SKRIPSI

PENERAPAN MVC DALAM PENGEMBANGAN SISTEM POINT OF SALES (STUDI KASUS TPOS PT. JAVASIGNA INTERMEDIA)

Disusun Oleh:

N a m a : Irfan Zainul Abidin

NIM : 16523136

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENERAPAN MVC DALAM PENGEMBANGAN SISTEM POINT OF SALES (STUDI KASUS TPOS PT. JAVASIGNA INTERMEDIA)

TUGAS AKHIR JALUR MAGANG

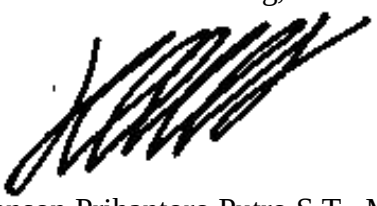
Disusun Oleh:

N a m a : Irfan Zainul Abidin

NIM : 16523136

Yogyakarta, 17 Desember 2020

Pembimbing,



(Hanson Prihantoro Putro S.T., M.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENERAPAN MVC DALAM PENGEMBANGAN SISTEM POINT OF SALES (STUDI KASUS TPOS PT. JAVASIGNA INTERMEDIA)

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 11 Januari 2021

Tim Penguji

Ketua Penguji

Hanson Prihantoro Putro S.T., M.T. _____

Anggota 1

Andhika Giri Persada, S.Kom., M.Eng. _____

Anggota 2

Kholid Haryono, S.T., M.Kom. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Irfan Zainul Abidin
NIM	:	16523136

Tugas akhir dengan judul:

PENERAPAN MVC DALAM PENGEMBANGAN SISTEM POINT OF SALES (STUDI KASUS TPOS PT. JAVASIGNA INTERMEDIA)

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 17 Desember 2020

(Irfan Zainul Abidin)

HALAMAN PERSEMBAHAN

Skripsi ini saya persembahkan untuk kedua orang tua saya yang sungguh saya cintai (Bapak Muhamad Turi dan Ibu Ambar Sari Yanu Astuti).

HALAMAN MOTO

“Menuntut ilmu itu wajib atas setiap Muslim.”
(HR. Ibnu Majah no. 224)

"Tanpa tindakan, pengetahuan tidak ada gunanya dan pengetahuan tanpa tindakan itu sia-sia."
(Abu Bakar)

"Aku tidak pernah sekalipun menyesali diamku. Tetapi aku berkali-kali menyesali bicaraku."
(Umar bin Khattab)

"Pengetahuan lebih baik daripada kekayaan, pengetahuan akan melindungimu, sedangkan kekayaan harus kamu lindungi."
(Utsman bin affan)

“Aku sudah pernah merasakan semua kepahitan dalam hidup dan yang paling pahit ialah berharap kepada manusia.”
(Ali bin Abi Thalib)

“Bersyukur itu perlu terus latihan. Jangan hanya rasa syukur saat memiliki sesuatu.”
(Bahaudin Nur Salim)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah penulis haturkan kepada Allah ﷻ, yang telah melimpahkan rahmat dan taufik serta hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "PENERAPAN MVC DALAM PENGEMBANGAN SISTEM POINT OF SALES (STUDI KASUS TPOS PT. JAVASIGNA INTERMEDIA)" ini. Tidak lupa shalawat dan salam juga penulis haturkan kepada Rasul ﷺ dan Para Sahabat.

Maksud dan tujuan dari penyusunan skripsi ini untuk memenuhi salah satu persyaratan kelulusan di Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta.

Penulis menyadari penyusunan laporan ini tidak lepas dari bantuan, bimbingan, dan tuntunan dari berbagai pihak, maka pada kesempatan ini penulis menyampaikan ucapan terima kasih yang setulusnya kepada:

1. Bapak Muhamad Turi dan Ibu Ambar Sari Yanu Astuti, selaku kedua orang tua penulis yang selalu memberikan dukungan moral maupun materi. Tanpa doa beliau berdua, penulis tidak akan mungkin menyelesaikan skripsi ini. Tujuan penulisan laporan atau pelaksanaan magang sebagai tugas akhir.
2. Hanson Prihantoro Putro, S.T., M.T. sebagai dosen pembimbing yang telah bersedia untuk meluangkan waktu untuk membimbing, memeriksa, serta memberikan petunjuk-petunjuk dan saran dalam penyusunan laporan ini.
3. Andhika Giri Persada, S.Kom., M.Eng. sebagai dosen penguji yang telah bersedia untuk meluangkan waktu untuk menguji, memeriksa, serta memberikan petunjuk-petunjuk dan saran dalam penyusunan laporan ini.
4. Kholid Haryono, S.T., M.Kom. sebagai dosen penguji yang telah bersedia untuk meluangkan waktu untuk menguji, memeriksa, serta memberikan petunjuk-petunjuk dan saran dalam penyusunan laporan ini.
5. Semua dosen Program Studi Informatika UII yang telah memberikan ilmu bermanfaat dan arahan positif selama menjalani studi ini.
6. Bapak Andwi Valentine sebagai *Chief Executive Officer* di PT. Javasigna Intermedia yang telah mengizinkan penulis melakukan kegiatan magang di perusahan beliau.
7. Mas Rangga sebagai pembimbing lapangan, yang telah bersedia untuk meluangkan waktu untuk membimbing, memeriksa, serta memberikan petunjuk-petunjuk selama pelaksanaan magang.

8. Semua rekan Program Studi Informatika UII angkatan 2016, yang saling membantu dan menyemangati satu sama lain.
9. Seluruh keluarga, sahabat, rekan, dan semua pihak yang tidak dapat penulis sebutkan satu persatu, terimakasih atas dukungan dan doa yang selalu dibacakan.

Akhir kata, semoga Allah ﷻ senantiasa melimpahkan karunia-Nya dan membalas segala amal budi serta kebaikan pihak-pihak yang telah membantu penulis dalam penyusunan laporan ini dan semoga tulisan ini dapat memberikan manfaat bagi pihak-pihak yang membutuhkan.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 17 Desember 2020

(Irfan Zainul Abidin)

SARI

TPOS (Toko *Point of Sales*) 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. Laporan ini bertujuan untuk menjelaskan penerapan konsep MVC pada sistem TPOS. Terdapat 14 kelas *model*, 17 kelas *controller*, dan 14 *folder view* inti pada sistem TPOS. Tiap *folder view* inti memiliki beberapa *file* yang berisikan halaman *view*. Hasil dari penerapan konsep MVC pada sistem TPOS berupa dokumentasi kode program yang terstruktur berdasar kelas *model*, *view*, dan *controller*.

Kata kunci: *Point of Sales*, MVC, *e-commerce*.

GLOSARIUM

Array	suatu tipe data terstruktur yang dapat menyimpan banyak data dengan suatu nama yang sama dan menempati tempat di memori yang berurutan serta bertipe data sama.
Backlog	antrian fleksibel pekerjaan-pekerjaan, yang diminta pihak bisnis untuk dikerjakan pihak eksekutor di masa depan.
Bug	suatu cacat desain pada perangkat keras atau perangkat lunak yang mengakibatkan terjadinya galat pada peralatan atau program sehingga tidak berfungsi sebagaimana mestinya.
Commit	sebuah titik pemeriksaan yang memberi tahu *git* untuk melacak semua perubahan yang telah terjadi dan membandingkannya dengan *commit* terakhir.
E-commerce	penyebaran, pembelian, penjualan, pemasaran barang dan jasa melalui sistem elektronik seperti internet, televisi, dan jaringan komputer lainnya.
Framework	struktur dasar yang melandasi sebuah sistem.
Git	perangkat lunak pengendali versi atau proyek manajemen kode perangkat lunak.
Library	kumpulan sumber daya non-*volatile* yang digunakan oleh program komputer, seringkali dimanfaatkan untuk pengembangan perangkat lunak.
Mainframe	komputer yang digunakan untuk memproses data dan aplikasi yang besar.
Pull	perintah yang digunakan untuk mengambil dan mengunduh konten dari penyimpanan tujuan dan segera memperbarui penyimpanan lokal agar sesuai dengan konten terbaru.
Push	perintah yang digunakan untuk mengunggah konten lokal ke penyimpanan tujuan.
Repository	tempat untuk menyimpan proyek yang berisikan kode program dan aset lain.
Resources	segala hal yang dapat diidentifikasi, baik digital, fisik, atau abstrak.
Session	cara yang digunakan untuk menyimpan informasi pada komputer server untuk digunakan pada beberapa halaman termasuk halaman itu sendiri.
Waterfall	metode pengembangan perangkat lunak.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

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 sales*. Secara teori POS (*Point of Sales)* 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 (Sukandar, 2019). TPOS dipesan secara khusus oleh salah satu klien PT. Javasigna Intermedia. Adapun masalah yang mendorong klien untuk dibuatkan sistem TPOS adalah hambatan manajemen transaksi penjualan dan stok barang pada masing-masing toko yang tersebar di berbagai wilayah. TPOS diharapkan dapat membantu transaksi penjualan dan perhitungan stok barang pada toko milik klien. 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.

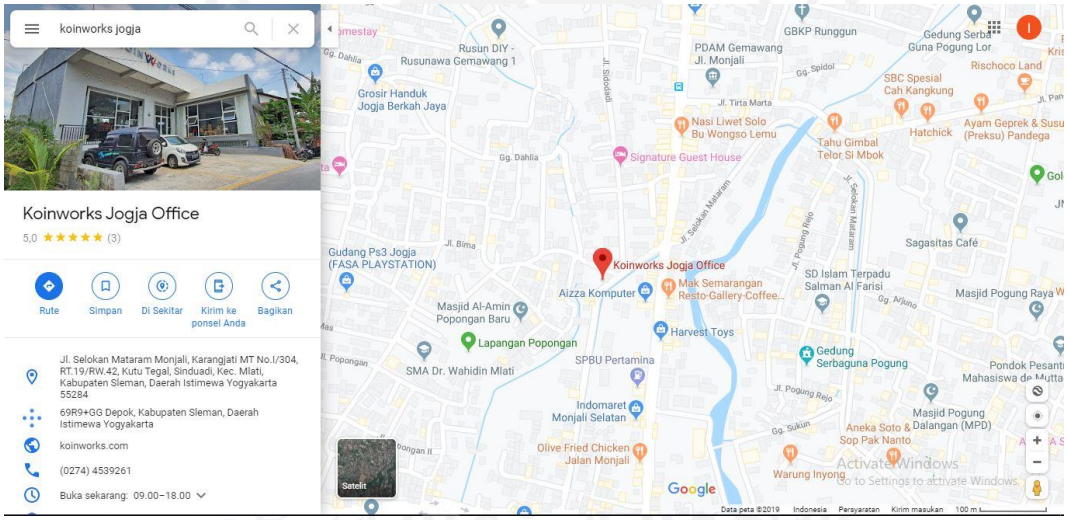
TPOS merupakan suatu sistem yang cukup kompleks dilihat dari banyaknya kebutuhan sistem. Proses integrasi kode program menjadi hambatan saat pengembangan sedang berlangsung. Hal ini karena berbagai macam penyebab, di antaranya adalah perbedaan aturan penulisan kode program setiap *programmer* berbeda dan pembagian kelas yang kurang jelas, seperti penggabungan beberapa kelas yang seharusnya 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.

Untuk memudahkan integrasi kode program dan memecah kompleksitas sistem menjadi kelas kelas berdasar fungsinya, diperlukan aturan pengembengan sistem yang baku. Adapun salah satu konsep yang dapat digunakan untuk mempermudah integrasi kode program dan memecah kompleksitas sistem adalah konsep MVC. MVC adalah arsitektur pengembangan aplikasi berbasis web. MVC membagi sistem menjadi tiga bagian utama yang saling terhubung. Bagian tersebut berupa: *model*, *view*, dan *controller*. Pemetaan sistem pada kelas *model*, *view*, dan *controller* dapat memecah kompleksitas sistem menjadi elementer yang bertugas sesuai kelas dan fungsinya masing-masing. Pada pekerjaan kali ini akan dilakukan pemetaan dan implementasi kelas MVC sistem

TPOS menggunakan *framework* CodeIgniter. Diharapkan penerapan arsitektur MVC dapat memudahkan pengembang melakukan pengembangan aplikasi secara modular.

1.1.1 Profil Perusahaan Magang

PT. Javasigna Intermedia (Javasign) adalah perusahaan IT (*Information Technology*) yang bergerak di bidang pengembangan perangkat lunak (aplikasi) berbasis *mobile*, *desktop* maupun *website*. Aplikasi yang dikembangkan berdasar pada kebutuhan klien, karena setiap klien memiliki kebutuhan yang berbeda-beda bergantung pada fungsi dan tujuan klien. PT. Javasigna Intermedia terletak di Jl. Selokan Mataram Monjali, Karangjati MT No.I/304, RT.19/RW.42, Kutu Tegal, Sinduadi, Kec. Mlati, Kabupaten Sleman, Daerah Istimewa Yogyakarta. Lokasi lebih jelas pada Gambar 1.1. Jadwal kegiatan PT. Javasigna Intermedia dimulai hari Senin hingga Jumat, dengan jam kerja *flexible* namun umumnya dari jam 09.30 hingga 17.30 WIB.

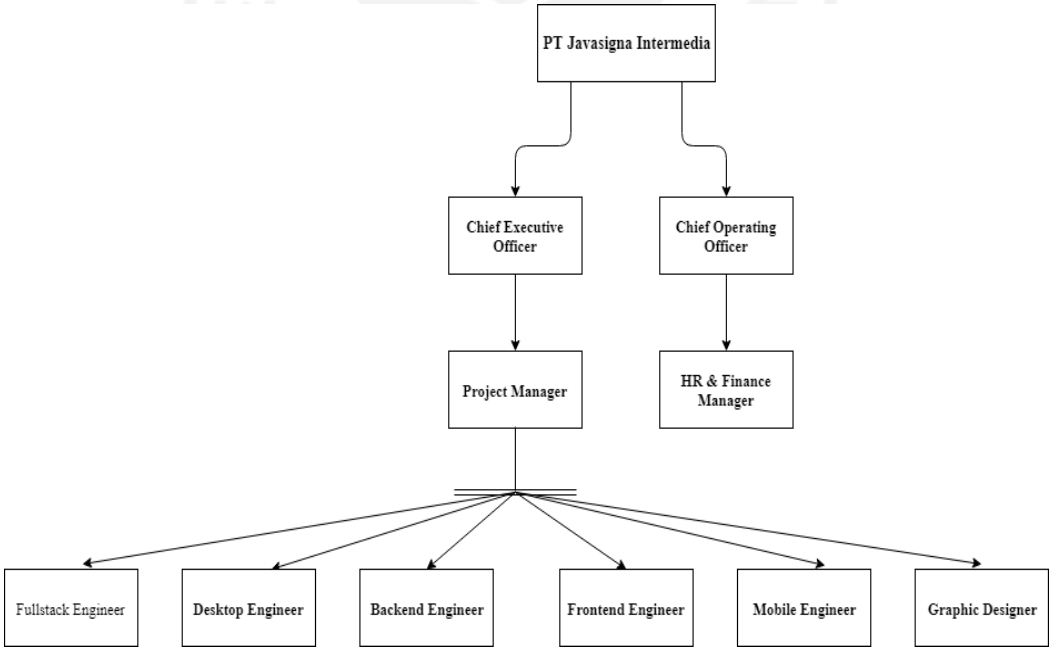


Gambar 1.1 Lokasi PT. Javasigna Intermedia Yogyakarta

Struktur Organisasi PT. Javasigna Intermedia terdiri dari beberapa divisi yang selengkapnya dapat dilihat pada Gambar 1.2. Untuk saat ini Javasign dipimpin oleh Bapak Andwi Valentine. Tugas dari divisi yang ada di PT. Javasigna Intermedia adalah sebagai berikut (Sudiroh, 2019):

a. *Chief Executive Officer*, bertanggung jawab merencanakan, mengelola, dan menganalisis fungsional bisnis seperti operasional, sumber daya manusia, keuangan dan pemasaran.
b. *Chief Operating Officer*, yang memiliki posisi tepat di bawah *Chief Executive Officer (CEO)* yang melaksanakan beberapa tugas seperti membantu pengambilan keputusan di dalam perusahaan.

c. *HR & Finance Manager,* bertugas untuk mengelola sumber daya manusia dan juga mengelola fungsi akuntansi dalam memproses data dan informasi keuangan untuk menghasilkan laporan keuangan yang dibutuhkan perusahaan secara akurat dan tepat waktu.
d. *Marketing Staff,* bertugas untuk memasarkan dan menjual produk.
e. *Project Manager,* bertanggung jawab mengontrol proyek yang ditanganinya. Proyek harus selesai sesuai dengan *budget*, sesuai dengan spesifikasi, dan waktu.
f. *Full Stack Engineer,* bertugas untuk pembuatan aplikasi dari *frontend* dan *backend*nya sendiri.
g. *Desktop Engineer,* bertugas untuk pembuatan aplikasi *desktop*.
h. *Backend Engineer,* bertugas untuk pembuatan fungsi-fungsi sebuah aplikasi.
i. *Frontend Engineer,* bertugas untuk membuat tampilan sebuah aplikasi.
j. *Mobile Engineer,* bertugas untuk pembuatan aplikasi *mobile android* dan *ios*.
k. *Quality Assurance Engineer,* bertugas untuk melakuan pengetesan aplikasi yang sudah jadi dan melaporkannya apabila ada fungsi yang masih salah atau *error*.



