penerapan MVC pada sistem *point of sale* yang memiliki data yang sangat besar sebagai upaya proses integrasi program dan pengganti metode pengembangan sistem konvensional.
- Penerapan MVC pada sistem yang memiliki kompleksitas tinggi sebagai sarana pemetaan kelas.
- Penerapan MVC pada sistem *e-commerce* sebagai sarana pengembangan sistem secara modular.

VI. REFERENSI

[1] C. A. Sukandar, "Warta Ekonomi," 23 April 2019. [Online]. Available: https://www.wartaekonomi.co.id/read224883/apa-itu-point-of-sale.html. [Accessed 18 Desember 2019].

[2] A. S. Sani, F. Pradana and D. S. Rusdianto, "Pembangunan Sistem Informasi Point Of Sales Terintegrasi Dalam Lingkup Rumah Makan Beserta Cabangnya (Studi Kasus: RM. Pecel Pincuk Bu Tinuk)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, pp. 3249-3257, 2018.

[3] A. Hidayat and B. Surarso, "Penerapan Arsitektur Model View Controller (MVC) dalam Rancangan Bangun Sistem Kuis Online Adaptif," in *Seminar Nasional Teknologi Informasi dan Komunikasi 2012 (SENTIKA 2012)* , Yogyakarta, 2012.

[4] T. Reenskaug and J. Coplien, "The DCI Architecture: A New Vision of Object-Oriented Programming," *aritma developer*, 20 Maret 2009.

[5] T. B. Tahir, M. Rais and M. Apriyadi, "Aplikasi Point of Sales Menggunakan Framework Laravel," *Jurnal Informatika dan Ilmu Komputer (JIKO)*, vol. 2, pp. 55-59, 2019.

[6] S. I. O. and V. (. , "Web integrated point-of-sale system". United States Patent US 2012/0296679 A1, 22 November 2012.

[7] M. Manno, "Web Based Point of Sale System". United States Patent US 2004/0181454 A1, 16 September 2004.