Gambar 1.2 Divisi di PT. Javasigna Intermedia (Sudiroh, 2019)

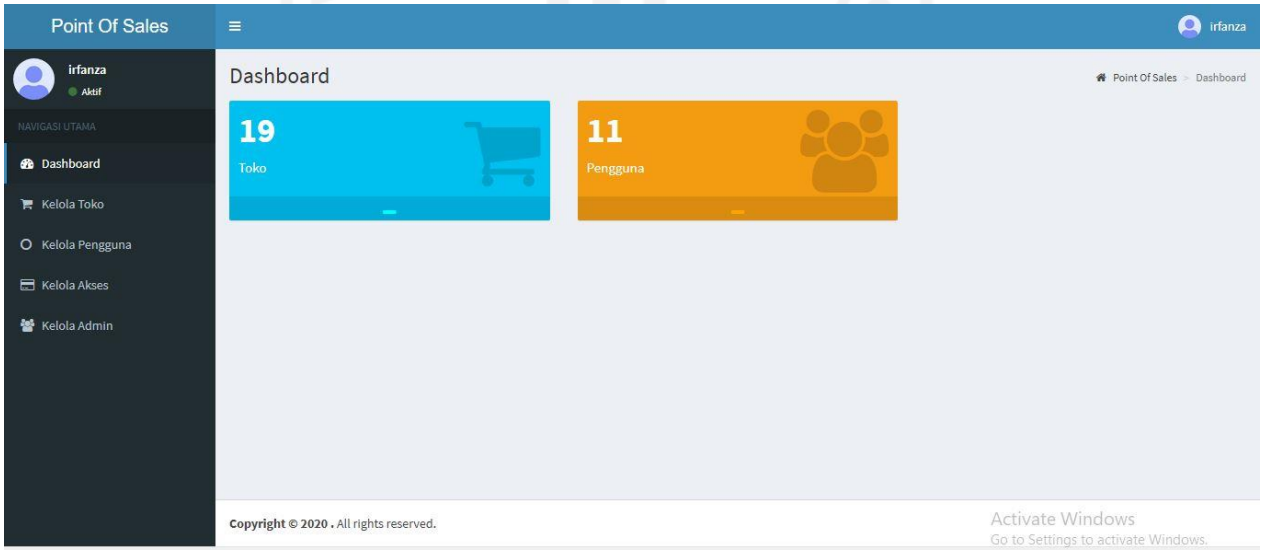
1.1.2 Keterlibatan dalam Proyek

Penulis mengembangkan dua aplikasi berbasis web, yaitu TPOS dan Troli di PT. Javasigna Intermedia selama 6 bulan. Laporan ini akan fokus membahas aplikasi TPOS. TPOS adalah

aplikasi web untuk mengelola penjualan produk pada tiap titik toko di berbagai wilayah di Indonesia, dalam bahasa Inggris konsep ini disebut *Point of Sales*. Secara teori *Point of Sales* 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 (Sukandar, 2019).

TPOS memiliki 3 aktor dan 2 situs. Aktor TPOS terdiri dari super admin, *user*, dan pegawai. Super admin dapat mengelola toko, pengguna, hak akses, dan admin di situs khusus untuk super admin. *User* dan pegawai memiliki tugas yang sama yaitu mengelola pelanggan, produk, pesanan, pegawai toko dan pengaturan toko di situs khusus untuk pegawai dan *user*.

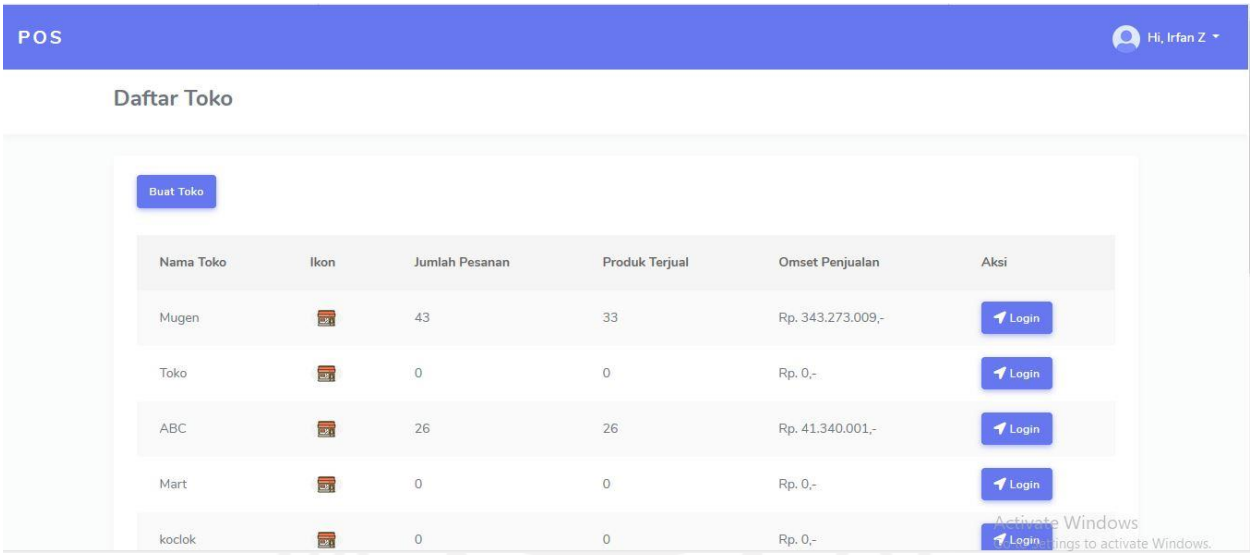
Super admin dapat mengelola toko berupa menambah, mengedit, menghapus dan melihat informasi toko di halaman kelola toko. Halaman pertama setelah login adalah Dashboard yang menampilkan jumlah toko dan jumlah pengguna. Super admin dapat mengubah status pengguna, menambah, mengedit, dan melihat informasi pengguna di halaman kelola pengguna. Halaman Akses menyediakan fitur pengubahan status, edit, dan hapus halaman pada situs kedua atau situs *user* dan pegawai. Halaman terakhir yaitu halaman Kelola Admin, di halaman ini super admin dapat menambah dan menghapus super admin baru. Gambar 1.3 menampilkan situs super admin.



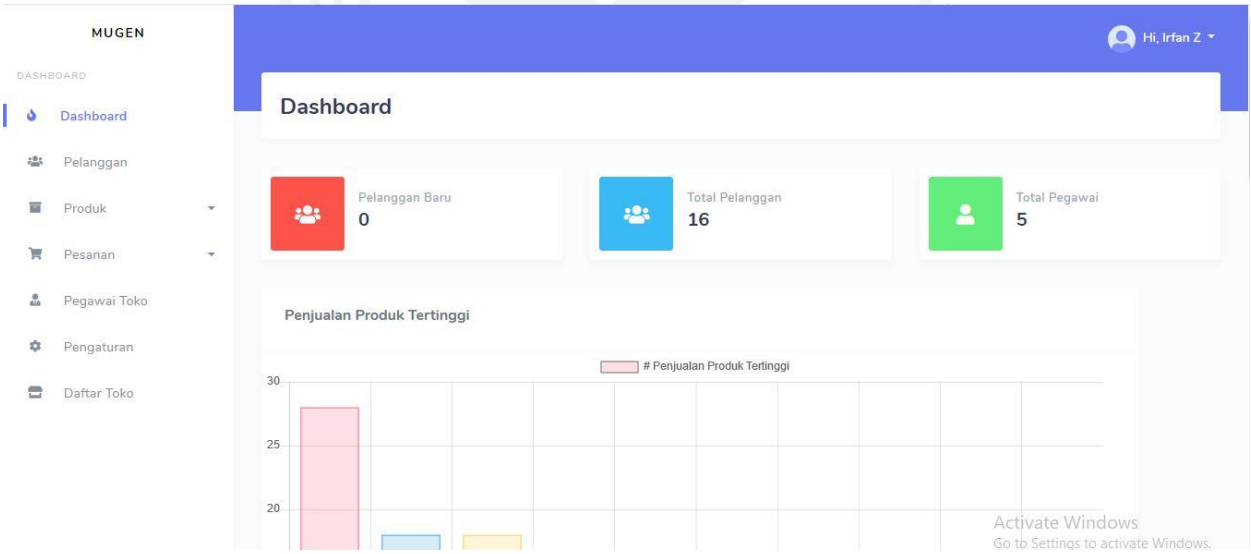
Gambar 1.3 Situs super admin

Situs ke-2 yaitu situs *user* dan pegawai. *User* dan pegawai hanya dapat mengelola toko yang mereka tangani (Lihat Gambar 1.4). Halaman Toko menampilkan halaman Dashboard, Pelanggan, Produk, Pesanan, Pegawai Toko, dan Pengaturan. Gambar 1.5 menampilkan halaman toko.

Halaman toko memuat berbagai informasi dari suatu toko berupa informasi penjualan, jumlah produk, daftar pesanan, pembayaran, info grafis, pegawai toko, dan sejumlah informasi lain yang berkaitan dengan toko.



Gambar 1.4 Daftar Toko



Gambar 1.5 Halaman Toko

Pada proses pengembangannya, TPOS menggunakan teknologi sebagai berikut:

a. Codeigniter, sebagai kerangka kerja sistem agar menghasilkan struktur pemrograman yang rapi
b. MySQL, untuk manajemen basis data

c. JSON, untuk menyimpan dan mengirim data terstruktur antara server dengan klien
d. Bootstrap, untuk membuat tampilan web yang menarik dan responsif
e. JQuery, sebagai *library* Javascript yang membantu mengatur interaksi antara JavaScript dan HTML
f. JavaScript, untuk membuat halaman web lebih dinamis dan interaktif
g. AJAX, untuk mempersingkat komunikasi dengan *server* pada kasus tertentu
h. PHP, sebagai bahasa pemrograman *script server-side* yang didesain untuk pengembangan web
i. Data Table, sebagai *plug-in* untuk *library* jQuery untuk memanipulasi data dalam tabel HTML

Teknologi-teknologi ini diharapkan dapat memenuhi semua kebutuhan pengguna.

1.2 Ruang Lingkup Magang

Penulis melaksanakan magang di PT. Javasigna Intermedia (Javasign) selama enam bulan, mulai dari akhir bulan Oktober 2019 hingga akhir April 2020. Perusahaan ini bergerak di bidang teknologi informasi khususnya dalam pengembangan perangkat lunak (*software house development*). Selama pelaksanaan penulis telah terlibat dalam beberapa proyek pengembangan perangkat lunak. Adapun aktivitas yang dilakukan selama magang adalah sebagai berikut:

a. Pengembangan sistem TPOS menggunakan *framework* Codeigniter. Pola penulisan kode program berdasar pada *framework* yang digunakan.
b. Proyek Kamar Bayi Rent. Penulis terlibat dalam melakukan perbaikan *bug* dan penambahan beberapa fitur di menu sewa barang.
c. Pemetaan fungsi program di kelas *model, view,* dan *controller* berdasar pada kebutuhan fungsionalitas sistem.
d. Pengerjaan fitur sesuai tugas yang diberikan oleh *project manager*.

1.3 Tujuan

Tujuan pekerjaan ini adalah untuk mengembangkan sistem TPOS dengan memanfaatkan *framework* Codeigniter untuk membuat struktur pemrograman dan dokumentasi program tertata dan memanfaatkan konsep MVC untuk memetakan fungsi program berdasar kelasnya.

1.4 Manfaat

Pengembangan sistem TPOS menggunakan konsep MVC yang dikerjakan selama berjalannya magang mempunyai beberapa manfaat. Manfaat dari implementasi MVC pada sistem TPOS adalah sebagai berikut:

a. Memudahkan proses penambahan, perubahan, atau penghapusan kode program
b. Memudahkan proses integrasi kode program
c. Memudahkan pengembangan sistem secara modular

1.5 Sistematika Penulisan

Untuk menguraikan topik yang akan dibahas, laporan ini menggunakan sistematika penulisan sebagai berikut:

a. BAB I – PENDAHULUAN

   Bab ini akan menguraikan tentang latar belakang Sistem TPOS, ruang lingkup magang, tujuan pemanfaatan MVC, manfaat penggunaan MVC, dan sistematika penulisan.

b. BAB II – KAJIAN PUSTAKA

   Bab ini akan membahas mengenai teori yang digunakan dalam pengembangan sistem TPOS, yaitu MVC, CodeIgniter, dan Sistem POS beserta ringkasan penelitian sebelumnya.

c. BAB III – PELAKSANAAN MAGANG

   Bab ini membahas aktivitas yang telah dilakukan saat pelaksanaan magang, penjelasan proyek apa yang dikerjakan dan pembahasan mengenai arsitektur MVC.

d. BAB IV – REFLEKSI PELAKSANAAN MAGANG

   Bab ini membahas perbandingan implementasi proyek dengan teori yang ada.

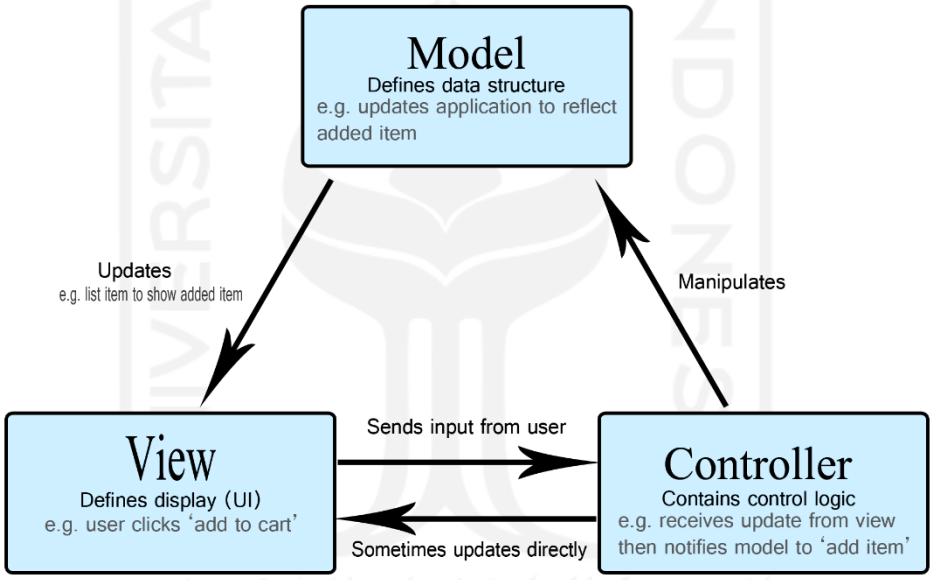
e. BAB V – KESIMPULAN DAN SARAN

   Bab ini berisi kesimpulan dari implementasi proyek dan saran terhadap pengembangan selanjutnya.

BAB II
KAJIAN PUSTAKA

2.1 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 (Reenskaug & Coplien, 2009).



Gambar 2.1 Skema MVC (*Model View Controller*) (MDN contributors, 2019)

Pada Gambar 2.1 dapat dilihat bahwa kelas *model* bertugas untuk menyediakan, memanipulasi dan mengorganisasikan data dari basis data sesuai dengan perintah dari kelas *controller* kemudian data terbaru akan ditampilkan melalui kelas *view*. Kelas *view* bertugas untuk menampilkan informasi kepada pengguna sesuai arahan dari kelas *controller* atau kelas *model*, kelas *view* menerima hasil masukan dari pengguna seperti klik tombol. Kelas *controller* berfungsi untuk mengatur tugas yang harus dilakukan *model* dan *view* mana yang harus ditampilkan berdasarkan permintaan dari *user*.

Beberapa sistem *point of sales* telah dikembangkan menggunakan arsitektur MVC menggunakan *framework* Laravel, salah satu contohnya adalah Aplikasi *Point of Sales* menggunakan *framework* Laravel (Tahir, Rais, & Apriyadi, 2019). 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.

Makalah lain membahas tentang penerapan arsitektur *Model View Controller* (MVC) dalam rancang bangun sistem kuis *online* adaptif (Hidayat & Surarso, 2012). 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.

2.2 CodeIgniter

CodeIgniter merupakan salah satu *framework* pengembangan aplikasi web. *Framework* mempunyai arti struktur dasar yang melandasi sebuah sistem. Sehingga keseluruhan sistem bergerak berdasarkan kerangka struktur dasar yang telah dibentuk sebelumnya (Oxford English Dictionary, 2020). Berkaitan dengan aplikasi web, *web framework* dapat diartikan sebagai kerangka yang digunakan untuk pengembangan aplikasi web termasuk web *service*, web *resources*, dan antarmuka pemrograman aplikasi web. Sebagian besar *framework* web telah menyediakan *library*, *framework template*, manajemen sesi, dan penggunaan kembali kode program.

CodeIgniter adalah kerangka kerja PHP yang sangat kuat dengan ukuran yang sangat kecil, dibuat untuk pengembang yang membutuhkan perangkat sederhana dan elegan untuk membuat aplikasi web berfitur lengkap (Codeigniter, 2020). *Framework* ini masih terus dikembangkan hingga laporan ini dibuat. Rilis terakhir adalah versi 4.0.4 yang bisa diunduh pada web resmi CodeIgniter.

Adapun publikasi ilmiah dengan judul Pemanfaatan Framework Codeigniter dalam Pengembangan Sistem Informasi Pendataan Laporan Kerja Praktek Mahasiswa Program Studi

Teknik Informatika Unsoed (Afuan, 2010). Tujuan penggunaan *framework* CodeIgniter dalam pengembangan sistem tersebut adalah mempermudah pengembang web mengembangkan aplikasi web yang kokoh secara cepat tanpa kehilangan fleksibilitas. Sistem tersebut digunakan untuk memudahkan admin melakukan pendataan laporan kerja praktek mahasiswa pada prodi Teknik Informatika Unsoed. Pada publikasi ilmiah tersebut dituturkan bahwa dengan menggunakan CodeIgniter, memungkinkan pengembangan sebuah web dapat dilakukan dengan cepat dan mempermudah pengolahan web tersebut. Hal ini karena CodeIgniter sudah memisahkan antara data dengan presentasi dan memiliki banyak fitur yang memudahkan pengembangan situs web.

Makalah lain yang membahas pengembangan sistem menggunakan *framework* CodeIgniter berjudul Implementasi Manajemen Perpustakaan menggunakan Framework Codeigniter (CI) Dengan Teknik Hierarchical model–view–controller (HMVC) (Muzakir, 2014). Sistem ini dikembangkan karena proses pengelolaan data dan pelayanan pada perpustakaan di SMA Negeri 1 Sungai Lilin masih dilakukan secara manual. Tujuan dikembangkan sistem tersebut adalah Untuk mempermudah pelayanan dan akses informasi serta pengelolaan data perpustakaan di antaranya, mempermudah pencarian informasi koleksi buku, informasi anggota, registrasi, pengunjung, peminjaman dan pengembalian buku, serta laporan. Arsitektur HMVC (*Hierarchical Model View Controller*) dipilih karena memiliki keuntungan yang akan didapat antara lain mudah untuk dilakukan perbaikan, penambahan atau pengurangan kode pada pengembangan perangkat lunak yang besar. Kesimpulan dari makalah tersebut adalah sistem berhasil dikembangkan menggunakan *framework* CodeIgniter dengan konsep HMVC. Konsep HMVC memberikan kemudahan kepada pengembang sistem dalam memanfaatkan kode program yang ada.

2.3 Sistem POS (Point of Sales)

Sistem POS (*Point of Sales*) 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 (United States Patent No. US 2012/0296679 A1, 2012). Sistem POS digunakan untuk mengintegrasikan sistem reservasi, pemesanan, dan sistem *merchant e-commerce* lainya yang disediakan melalui situs web. Sistem 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

(United States Patent No. US 2004/0181454 A1, 2004). 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.

Contoh makalah lain membahas pengembangan sistem POS adalah Pembangunan Sistem Informasi *Point of Sales* Terintegrasi Dalam Lingkup Rumah Makan Beserta Cabangnya (Studi Kasus: RM. Pecel Pincuk Bu Tinuk) (Sani, Pradana, & Rusdianto, 2018). Makalah tersebut menjelaskan proses pengembangan sistem POS untuk digitalisasi proses konvensional yang sudah ada. Beberapa sebab dikembangkan sistem tersebut adalah kendala dalam menjalankan proses bisnis yang sudah ada dan dalam mengumpulkan informasi berupa pendapatan, pengeluaran dan stok dari masing-masing outlet. Tujuan dikembangkan sistem tersebut agar pemilik dapat melihat informasi secara langsung dan akurat tanpa harus membuat pembukuan yang memakan banyak waktu. Konsep MVC digunakan setelah semua kebutuhan sistem diketahui. Setelah melalui pengujian *whitebox* dan *blackbox*, web dinyatakan telah memenuhi semua kebutuhan sistem.

BAB III
PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Pengembangan sistem TPOS telah dilaksanakan melalui tahapan-tahapan manajemen proyek sebagai berikut:

3.1.1 Pendefinisian Proyek

TPOS merupakan sebuah aplikasi yang dikembangkan oleh PT. Javasigna Intermedia untuk mengelola penjualan produk pada tiap titik toko di berbagai wilayah di Indonesia melalui situs web. Tujuan dikembangkan sistem tersebut untuk mempermudah klien mengelola barang, transaksi penjualan, dan pembuatan laporan penjualan. 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 pencatatan laporan penjualan.

TPOS dikembangkan oleh tim yang terdiri dari *project manager*, pengembang, dan *tester*. *Project manager* bertugas mengatur jalannya proyek agar mencapai hasil yang sudah ditentukan, salah satu contoh pekerjaan *project manager* adalah menerjemahkan kebutuhan sistem menjadi tugas-tugas yang nantinya akan dikerjakan oleh tim pengembang. Pengembang bertugas membuat desain bersama *project manager* dan implementasi program. *Tester* bertugas menguji semua fungsi sistem apakah dapat berjalan sesuai parameter yang ditentukan atau tidak. Jadwal eksekusi proyek dilakukan dalam 5 hari kerja yaitu dari hari Senin hingga hari Jumat dengan 8 jam kerja sehari.

TPOS dikembangkan sebagai suatu sistem aplikasi berbasis web menggunakan spesifikasi sistem seperti yang ditunjukan oleh Tabel 3.1

Tabel 3.1 Arsitektur Teknologi TPOS

Komponen	**Spesifikasi**
Basis Aplikasi	*Web Application*

Komponen	Spesifikasi
Platform	PHP 7.3
Framework	CodeIgniter
Arsitektur	MVC
Javascript *Library*	DataTabes
CSS *Library*	Bootstrap
Basis Data	MySQL
Browser	Firefox, Chrome, Microsoft Edge
Sistem Operasi	Windows 10

3.1.2 Perencanaan Proyek

Pengembangan sistem TPOS memiliki beberapa tahapan. Tahapan-tahapan tersebut adalah tahap analisis kebutuhan sistem, tahap desain, tahap implementasi, tahap uji coba dan tahap akhir berupa pemeliharaan sistem setelah sistem berhasil dikembangkan dan diserahkan kepada klien. Lingkup pengerjaan proyek berupa pengerjaan tugas-tugas yang akan dibagi oleh *project manager*. Waktu pengembangan proyek dilakukan secara berkelanjutan (fleksibel), lebih tepatnya proyek mulai dikerjakan tanggal 17 Juli 2019 dan perkiraan selesai pada akhir bulan April 2020. Setelah tidak ada lagi daftar tugas yang harus dikerjakan oleh pengembang pada antrian tugas, maka pimpinan dari *project manager* akan memperbarui web yang aktif dengan versi sistem web terbaru. Beberapa contoh tugas dalam bentuk kebutuhan fungsionalitas sistem TPOS dapat dilihat pada Tabel 3.2.

Seperti yang sudah dijelaskan di subsubbab 1.1.2 Keterlibatan Dalam Proyek, TPOS memiliki 3 aktor yaitu super admin, *user,* dan pegawai. Terdapat 2 situs utama, yaitu situs untuk super admin dan situs untuk *user* dan pegawai. Super admin akan mengelola situs khusus untuk super admin. *User* bertugas mengelola situs khusus untuk *user* dan pegawai. Perbedaan anatara *user* dan

pegawai terletak pada hak aksesnya. Pegawai hanya dapat mengakses menu yang diizinkan oleh *user*.

Tabel 3.2 Kebutuhan fungsionalitas sistem TPOS

Kode	Deskripsi
TF-1	Sistem harus dapat melakukan registrasi via nomor telepon dan *email.*
TF-2	Sistem harus dapat *login* menggunakan nomor telepon dan *email.*
TF-3	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus toko.
TF-4	Sistem harus dapat *login* berdasar hak akses yang dimiliki pengguna.
TF-5	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* di situs *user* dan pegawai.
TF-6	Sistem harus dapat menambah, mengedit, menampilkan, dan menonaktifkan data pelanggan di situs *user* dan pegawai.
TF-7	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus data produk di situs *user* dan pegawai.
TF-8	Sistem harus dapat membagikan produk via *whatsapp* di situs *user* dan pegawai.
TF-9	Sistem harus dapat menyimpan produk sebagai *draft* di situs *user* dan pegawai.
TF-10	Sistem harus dapat mencari produk berdasar kode atau nama produk di situs *user* dan pegawai.
TF-11	Sistem harus dapat memindahkan produk ke toko lain di situs *user* dan pegawai.
TF-12	Sistem harus dapat menambah dan mengedit kategori dan sub kategori produk di situs *user* dan pegawai.
TF-13	Sistem harus dapat melakukan proses pemesanan produk di situs *user* dan pegawai.
TF-14	Sistem harus dapat menyimpan data pesanan di situs *user* dan pegawai.
TF-15	Sistem harus dapat menampilkan, mencetak, memfilter berdasarkan tanggal, dan mencari data pesanan di situs *user* dan pegawai.
TF-16	Sistem harus dapat mengubah status pesanan dan mengirimkan informasi status ke pelanggan melalui *email* di situs *user* dan pegawai.

Kode	Deskripsi
TF-17	Sistem harus dapat mencetak dokumen *invoice* dan mengirimkan dokumen *invoice* melalui *whatsapp* dan *email* di situs *user* dan pegawai.
TF-18	Sistem harus dapat mengatur pesanan berupa metode pengiriman dan metode pembayaran produk di situs *user* dan pegawai.
TF-19	Sistem harus dapat mengembalikan jumlah stok produk jika pesanan dikembalikan di situs *user* dan pegawai.
TF-20	Sistem harus dapat menampilkan info grafis penjualan, pelanggan, dan lokasi di situs *user* dan pegawai.
TF-21	Sistem harus dapat menambah, mengedit, menampilkan, dan menghapus data pegawai toko di situs *user* dan pegawai.
TF-22	Sistem harus dapat memfilter pegawai toko berdasar hak akses di situs *user* dan pegawai.
TF-23	Sistem harus dapat mengedit profil pengguna di situs *user* dan pegawai.
TF-24	Sistem harus dapat mengubah menu akses *user* dan pegawai di situs *user* dan pegawai.
TF-25	Sistem harus dapat menampilkan jumlah toko dan jumlah pengguna di halaman *dashboard* di situs super admin.
TF-26	Sistem harus dapat menambah, menampilkan, mengedit, dan menghapus toko di situs super admin.
TF-27	Sistem harus dapat memfilter, mengedit, menampilkan, dan mengubah status *user* dan pegawai di situs super admin.
TF-28	Sistem harus dapat mengedit dan menghapus menu situs *user* dan pegawai di situs super admin.
TF-29	Sistem harus dapat menambah dan menghapus super admin di situs super admin.
TF-30	Sistem harus dapat mengedit profil super admin di situs super admin.

3.1.3 Pelaksanaan Proyek

Sistem TPOS telah dikembangkan menggunakan metode *waterfall*. Berikut merupakan penjelasan secara singkat tahapan pengembangan aplikasi menggunakan metode *waterfall* (Munassar & Govardhan, 2010):

1. *Requirements analysis*

Tahap ini merupakan proses menganalisis kebutuhan fungsionalitas dan batasan sistem TPOS yang akan dikembangkan. Informasi didapat dari diskusi dengan klien. Setelah informasi terkumpul, dilakukan proses analisis kebutuhan fungsionalitas sistem. Tabel analisis kebutuhan fungsionalitas sistem TPOS dapat dilihat pada Tabel 3.2. Berkaitan dengan konsep MVC, pemetaan kelas *model*, *view*, dan *controller* akan dilakukan setelah semua kebutuhan sistem TPOS diketahui.

2. *System Design*

Tahap selanjutnya setelah kebutuhan fungsionalitas 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 mulai dilakukan pemetaan kelas MVC, setiap kelas *model, view*, dan *controller* memiliki fungsi atau prosedur program yang sesuai dengan kelasnya masing-masing.

3. *Implementation*

Langkah selanjutnya yaitu proses mengubah rancangan yang telah ditentukan menjadi kode program. Pembuatan kelas MVC dari rancangan kelas sebelumnya mulai diimplementasikan. Pada proses ini juga, akan dilakukan penulisan semua kode program yang telah dirancang termasuk fungsi atau prosedur program pada masing-masing kelas *model, view,* dan *controller.*

4. *Integration and Testing*

Pada tahap ini modul-modul digabungkan, dan dilakukan pengetesan oleh *tester* 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. Jika semua fungsi program berjalan dengan baik, maka tugas dianggap selesai.

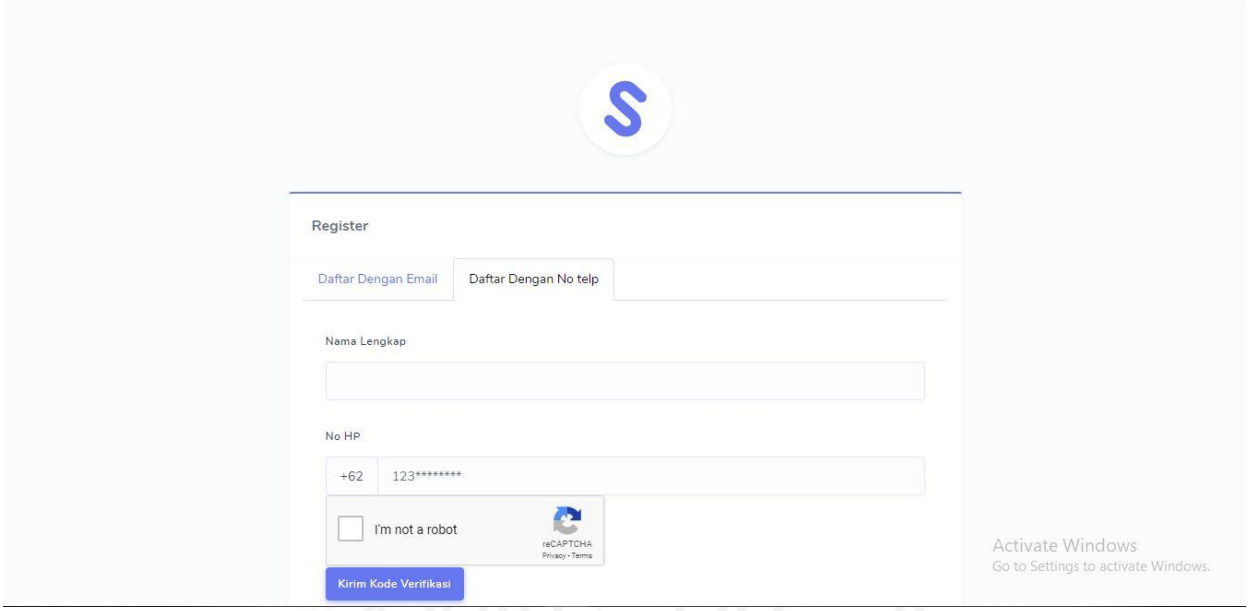
5. *Operation and Maintenance*

Tahap ini merupakan proses pemeliharaan program berupa perbaikan *bug* atau peningkatan sistem TPOS setelah sistem diserahkan kepada klien. Klien akan memberikan tanggapan terhadap sistem yang sudah dikembangkan. Jika klien menginginkan penambahan fitur baru, maka akan dilakukan peningkatan sistem. Peningkatan sistem dapat menjadi kebutuhan baru.

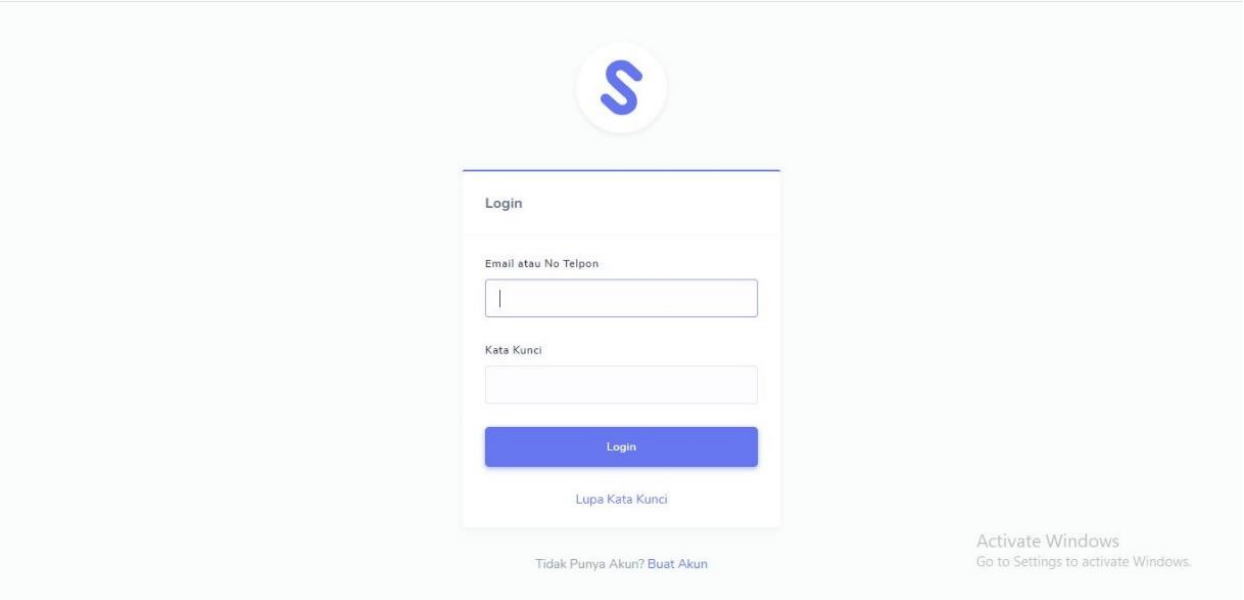
Dalam pelaksanaan proyek, beberapa aktivitas dari keseluruhan aktivitas yang telah dilaksanakan oleh penulis adalah sebagai berikut:

1. Membuat Autentikasi dan Login Via Nomor Telepon

Pengguna TPOS dibedakan menjadi 3 aktor yaitu, super admin, *user*, dan pegawai. Setiap aktor memiliki peran yang berbeda seperti yang telah dijabarkan pada bab 1. Sebelum dapat masuk ke dalam sistem untuk pertama kali, *user* dan pegawai diharuskan melakukan registrasi akun melalui *email* atau nomor telepon. Firebase digunakan untuk melakukan autentikasi secara otomatis pada fitur autentikasi via nomor telepon. Tampilan halaman registrasi via nomor telepon dan halaman login dapat dilihat pada Gambar 3.1 dan Gambar 3.2 secara berurutan.



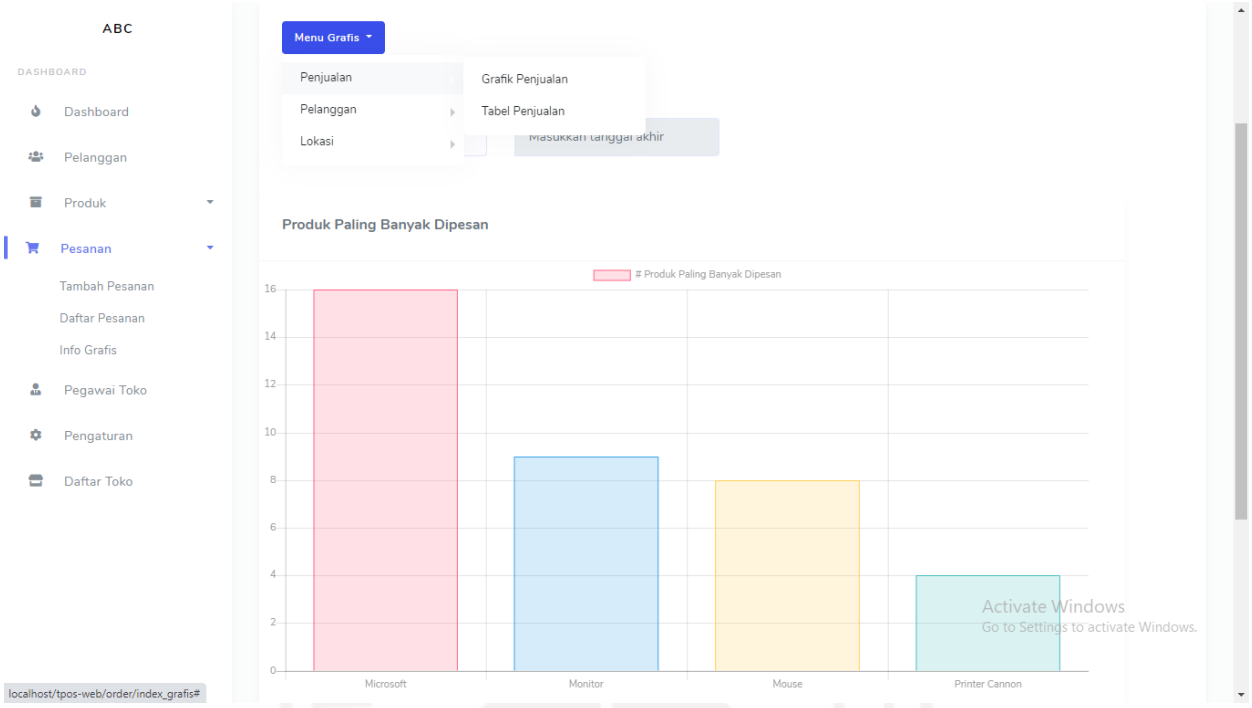
Gambar 3.1 Halaman registrasi via nomor telepon



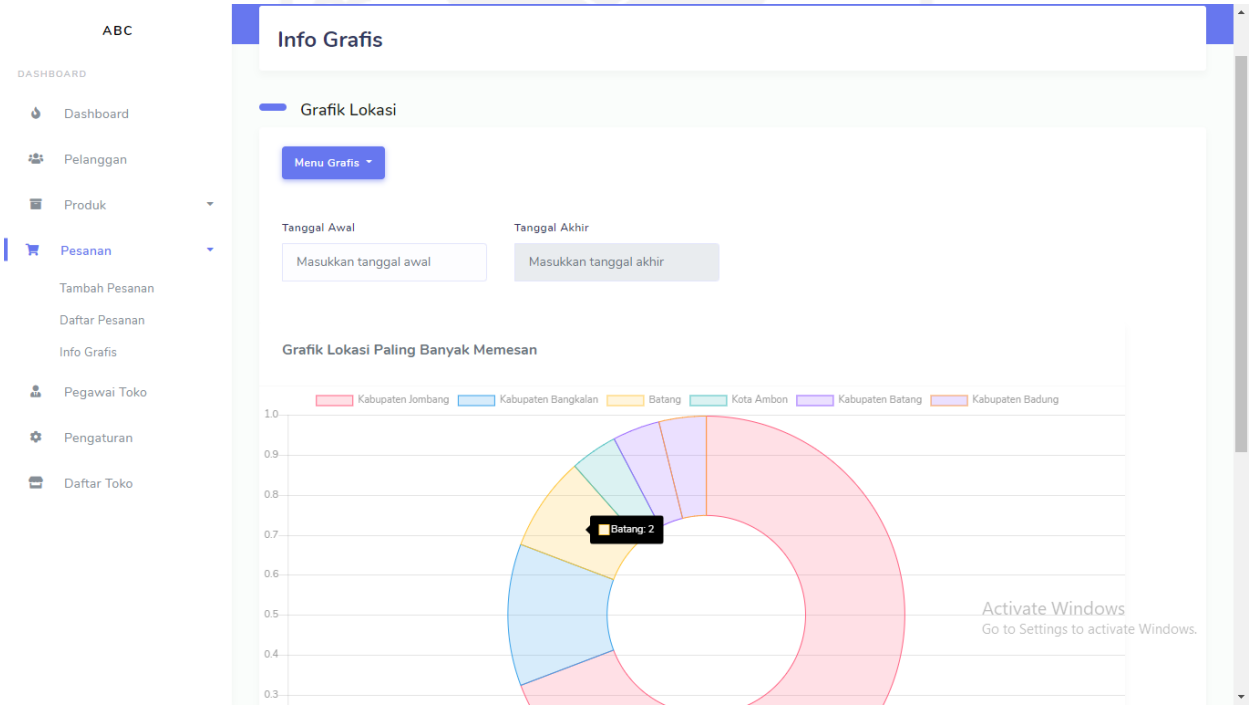
Gambar 3.2 Halaman login

2. Membuat Fitur Info Grafis

Info grafis menampilkan informasi penjualan, pelanggan, dan lokasi pemesan. Setiap sub menu info grafis menampilkan grafik dan tabel. Info grafis penjualan menampilkan grafik dan tabel produk paling banyak dipesan oleh pelanggan. Info grafis pelanggan menampilkan grafik dan tabel pelanggan yang paling sering memesan produk. Info grafis lokasi menampilkan grafik dan tabel daerah dengan jumlah pesanan terbanyak. Setiap halaman pada sub menu info grafis memiliki filter tanggal, filter tanggal digunakan untuk menampilkan data terbanyak hingga terkecil berdasar masukan tanggal awal dan akhir. Gambar 3.3 dan Gambar 3.4 menunjukan grafik penjualan dan grafik lokasi secara berurutan.



Gambar 3.3 Grafik penjualan

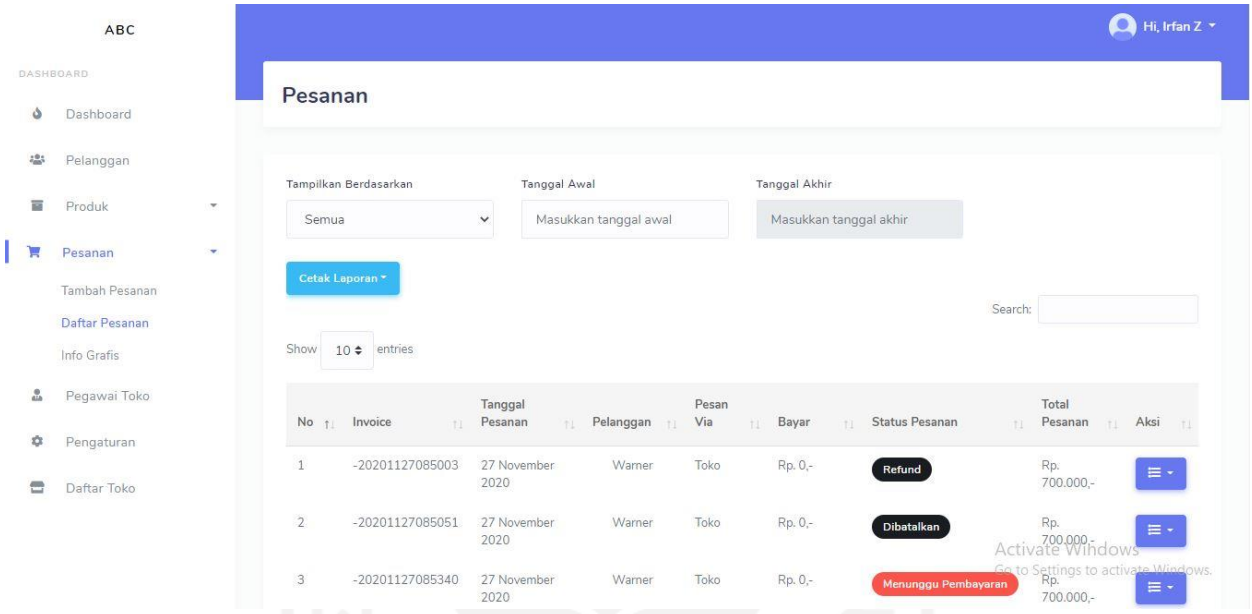


Gambar 3.4 Grafik lokasi

3. Membuat Fitur Refund dan Dibatalkan di Daftar Pesanan

Fitur ini memungkinkan pelanggan membatalkan atau mengembalikan pesanan yang telah dipesan. Fitur ini dibuat untuk mengantisipasi kesalahan pengiriman atau masalah kepuasan

pelanggan terhadap produk yang telah dipesan. Jika pesanan berstatus dibatalkan atau *refund*, maka produk harus kembali ke database penyimpanan sesuai jumlah dan banyak produk yang telah dipesan. Gambar 3.5 menunjukan tangkapan layar status *refund* dan dibatalkan di daftar pesanan. Gambar 3.6 menampilkan fitur *refund*.



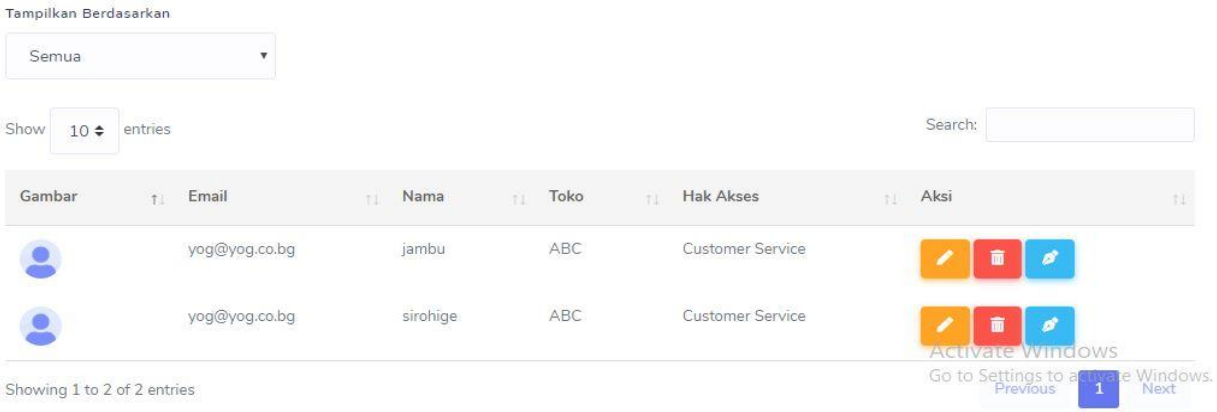
Gambar 3.5 Status *refund* dan dibatalkan di daftar pesanan

Gambar 3.6 Fitur *refund*

4. Menambahkan Filter Berdasarkan Hak Akses Pegawai

Fitur ini dapat menampilkan daftar pegawai berdasarkan hak akses yang dimiliki pegawai. Terdapat empat hak akses yang dimiliki pegawai yaitu, *Customer Service*, Gudang, Keuangan, Manajer. Fitur ini memudahkan pengguna untuk memilah pegawai mana yang ingin ditampilkan

berdasarkan hak akses yang sudah disebutkan sebelumnya. Gambar 3.7 menunjukan tangkapan layar fitur filter berdasar hak akses pegawai.



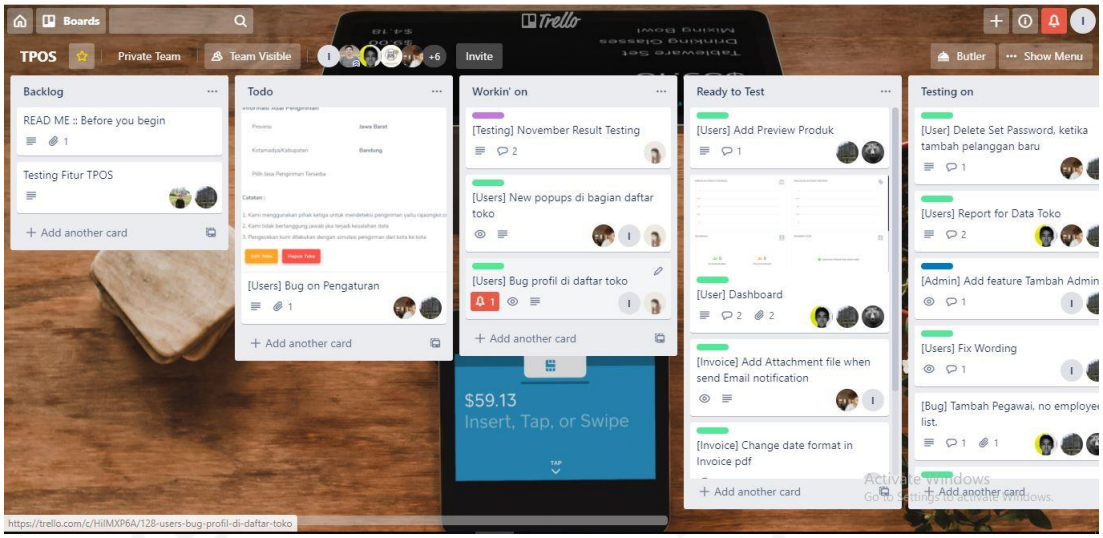
Gambar 3.7 Fitur filter berdasar hak akses pegawai

3.1.4 Pemantauan Proyek

Proses pemantauan dan pengontrolan proyek dilakukan melalui aplikasi pendukung dari pihak ketiga, aplikasi tersebut adalah:

a. Trello

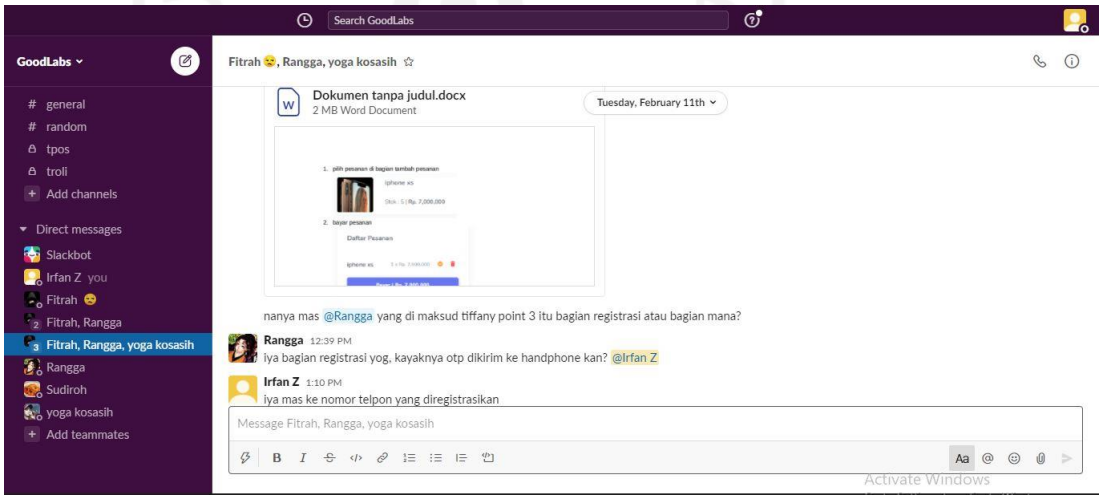
Trello merupakan sebuah aplikasi kolaborasi berbasis web yang digunakan untuk mengelola sebuah proyek. Semua orang dalam proyek tersebut termasuk peserta magang yang berperan sebagai pengembang, dapat mengetahui aktivitas apa yang sedang dikerjakan oleh masing-masing anggota proyek. Gambar 3.8 merupakan tampilan tangkapan layar dari manajemen proyek TPOS pada aplikasi Trello. Aktivitas pada bagian *Backlog* merupakan antrian pekerjaan yang diminta oleh klien berupa hal apa saja yang diharapkan dalam sistem TPOS. Antrian pekerjaan di *Backlog* akan diterjemahkan oleh *project manager* menjadi tugas atau aktivitas yang harus dikerjakan oleh pengembang di antrian *Todo*. Antrian pada *Workin' on* merupakan tugas yang sedang dikerjakan oleh pengembang, jika tugas sudah selesai, maka tugas akan dipindahkan ke antrian *Ready to Test* dan siap untuk diuji oleh *tester*. *Testing on* adalah kumpulan aktivitas yang sudah selesai dikerjakan dan siap untuk di tes oleh *tester*, apakah terdapat *bug* atau *error*. *Ready to Deploy* merupakan antrian aktivitas yang sudah lolos dari *bug* dan *error* tetapi masih perlu pemantauan akhir oleh *project manager*. Jika *project manager* menyatakan suatu aktivitas lolos, maka aktivitas tersebut dinyatakan selesai dan siap untuk dipindahkan ke antrian *Done*.



Gambar 3.8 Manajemen proyek TPOS pada aplikasi Trello

b. Slack

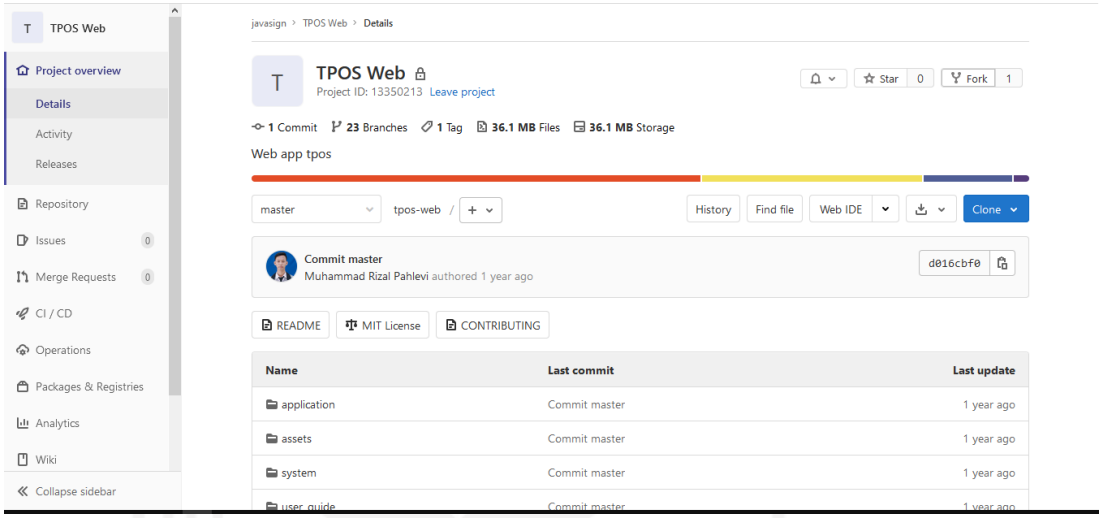
Slack merupakan aplikasi yang digunakan untuk berkomunikasi antar individu atau tim proyek. Slack juga dapat digunakan untuk berbagi berkas, hal ini lebih mudah dilakukan karena berkas yang berisi kode program dapat diulas secara langsung pada desktop tanpa harus mengunduhnya terlebih dahulu. Gambar 3.9 menunjukan tampilan aplikasi Slack.



Gambar 3.9 Tampilan aplikasi Slack

c. GitLab

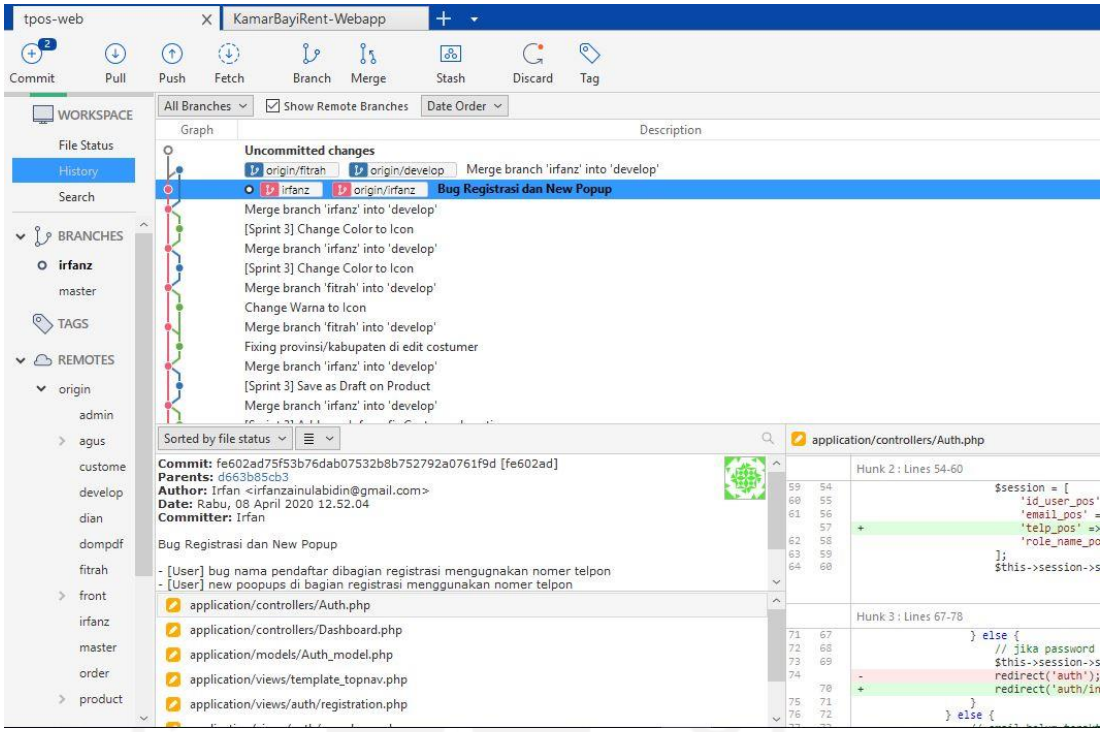
GitLab adalah layanan yang menyediakan *remote access* ke Git *repository*. Proyek TPOS menggunakan layanan GitLab untuk memudahkan penggabungan tugas antar pengembang dan meminimalisir *error* dengan cara pengecekan atau pengambilan kembali versi pengembangan sistem yang sesuai. Gambar 3.10 merupakan tampilan proyek TPOS pada GitLab.



Gambar 3.10 Tampilan proyek TPOS pada GitLab

d. Sourcetree

Sourcetree adalah aplikasi yang memudahkan pengembang berinteraksi dengan Git *repository*. Sourcetree memvisualisasikan pengelolaan Git *repository* melalui GUI Git Sourcetree yang sederhana. Pada pengembangan proyek TPOS, penulis menggunakan Sourcetree untuk keperluan *push*, *pull*, dan *commit* kode program. Selain itu Sourcetree juga membantu melihat perubahan yang telah dilakukan pada *file* termasuk penambahan dan pengurangan *file* gambar. Gambar 3.11 menunjukan tampilan Sourcetree.



Gambar 3.11 Tampilan Sourcetree

3.1.5 Penutupan Proyek

Setelah tidak ada lagi tugas yang harus dikerjakan oleh pengembang pada antrian tugas, maka pimpinan dari *project manager* akan memperbarui web yang aktif dengan versi sistem web terbaru. Pada tahap ini web sudah berhasil dikembangkan dan kegiatan magang sudah diselesaikan. Selanjutnya sistem akan digunakan oleh klien. Jika klien menginginkan fitur tambahan, klien akan berkoordinasi dengan *project manager*. *Project manager* akan menerjemahkan keinginan klien menjadi kebutuhan sistem baru di antrian tugas baru.

3.2 Metode Pengembangan

Pada pengembangan sistem, khususnya dalam penerapan arsitektur MVC dalam sistem TPOS, dilakukan beberapa tahapan. Tahapan tersebut adalah sebagai berikut:

1. Analisis, pada tahap ini akan ditentukan kebutuhan sistem
2. Rancangan Arsitektur, pemetaan kelas MVC dilakukan pada tahap ini
3. Implementasi MVC TPOS, yaitu penulisan kode program pada masing-masing kelas
4. Pengujian, pengecekan fungsi sistem apakah dapat bekerja sesuai semestinya
5. Pemeliharaan, yaitu tahap pemeliharaan sistem setelah selesai dikembangkan

3.2.1 Analisis

Analisis dilakukan dengan tujuan mendapatkan informasi terkait kebutuhan fungsionalitas sistem. Pemetaan kelas *model, view,* dan *controller* dilakukan setelah kebutuhan fungsionalitas TPOS diketahui. Tabel 3.2 di subsubbab Perencanaan Proyek menunjukan 30 kebutuhan fungsionalitas sistem TPOS. Pada laporan ini akan dibahas enam kebutuhan fungsionalitas yang utama.

Deskripsi kebutuhan fungsionalitas:

a. TF-1 Registrasi via nomor telepon dan *email*

   Pegawai dan *user* dapat memilih untuk melakukan registrasi melalui via nomor telepon atau alamat *email*. Diperlukan 2 halaman tampilan, yaitu halaman daftar dengan *email* dan halaman daftar dengan nomor telepon.

b. TF-2 *Login* menggunakan no telepon dan *email*

   Pegawai dan *user* dapat melakukan *login* melalui akun yang sudah didaftarkan. Jika mendaftar menggunakan nomor telepon, maka harus memasukan nomor telepon dan *password* sebelum dapat *login* ke sistem. Jika mendaftar menggunakan *email*, maka harus memasukan *email* dan *password* sebelum dapat *login* ke sistem. Diperlukan 1 halaman tampilan, yaitu halaman login.

c. TF-3 Menambah, mengedit, menampilkan, dan menghapus toko

   Pegawai dan *user* dapat menambah toko baru, mengedit data toko, melihat daftar toko yang dikelola, dan menghapus data toko. Diperlukan 3 halaman tampilan, yaitu halaman buat toko baru, halaman daftar toko, dan halaman pengaturan toko.

d. TF-4 *Login* berdasar hak akses yang dimiliki pengguna.

   Sistem dapat membedakan peran yang dimiliki pengguna selama proses *login*. Jika pengguna adalah super admin, maka akan diarahkan ke halaman *dashboard* super admin, jika pengguna adalah pegawai atau *user*, maka akan diarahkan ke halaman *dashboard* pegawai dan *user.*

e. TF-5 Menampilkan data toko di halaman *dashboard* pegawai dan *user*

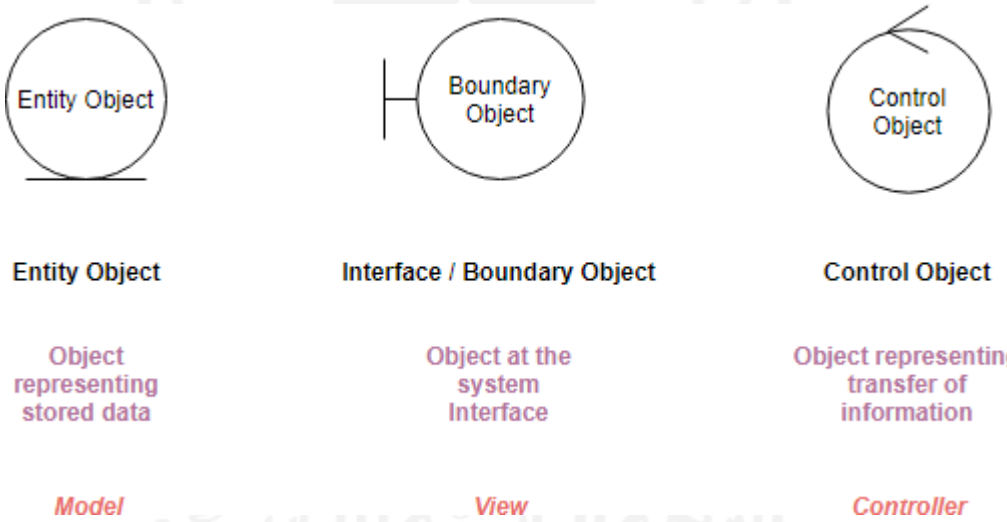
   Sistem dapat menampilkan data toko di halaman *dashboard* pegawai dan *user.* Data yang dimaksud berupa jumlah pelanggan baru dalam waktu seminggu terakhir, total pelanggan, total pegawai, grafik penjualan produk, grafik penjualan kategori, dan pengingat stok. Diperlukan 1 halaman *dashboard*, yaitu halaman *dashboard* toko.

f. TF-6 Menambah, mengedit, menampilkan, dan menonaktifkan data pelanggan

pegawai dan *user* dapat menambah, mengedit, menampilkan, dan menonaktifkan data pelanggan. Diperlukan 4 halaman, yaitu halaman tambah pelanggan, halaman daftar pelanggan, halaman detail pelanggan, dan halam edit pelanggan.

3.2.2 Rancangan Arsitektur

Setelah kebutuhan sistem terpenuhi, tahap selanjutnya adalah merancang arsitektur kelas MVC (*Model View controller*) menggunakan diagram *robustness*. Diagram *robustness* pada dasarnya adalah diagram komunikasi UML atau diagram kolaborasi yang disederhanakan dengan menggunakan simbol yang telah ditentukan (Ambler, 2004). Simbol-simbol tersebut dapat dilihat pada Gambar 3.12. Analisis diagram *robustness* membantu menjembatani kebutuhan sistem, kelas domain dan arsitektur MVC (Visual Paradigm, 2020).



Gambar 3.12 Simbol-simbol pada diagram *robustness* (Visual Paradigm, 2020)

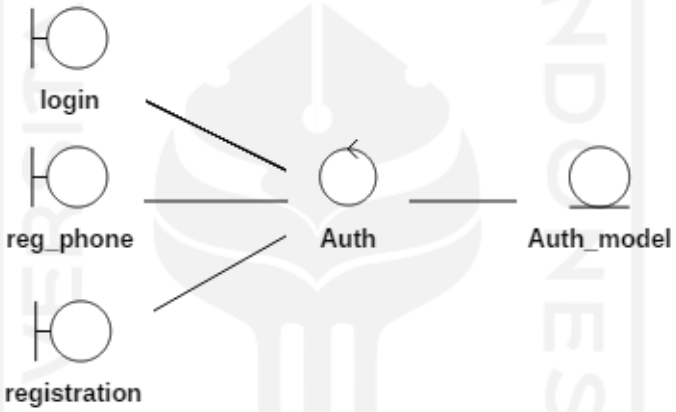
Penjelasan simbol pada gambar 3.12 adalah sebagai berikut:

- *Entity Object*, mengelola informasi yang dibutuhkan sistem untuk menyediakan fungsionalitas yang diperlukan. *Entity Object* merepresentasikan *model*
- *Interface* atau *Boundary Object*, merepresentasikan halaman antar muka atau *view*
- *Control Object*, merepresentasikan logika dari kebutuhan sistem dan mengkoordinasikan kelas lainnya, simbol ini menggambarkan kelas *controller*

Kelas MVC dipetakan berdasarkan kebutuhan fungsionalitas sistem. Berikut merupakan beberapa contoh arsitektur MVC berdasar kebutuhan fungsionalitas sistem:

1. Diagram *robustness* perancangan TF-1 dan TF-2

Skema pertama membutuhkan 5 kelas, terdiri dari 3 kelas *view*, 1 kelas *controller*, dan 1 kelas *model*. Tiga kelas *view* tersebut diberi nama login, reg_phone, dan registration. Satu kelas *controller* diberi nama Auth, *controller* Auth akan menghubungkan ketiga kelas *view* dengan kelas *model* yang diberi nama Auth_model. Gambar 3.13 menunjukan diagram *robustness* perancangan TF-1 dan TF-2. Peran dan hubungan antar kelas *model*, *view*, dan *controller* untuk TF-1 dan TF-2 dijabarkan pada Tabel 3.3.



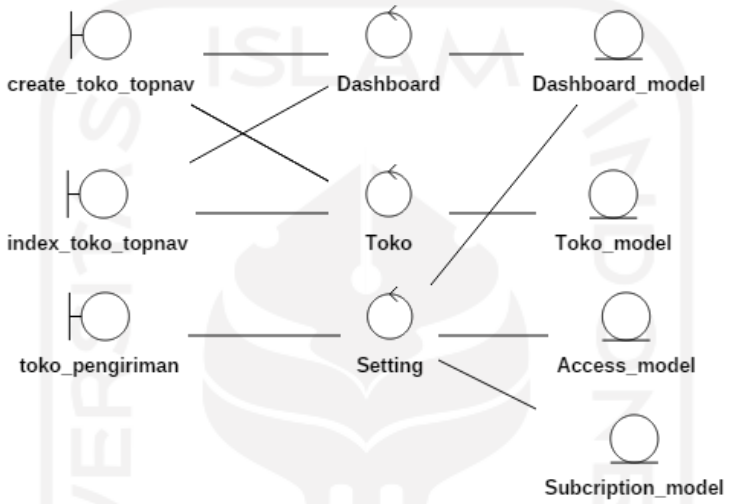
Gambar 3.13 Diagram *robustness* perancangan TF-1 dan TF-2

Tabel 3.3 Tabel *Class Responsibility* TF-1 dan TF-2

No	Kelas	Jenis	Peran/Tanggung Jawab
1	login	*view*	Sebagai halaman login
2	reg_phone	*view*	Sebagai halaman registrasi via nomor telepon
3	registration	*view*	Sebagai halaman registrasi via *email*
4	Auth	*controller*	Sebagai kelas yang akan memproses nilai masukan dari ketiga kelas *view* yaitu login, reg_phone, dan registration
5	Auth_model	*model*	Memproses dan menyiapkan data yang dibutuhkan oleh *controller* Auth

2. Diagram *robustness* perancangan TF-3

Skema kedua membutuhkan 10 kelas, terdiri dari 3 kelas *view*, 3 kelas *controller*, dan 4 kelas *model*. Tiga kelas *view* tersebut diberi nama create_toko_topnav, index_toko_topnav, dan toko_pengiriman. Tiga kelas *controller* diberi nama Dashboard, Toko dan Setting. Empat kelas *model* diberi nama Dashboard_model, Toko_model, Access_model, dan Subcription_model. Gambar 3.14 menunjukan diagram *robustness* perancangan TF-3. Tabel 3.4 menunjukan peran masing-masing kelas pada skema TF-3.



Gambar 3.14 Diagram *robustness* perancangan TF-3

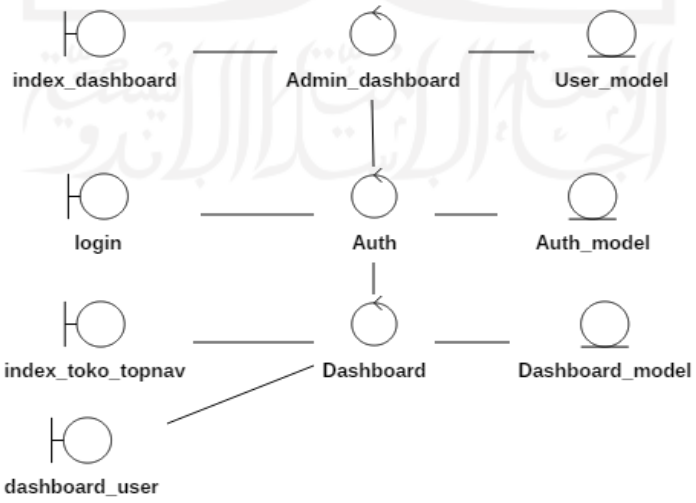
Tabel 3.4 Tabel *Class Responsibility* TF-3

No	Kelas	Jenis	Peran/Tanggung Jawab
1	create_toko_topnav	*view*	Sebagai halaman tambah toko di situs *user* dan pegawai
2	index_toko_topnav	*view*	Sebagai halaman daftar toko di situs *user* dan pegawai
3	toko_pengiriman	*view*	Sebagai halaman *setting* toko di situs *user* dan pegawai
4	Dashboard	*controller*	Menyiapkan dan memproses data toko untuk ditampilkan di halaman daftar toko
5	Toko	*controller*	Memproses *form* tambah toko dari halaman tambah toko

No	Kelas	Jenis	Peran/Tanggung Jawab
6	Setting	*controller*	Menyiapkan data dari Dashboard_model kemudian dikirimkan ke halaman *setting* toko
7	Dashboard_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Dashboard dan Setting
8	Toko_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Toko
9	Access_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Setting
10	Subcription_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Setting

3. Diagram *robustness* perancangan TF-4 dan TF-5

Skema untuk TF-4 dan TF-5 membutuhkan 10 kelas, terdiri dari 4 kelas *view,* 3 kelas *controller,* dan 3 kelas *model*. Empat kelas *view* tersebut diberi nama index_dashboard, Login, index_toko_topnav, dan dashboard_user. Tiga kelas *controller* diberi nama Dashboard, Auth dan Admin_dashboard. Tiga kelas *model* diberi nama Dashboard_model, Auth_model, dan User_model. Gambar 3.15 menunjukan diagram *robustness* perancangan TF-4 dan TF-5. Peran dan hubungan antar kelas *model, view,* dan *controller* untuk TF-4 dan TF-5 dijabarkan pada Tabel 3.5.



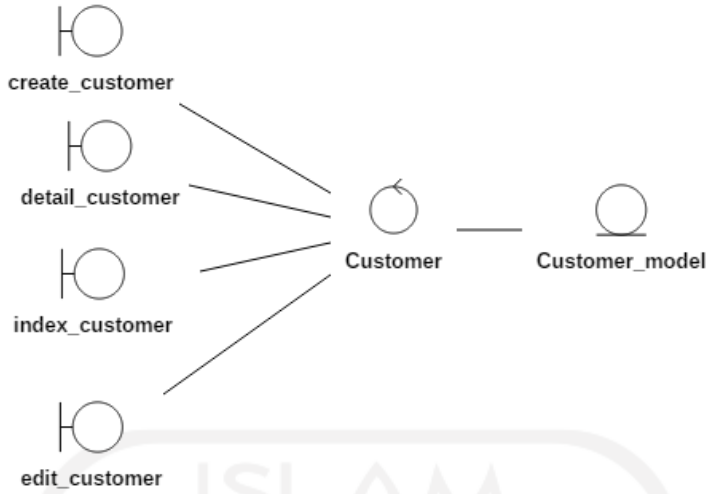
Gambar 3.15 Diagram *robustness* perancangan TF-4 dan TF-5

Tabel 3.5 Tabel *Class Responsibility* TF-4 dan TF-5

No	Kelas	Jenis	Peran/Tanggung Jawab
1	index_dashboard	*view*	Sebagai halaman *dashboard* di situs super admin
2	login	*view*	Sebagai halaman login
3	index_toko_topnav	*view*	Sebagai halaman daftar toko di situs *user* dan pegawai
4	dashboard_user	*view*	Sebagai halaman *dashboard* di situs *user* dan pegawai
5	Admin_dashboard	*controller*	Menyiapkan data untuk ditampilkan di halaman *dashboard* super admin
6	Auth	*controller*	Memproses nilai masukan dari halaman login, jika pengguna adalah super admin maka akan dialihkan ke *controller* Admin_dashboard, jika pengguna adalah *user* atau pegawai maka akan dialihkan ke *controller* Dashboard
7	Dashboard	*controller*	Menyiapkan data untuk ditampilkan di halaman *dashboard* dan halaman daftar toko di situs *user* dan pegawai.
8	User_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Admin_dashboard
9	Auth_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Auth
10	Dashboard_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Dashboard

4. Diagram *robustness* perancangan TF-6

Skema untuk TF-6 membutuhkan 6 kelas, terdiri dari 4 kelas *view,* 1 kelas *controller,* dan 1 kelas *model*. Empat kelas *view* tersebut diberi nama create_customer, detail_customer, index_customer, dan edit_customer. Satu kelas *controller* diberi nama Customer. Satu kelas *model* diberi nama Customer_model. Gambar 3.16 menunjukan diagram *robustness* perancangan TF-6. Tabel 3.6 menunjukan peran masing-masing kelas pada skema TF-6.



Gambar 3.16 Diagram robustness perancangan TF-6

Tabel 3.6 Tabel *Class Responsibility* TF-6

No	Kelas	Jenis	Peran/Tanggung Jawab
1	create_customer	*view*	Sebagai halaman tambah pelanggan
2	detail_customer	*view*	Sebagai halaman detail pelanggan
3	index_customer	*view*	Sebagai halaman daftar pelanggan
4	edit_customer	*view*	Sebagai halaman edit pelanggan
5	Customer	*controller*	Memproses nilai masukan dari halaman tambah pelanggan, daftar pelanggan, dan edit pelanggan. Menyiapkan data untuk ditampilkan di halaman daftar pelanggan, edit pelanggan, dan detail pelanggan
6	Customer_model	*model*	Menyiapkan data yang dibutuhkan oleh *controller* Customer

3.2.3 Implementasi MVC

Implementasi kelas MVC berupa penulisan kode program ke kelas yang sudah dibuat. Keterkaitan antar kelas akan dijabarkan sebagai berikut:

1 Implementasi MVC TF-1

Registrasi via nomor telepon dan *email* membutuhkan 2 kelas *view*. Dua kelas *view* tersebut diatur oleh *controller* Auth. Halaman registrasi via nomor telepon dan *email* masing-masing berisi

form yang berisi *input element* berupa data diri dan kata sandi yang akan digunakan. Contoh *form* HTML pada kelas *view* registration yang menampilkan halaman registrasi via *email* ditunjukan oleh Gambar 3.17.

```
// form akan dikirim ke fungsi registration pada controller auth

<form method="POST" action="<?= base_url('auth/registration'); ?>">
 <div class="row">
  <div class="form-group col-12">
   <label for="frist_name">Nama Lengkap</label>
   <input id="nama_user" type="text" class="form-control" name="nama_user"
autofocus value="<?= set_value('nama_user') ?>">
<?php echo form_error('nama_user', '<small class="text-danger pl-
3">','</small>'); ?>
  </div>
 </div>
.
.
.
</form>
```

Gambar 3.17 *Form* HTML kelas *view* registration

Atribut *action* akan memanggil fungsi registration di kelas *controller* Auth. Data *email* yang masuk akan dicocokan dengan data *email* di basis data (lihat Gambar 3.18), jika tidak sama maka data akan diverifikasi ke tahap berikutnya. Setelah semua data terverifikasi oleh sistem dan dinyatakan valid maka data akan disimpan ke basis data, kemudian email verifikasi akan dikirimkan ke alamat email terdaftar. Contoh fungsi di kelas *model* Auth_model dapat dilihat pada Gambar 3.19.

```
public function registration()
{
$this->form_validation->set_rules('email','Email',
'required|trim|is_unique[user.email]|valid_email',['is_unique' => 'This email
has already registered!'
]);
.
.
.
}
```

Gambar 3.18 Pencocokan data *email* di fungsi registration pada *controller* Auth

```
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 3.19 Fungsi getUserLogin di kelas *model* Auth_model

2 Implementasi MVC TF-2

Halaman *login* membutuhkan 1 kelas *view* yaitu kelas login. Kelas login dikendalikan oleh *controller* Auth. Halaman *login* berisi *form* yang berupa 2 *input element*, yaitu *input element* untuk *email* atau nomor telepon dan *input element password*. *Form* login dapat dilihat pada Gambar 3.20. *Form* ini akan dikirim ke *controller* Auth dengan nama fungsi index.

```
<form method="POST" action="<?php echo site_url('auth/') ?>" class="needs-
validation">
 <div class="form-group">
  <label for="email">Email atau No Telpon</label>
  <input id="email" type="text" class="form-control" name="email" tabindex="1"
required autofocus value="<?php echo set_value('email'); ?>">
  <?php    echo    form_error('email',    '<small    class="text-danger    pl-
3">','</small>'); ?>
  <div class="invalid-feedback">
   Please fill in your email
  </div>
 </div>
.
.
.
</form>
```

Gambar 3.20 *Form login* di kelas *view* Login

Fungsi index akan memanggil fungsi _login, di dalam fungsi _login terdapat variabel email yang akan menampung nilai dari *input element email* dari kelas *view* login. Jika nilai variabel email berupa angka 0 di awal, maka akan diganti menjadi +62 untuk kemudian dicocokan dengan format nomor telepon di dalam basis data. Gambar 3.21 menunjukan variabel email dan pengubahan nilai *string* menjadi +62 jika *substring* pertama adalah 0. Langkah berikutnya yaitu *login* berdasar hak akses akan dijabarkan pada implementasi MVC TF-4. Contoh fungsi untuk mendapat data pegawai di kelas *model* Auth_model dapat dilihat pada Gambar 3.22.

```
$email = trim($this->input->post('email'));
        if(substr($email, 0, 1)=='0'){
            $email = trim('+62'.substr($email,1,15));
        }
```

Gambar 3.21 Variabel email dan pengubahan format *string* di fungsi _login pada *controller* Auth

```
public function getPegawaiLogin($email, $password)
    {
        $this->db->select('*');
        $this->db->from('pegawai_toko');
        $this->db->join('hak_akses_pegawai',
'hak_akses_pegawai.id_hak_akses_pegawai = pegawai_toko.id_hak_akses_pegawai');
        $this->db->where(['pegawai_toko.email_pegawai_toko' => $email]);
        $this->db->where(['pegawai_toko.password_pegawai_toko'              =>
base64_encode($password)]);
        return $this->db->get();
    }
```

Gambar 3.22 Fungsi getPegawaiLogin di kelas *model* Auth_model

3 Implementasi MVC TF-3

Fitur menambah atau membuat toko baru dilakukan di kelas *view* create_toko_topnav. Terdapat *form* yang berisi *input element* mengenai data toko yang harus diisi. Gambar 3.23 menunjukan *form* buat toko baru. Atribut *action* memanggil fungsi create di kelas *controller* Toko.

```
<form      method="POST"      enctype="multipart/form-data"      action="<?=
site_url('toko/create'); ?>">
   <div class="form-group">
      <label for="nama_toko">Nama Toko</label>
      <input type="text" name="nama_toko" id="nama_toko" class="form-control"
value="<?= set_value('nama_toko'); ?>">
      <?php echo form_error('nama_toko', '<small class="text-danger pl-3">',
'</small>'); ?>
   </div>
```

Gambar 3.23 *Form* buat toko baru di kelas *view* create_toko_topnav

Pada fungsi create terdapat variabel data yang akan menampung nilai dari *input element* dari *form* buat toko baru dalam bentuk *array*. Selanjutnya data *array* akan disimpan di basis data (lihat Gambar 3.24). Contoh fungsi insert_toko di kelas *model* Toko_model dapat dilihat pada Gambar 3.25.

```
$data = [
      'nama_toko' => $this->input->post('nama_toko'),
      'sosmed' => $this->input->post('sosmed'),
      'mata_uang'=> $this->input->post('mata_uang'),
      'deskripsi_toko'=>$this->input->post('deskripsi_toko'),
      'alamat'=>$this->input->post('alamat_toko'),
      'invoice_header'=>$this->input->post('invoice_header'),
      'informasi_pembayaran'=>'-',
      'ongkir_manual'=>$ongkir_manual,
      'ongkir_vendor'=>$ongkir_vendor,
      'provinsi_toko'=>$provinsi,
      'nama_provinsi'=>$provinsi_name,
      'kabupaten_toko'=>$kabupaten,
      'nama_kabupaten'=>$kabupaten_name,
      'user_id'=>$this->session->userdata('id_user_pos')
         ];
$this->db->insert('toko',$data);
```

Gambar 3.24 Variabel data dan proses penyimpanan data toko di kelas *controller* Toko

```
public function insert_toko($data)
       {
           $this->db->insert('toko', $data);
           return $this->db->insert_id();
       }
```

Gambar 3.25 Fungsi insert_toko di kelas *model* Toko_model

4 Implementasi MVC TF-4

Halaman *login* membutuhkan 1 kelas *view* yaitu kelas login. Kelas login dikendalikan oleh *controller* Auth. Halaman *login* berisi *form* yang berupa 2 *input element*, yaitu *input element* untuk *email* atau nomor telepon dan *input element password*. *Form* login dapat dilihat pada Gambar 3.20 di implementasi MVC TF-2. *Form* ini akan dikirim ke *controller* Auth dengan nama fungsi index. Fungsi index akan memanggil fungsi _login, di dalam fungsi _login terdapat variabel email yang akan menampung nilai dari *input element email* dari kelas *view* login. Variabel tersebut akan dikirimkan ke kelas model Auth_model dengan nama fungsi getUserLogin untuk dicari apakah ada data yang dimaksud, kemudian data akan disimpan dalam bentuk *array* di dalam variabel user (lihat Gambar 3.26).

Gambar 3.27 menunjukan fungsi getUserLogin di kelas *model* Auth_model. Jika ada data yang dimaksud, maka *session* akan dibuat pada langkah selanjutnya. Data yang telah didapat mengandung informasi peran dari pengguna yang melakukan *login.* Jika peran pengguna adalah super admin, maka pengguna akan dialihkan ke *controller* Admin_dashboard. Kemudian *controller* Admin_dashboard akan menampilkan kelas *view* index_dashboard (halaman *dashboard* super admin). Jika peran pengguna adalah pegawai atau *user* maka pengguna akan dialihkan ke *controller* Dashboard, *controller* Dashboard akan menampilkan kelas *view* index_toko_topnav

(halaman daftar toko). Gambar 3.28 menampilkan kode program peralihan *controller* di fungsi _login pada kelas *controller* Auth.

```
$email = trim($this->input->post('email'));
$user = $this->auth->getUserLogin(['user.email' => $email])->row_array();
```

Gambar 3.26 Variabel email dan variable user di fungsi _login pada kelas *controller* Auth

```
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 3.27 Variabel email dan variable user di kelas *model* Auth_model

```
if ($user_phone['role_name'] != "User")
{
      redirect('admin_dashboard');
} else {
      redirect('dashboard/list_toko');
}
```

Gambar 3.28 Kode program peralihan *controller* di fungsi _login pada kelas *controller* Auth

5 Implementasi MVC TF-5

Fitur menampilkan data toko di halaman *dashboard* menggunakan 1 kelas *view* dashboard_user sebagai halaman *dashboard* pegawai dan *user*. Sebelum ke halaman *dashboard*, pengguna melakukan *login* ke toko yang diinginkan. Tombol *login* toko akan menyambungkan ke kelas *controller* Dashboard dengan nama fungsi masuk_toko (lihat Gambar 3.29). Fungsi masuk_toko akan membuat *session* id toko kemudian memanggil fungsi index. Fungsi index akan mencari data toko dengan id toko yang dimaksud di dalam basis data kemudian akan disimpan di variabel datatoko (lihat Gambar 3.30).

```
<td><a href="<?= site_url('dashboard/masuk_toko/' . $row->id_toko); ?>"
class="btn btn-primary"><i class="fas fa-location-arrow"></i> Login</a></td>
```

Gambar 3.29 Tombol *login* toko di halaman daftar toko kelas *view* index-toko-topnav

```
$datatoko = $this->db->get_where('toko', ['id_toko' =>
            $this->session->userdata('login_toko_id_pos')])->row_array();
```

Gambar 3.30 Variabel data toko di fungsi index kelas *controller* Dashboard

Setelah data ditemukan maka akan diproses ke langkah berikutnya. Data yang sudah diproses akan dikirimkan ke halaman *dashboard* kelas *view* dashboard_user. Di halaman *dashboard* akan ditampilkan data toko dari *controller* Dashboard. Contoh fungsi hitungpelangganbaru di kelas *model* Dashboard_model dapat dilihat pada Gambar 3.31.

```
 public function hitungpelangganbaru($where)
    {
        $this->db->select('customer.toko_id,(SELECT COUNT(*) FROM customer
WHERE customer.toko_id = toko.id_toko) AS jumlah_userbaru');
        $this->db->from('customer');
        $this->db->join('toko','toko.id_toko=customer.toko_id');
        $this->db->where($where);
        return $this->db->get();
     }
```

Gambar 3.31 Fungsi hitungpelangganbaru di kelas *model* Dashboard_model

6 Implementasi MVC TF-6

Menampilkan halaman detail pelanggan dipicu oleh aksi pengguna menekan tombol detail di tabel pelanggan. Id pelanggan yang dipilih akan dikirim ke kelas *controller* Customer dengan nama fungsi detail (lihat Gambar 3.32). Data pelanggan akan dicari di dalam basis data berdasar pada id pelanggan. Setelah data pelanggan ditemukan, data akan dikirim ke kelas *view* detail_customer (halaman detail pelanggan). Kelas *view* detail_customer akan menampilkan data pelanggan dari kelas *controller* Customer (lihat Gambar 3.33). Contoh fungsi datacustomerWhere di kelas *model* Customer_model dapat dilihat pada Gambar 3.34.

```
public function detail($id)
{
      cek_subcription();
      $data['page'] = 'Detail Customer';
      $data['customer'] = $this->db->get_where('customer', ['id_customer' =>
$id])->row_array();
      $this->template->load('template', 'customer/detail-customer', $data);
}
```

Gambar 3.32 Fungsi detail di kelas *controller* Customer

```
<div class="section-body">
  <h2 class="section-title">Ini,<?php  echo $customer['nama_customer'] ?>!</h2>
    <p class="section-lead"> Pelanggan Anda </p>
.
.
.
```

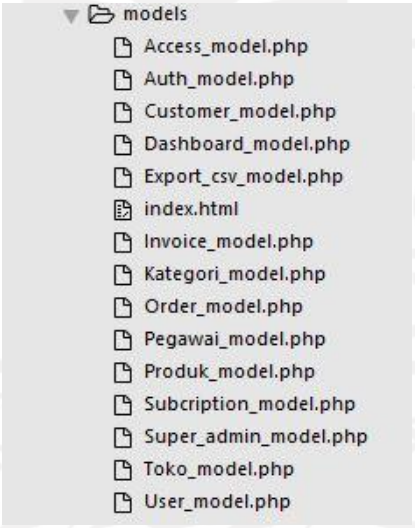
Gambar 3.33 Halaman detail pelanggan di kelas *view* detail_customer

```
public function datacustomerWhere($filter){

 $this->db->select('*');
 $this->db->from('customer');
 $this->db->join('toko','toko.id_toko = customer.toko_id');
 $this->db->where(['customer.toko_id'=>$this->session-
>userdata('login_toko_id_pos')]);
 $this->db->where('status',$filter);
 return $this->db->get();
 }
```

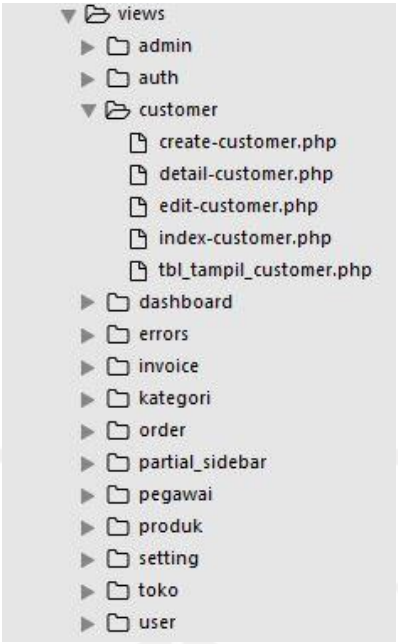
Gambar 3.34 Fungsi datacustomerWhere di kelas *model* Customer_model

Akhirnya, TPOS dikembangkan menggunakan arsitektur MVC untuk memetakan program menjadi tiga bagian utama yaitu kelas *model, view,* dan *controller*. 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 3.35.



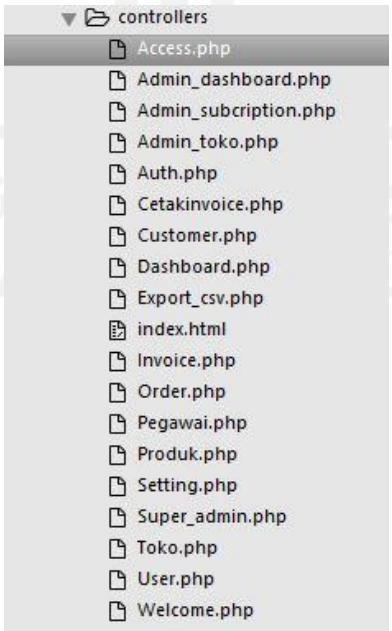
Gambar 3.35 Kelas *model*

Terdapat 14 *folder view* inti, di mana setiap *folder* memiliki beberapa kelas *view* sebagai tampilan antarmuka. Empat belas *folder* tersebut adalah admin, auth, customer, dashboard, errors, invoice, kategori, order, partial_sidebar, pegawai, produk, setting, toko, dan user. Gambar 3.36 menunjukan 14 *folder view* inti TPOS.



Gambar 3.36 Kelas *view*

TPOS memiliki 17 kelas *controller*. Tujuh belas 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, dan User. Gambar 3.37 menunjukan 17 kelas *controller* TPOS.



Gambar 3.37 Kelas *controller*

3.2.4 Pengujian

Pada tahap pengujian, *tester* menguji halaman yang telah dibuat oleh pengembang. Metode pengujian yang dilakukan adalah *black box testing. Black box testing* adalah pengujian fungsionalitas halaman tanpa perlu mengecek kode dalam sistem. Pengujian dilakukan oleh *tester* di luar lingkup pekerjaan pengembang. Hasilnya adalah semua kebutuhan sistem berhasil diimplementasikan. Sampai laporan ini dibuat, tidak ditemukan galat yang berarti.

3.2.5 Pemeliharaan

Tahap pemeliharaan merupakan proses pemeliharaan program berupa perbaikan *bug* atau peningkatan sistem TPOS setelah sistem diserahkan kepada klien. Klien akan memberikan tanggapan terhadap sistem yang sudah dikembangkan. Jika saat penggunaan aplikasi ditemukan *bug* oleh klien, maka klien akan melaporkan masalah ini ke pihak Javasign. Kemudian, *bug* akan dikonversi menjadi tugas pada antrian tugas. Jika klien menginginkan penambahan fitur baru, maka akan dilakukan peningkatan sistem. Peningkatan sistem dapat menjadi kebutuhan baru.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Implementasi MVC menggunakan CodeIgniter

Implementasi MVC menggunakan *framework* CodeIgniter merupakan hal yang paling sering dikerjakan penulis selama kegiatan magang. Implementasi ini memanfaatkan format penulisan kode program yang disediakan CodeIgniter. Hal ini memudahkan pengembang untuk menulis kode program, seperti saat memproses data berupa *array* atau objek cukup menggunakan sintak yang disediakan oleh CodeIgniter. Proses menyimpan, mengedit, dan menghapus data juga lebih mudah dilakukan karena sudah disediakan prosedur yang jelas. CodeIgniter membagi *folder* kerja sesuai fungsinya masing-masing, sehingga akan sangat membantu proses pengembangan sistem. Pemetaan kelas MVC meringankan pengembang memecah kompleksitas sistem menjadi 3 kelas, yaitu kelas *model*, kelas *view*, dan kelas *controller*. Hal ini memungkinkan pengembagan sistem secara modular lebih mudah dilakukan.

Implementasi penggunaan *framework* kurang ditekankan di matakuliah Pengembangan Aplikasi Berbasis Web semester tiga kurikulum 2016. Pembahasan mengenai pengembangan web dilakukan menggunakan bahasa pemrograman PHP *native*. Sehingga akan lebih sulit untuk melakukan manajemen sistem jika sistem yang dikerjakan cukup kompleks. Kesulitan lain yang ditemui adalah jika terdapat proyek baru, maka kita harus membuat program dari awal lagi karena tidak ada format dokumentasi yang baku pada proyek sebelumnya. Masalah lain yang ditimbulkan adalah kecenderungan pengerjaan proyek secara individu, karena setiap mahasiswa memiliki format penulisan kode program yang berbeda. Selama proses magang ditemui bahwa masalah-masalah tersebut dapat diminimalisir menggunakan *framework* yang menyediakan arsitektur MVC, seperti *framework* CodeIgniter. Pengembangan sistem secara modular menggunakan *framework* akan lebih mudah dilakukan, karena kelas telah dipetakan dengan jelas dan sudah disediakan format penulisan yang baku.

Selama pengembangan sistem TPOS menggunakan arsitektur MVC pada *framework* CodeIgniter, tim pengembang juga mengalami beberapa kendala dalam penerapannya. Beberapa di antaranya adalah kendala mengatur kelas yang semakin banyak seiring kebutuhan sistem terus meningkat, hal ini menyita waktu lumayan lama untuk sekedar mencari *file* kelas yang dimaksud. Permasalahan kedua adalah penulisan kode program yang sama pada kelas yang sama, hal ini menyebabkan kode program menjadi tidak ringkas. Kendala lain adalah sedikit ditemukannya

penjelasan kode program melalui komentar untuk *method* tertentu, sehingga akan sulit dipahami oleh pengembang baru. Adapun hambatan khusus yang dialami penulis berupa sulitnya membaca beberapa kode program, hal ini karena terdapat beberapa nama variabel menggunakan bahasa daerah yang tidak penulis pahami. Sehingga perlu dilakukan pemahaman secara mendalam untuk dapat memahami maksud baris program yang dimaksud. Faktor tersebut dapat menambah tingkat kesulitan manajemen proyek bagi para pengembang sistem TPOS berikutnya.

Manajemen proyek memiliki tahapan irisan dengan metode *waterfall* dalam pengembangan sistem TPOS. Tahap pendefinisian proyek berisi penjelasan sistem TPOS, pembagian tim, dan teknologi yang digunakan. Tahap perencanaan proyek berisi rencana penggunaan metode pengembangan, lingkup pekerjaan proyek, waktu pengerjaan proyek, situs yang digunakan, dan tabel kebutuhan fungsionalitas TPOS. Tahap Pelaksanaan proyek berisi penggunaan metode *waterfall* dan beberapa aktivitas yang sudah dikerjakan. Pemantauan proyek berisi layanan yang digunakan untuk membantu memantau jalannya proyek. Penutupan proyek berisi penyerahan sistem TPOS kepada klien.

4.2 Perbandingan Solusi dengan Alternatif Lain

Format penulisan kode program telah disediakan oleh *framework* CodeIgniter. Hal ini akan membantu pengembang dalam mengolah data dan menambah tingkat keseragaman pada sistem, jika sistem dikembangkan secara modular. Metode lain pengembangan sistem dapat dilakukan tanpa menggunakan *framework* yaitu murni menggunakan bahasa pemrograman *native* PHP. Perbandingan pengembangan sistem menggunakan *framework* dengan *native* PHP akan diperlihatkan sebagai berikut:

1 *Native* PHP

Native PHP merupakan bahasa *server side script* serba guna yang populer digunakan untuk pengembangan web. Menggunakan *native* PHP dalam pengembangan sistem web berarti membuat kode program mulai dari awal tanpa ada tambahan alat pendukung, seperti aset, konfigurasi, dan berbagai macam *library*. Umumnya setiap orang memiliki gayanya tersendiri saat menulis kode program menggunakan *native* PHP, hal ini akan membuat orang lain kesulitan memahami kode program yang sudah dibuat. Namun pengembangan web menggunakan *native* PHP memiliki beberapa manfaat di antaranya adalah:

- Cocok digunakan untuk pengembangan web skala kecil
- Pemrograman yang dibangun atas dasar pemikiran pengembang itu sendiri
- Ukuran *file* relatif lebih kecil dibandingkan dengan *framework*

Native PHP memang mudah digunakan untuk mengembangkan aplikasi web skala kecil tetapi penggunaannya tidak dianjurkan untuk pengembangan sistem dengan kompleksitas yang cukup tinggi. Terlebih pengembangan secara modular akan sulit dilakukan jika menggunakan *native* PHP, karena kelas-kelas tidak dipetakan dengan jelas.

Contoh kasus penulisan kode program yang tidak baku menggunakan native PHP adalah saat konfigurasi basis data. Konfigurasi dikembangkan dari awal, pengembang harus menuliskan nama variabel dan melakukan pengecekan kondisi jika koneksi dengan basis data gagal. Tentu setiap pengembang memiliki gaya penamaan variabel yang berbeda untuk menampung nama basis data, nama pengguna, kata kunci, dan konfigurasi lain. Gambar 4.1 menunjukan konfigurasi basis data menggunakan *native* PHP.

```
<?php

$namaServer = "localhost";
$namaPengguna = "root";
$password = "";
$nama_db = "laporan_keuangan";

$koneksi = new mysqli($namaServer, $namaPengguna, $password, $nama_db);

if ($koneksi->connect_error){
      die("Koneksi gagal : ". $koneksi->connect_error."<br>");
}
echo "<br >";
?>
```

Gambar 4.1 Konfigurasi basis data menggunakan *native* PHP

2 MVC CodeIgniter

CodeIgniter merupakan salah satu *framework* pengembangan aplikasi web. *Framework* ini menggunakan arsitektur MVC dalam penerapanya. CodeIgniter memiliki beberapa fitur yang dapat membantu pengembang mengembangkan sistem web, beberapa di antaranya adalah:

- *Model-View-Controller Based System,* yaitu menggunakan arsitektur MVC
- *Full Featured database classes,* menyediakan fitur penuh untuk kelas basis data bagi beberapa platform
- *Query Builder Database Support,* menyediakan format penulisan *query*
- *Form and Data Validation,* menyediakan validasi data dan *form*
- *Session Management,* menyediakan manajemen sesi
- *Email Sending Class,* adalah kelas yang mendukung lampiran, teks HTML *email,* dan berbagai protokol seperti sendmail, SMTP, and Mail

- *Image Manipulation Library*, menyediakan *library* manipulasi gambar seperti pemotongan, pengubahan ukuran, rotasi.
- *File Uploading Class,* yaitu kelas untuk mengunggah *file*
- *Pagination*, menyediakan fitur penomoran halaman dan pindah halaman.
- *Data Encryption*, menyediakan fitur enkripsi data
- *Error Logging*, merupakan log untuk mengatur *error*
- *Flexible URI Routing*, menyediakan fitur *routing* yang fleksibel

Masih banyak fitur yang dapat mempermudah pengembangan sistem. Dokumentasi terkait fitur-fitur unggulan lain dapat dilihat di web resmi CodeIgniter. Pemahaman terhadap fitur-fitur tersebut akan meringankan pengembang melakukan proses pengembangan sistem web yang cukup kompleks.

Berdasarkan contoh kasus konfigurasi menggunakan *native* PHP sebelumnya, konfigurasi basis data menggunakan *framework* CodeIgniter lebih mudah dilakukan karena sudah disediakan penulisan kode program yang baku. Gambar 4.2 menunjukan konfigurasi basis data menggunakan *framework* CodeIgniter. CodeIgniter sudah menyediakan nama variabel konfigurasi yang baku, dapat ditambah atau dikurangi sesuai kebutuhan.

```
$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
      'dsn' => '',
      'hostname' => 'localhost',
      'username' => 'root',
      'password' => '',
      'database' => 'tpos',
      'dbdriver' => 'mysqli',
      'dbprefix' => '',
      'pconnect' => FALSE,
      'db_debug' => (ENVIRONMENT !== 'production'),
      'cache_on' => FALSE,
      'cachedir' => '',
      'char_set' => 'utf8',
      'dbcollat' => 'utf8_general_ci',
      'swap_pre' => '',
      'encrypt' => FALSE,
      'compress' => FALSE,
      'stricton' => FALSE,
      'failover' => array(),
      'save_queries' => TRUE
);
```

Gambar 4.2 Konfigurasi basis data menggunakan *framework* CodeIgniter

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Telah berhasil dilaksanakan penerapan MVC dalam pengembangan sistem Toko Point of Sales (TPOS). Penerapan MVC pada sistem TPOS terdiri dari 5 tahapan sesuai metode pengembangan *waterfall*, yaitu analisis, perancangan arsitektur, implementasi MVC, pengujian, dan pemeliharaan. Pada tahap analisis dilakukan pengumpulan informasi terkait kebutuhan fungsionalitas sistem. Tujuannya agar proses perancangan MVC menjadi lebih mudah. Kemudian pada tahap perancangan arsitektur dilakukan pemetaan masing-masing kelas MVC menggunakan diagram *robustness*, agar gambaran kelas dapat dilihat dengan lebih jelas. Setelah membuat rancangan arsitektur, maka tahap selanjutnya adalah implementasi MVC, berupa penulisan kode program pada masing masing kelas. Tahap berikutnya adalah pengujian fitur yang telah dikerjakan, yang dilakukan oleh *tester*. Tahap akhir adalah pemeliharaan sistem, berupa pembetulan *bug* yang ditemukan oleh klien atau penambahan fitur baru setelah sistem diserahkan kepada klien. Kelebihan yang didapatkan dari penerapan arsitektur MVC menggunakan *framework* CodeIgniter adalah format penulisan program yang baku beserta pemetaan kelas yang jelas. Hasil dari pemetaan kelas yang jelas membantu memecah kompleksitas sistem menjadi elementer yang bertugas sesuai kelas dan fungsinya masing-masing. Setelah elementer kelas ditentukan, proses pengembangan sistem secara modular akan lebih mudah dilakukan.

Arsitektur MVC memecah sistem menjadi kelas *model, view,* dan *controller*. Kelas *view* bertanggung jawab sebagai tampilan antarmuka atau halaman web. Kelas *controller* bertugas mengolah data yang dimasukan dari kelas *view* dan meneruskannya ke kelas *model* jika itu diperlukan. Kelas *controller* juga bertanggung jawab menyiapkan data yang dibutuhkan dari kelas *model* atau dari kelas *controller* itu sendiri untuk diolah kemudian dikirimkan kembali ke kelas *view*. Kelas model bertugas mengambil, mengedit, menambah atau memproses data dari atau ke dalam basis data sesuai permintaan dari kelas *controller*. Manajemen proyek memiliki tahapan irisan dengan metode *waterfall* dalam pengembangan sistem TPOS.

5.2 Saran

Penerapan MVC dalam pengembangan sistem Toko Point of Sales (TPOS) yang dilakukan dalam laporan ini masih memiliki banyak kekurangan. Untuk mengoptimalkan proses pengembangan selanjutnya, maka perlu dilakukan beberapa saran sebagai berikut:

- Perlu dibuat model hirarki kelas MVC, jika kebutuhan sistem semakin kompleks.
- Penggunaan fungsi tertentu perlu dioptimalkan untuk menjaga kode program tetap ringkas.
- Perlu ditambahkan penjelasan kode program melalui komentar untuk *method* yang cukup rumit.
- Perlu dikaji jika pembagian tugas pengembang berdasarkan kelasnya, pengembang yang hanya membuat kelas *model* saja, kelas *view* saja, atau kelas *controller* saja.
- Penulisan nama variabel perlu menggunakan nama yang mudah dipahami pengembang lain, seperti menggunakan kosakata bahasa Indonesia atau bahasa Inggris.

DAFTAR PUSTAKA

Afuan, L. (2010). Pemanfaatan Framework Codeigniter dalam Pengembangan Sistem Informasi Pendataan Laporan Kerja Praktek Mahasiswa Program Studi Teknik Informatika Unsoed. *Juita*, 39-44.

Ambler, S. W. (2004). *The Object Primer (Agile Model-Driven Development with UML 2.0).* England: Cambridge University Press; 3rd edition.

Codeigniter. (2020). *Codeigniter Rocks*. Diambil kembali dari Codeigniter: https://codeigniter.com/home

Hidayat, A., & Surarso, B. (2012). Penerapan Arsitektur Model View Controller (MVC) dalam Rancangan Bangun Sistem Kuis Online Adaptif . *Seminar Nasional Teknologi Informasi dan Komunikasi 2012 (SENTIKA 2012)* , *2*, hal. 56-66. Yogyakarta.

Manno, M. (2004, September 16). *United States Patent No. US 2004/0181454 A1*.

MDN contributors. (2019, Maret 18). *MDN Web Docs Glossary* . Diambil kembali dari MDN Web Docs: https://developer.mozilla.org/en-US/docs/Glossary/MVC

Munassar, N. M., & Govardhan, A. (2010, September). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues, 7*, 94-101.

Muzakir, A. (2014). Implementasi Manajemen Perpustakaan menggunakan Framework Codeigniter (CI) Dengan Teknik Hierarchical model–view–controller (HMVC). *Seminar Nasional Sains dan Teknologi Informasi 2014 (SeNASTi 2014)*, 2355-536X.

Oxford English Dictionary. (2020). *Oxford English Dictionary.* Inggris: Oxford University Press.

Reenskaug, T., & Coplien, J. (2009, Maret 20). The DCI Architecture: A New Vision of Object-Oriented Programming. *aritma developer*.

Sani, A. S., Pradana, F., & Rusdianto, D. S. (2018). 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, 2*, 3249-3257.

Sudiroh. (2019). *Laporan Program Magang Industri.* Indramayu.

Sukandar, C. A. (2019, April 23). *Warta Ekonomi*. Dipetik Desember 18, 2019, dari https://www.wartaekonomi.co.id/read224883/apa-itu-point-of-sale.html

Sung Im, Oakton, & VA (US). (2012, November 22). *United States Paten No. US 2012/0296679 A1*.

Tahir, T. B., Rais, M., & Apriyadi, M. (2019). Aplikasi Point of Sales Menggunakan Framework Laravel. *Jurnal Informatika dan Ilmu Komputer (JIKO), 2*, 55-59.

Visual Paradigm. (2020). *A Practical Tutorial on Robustness Analysis.* Diambil kembali dari Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/robustness-analysis-tutorial/