

**LAPORAN KEGIATAN  
PRAKTIK KERJA LAPANGAN  
PT. ARISTI JASADATA**



**Disusun oleh :**

- |                               |                      |
|-------------------------------|----------------------|
| 1. Mutiara Windi Pranila      | (16/396289/SV/10502) |
| 2. Ulfa Intania Ramadhani     | (16/396301/SV/10514) |
| 3. Erlina Cahyani             | (16/401022/SV/11526) |
| 4. Muhammad Rohman Irfanuddin | (16/401045/SV/11549) |
| 5. Paska Anugrah Adil         | (16/401050/SV/11554) |

**Dosen Pembimbing :**

Muhammad Fakhurrifqi, S. Kom., M. Cs.

**PROGRAM DIPLOMA ILMU KOMPUTER DAN SISTEM INFORMASI  
SEKOLAH VOKASI UNIVERSITAS GADJAH MADA  
YOGYAKARTA  
2018**

## **KATA PENGANTAR**

Tidak ada kata yang lebih indah selain syukur yang bisa kami ucapkan kepada Tuhan Yang Maha Esa karena telah melimpahkan rahmat, hidayah, serta karunia-Nya sehingga penulis dapat melaksanakan Praktik Kerja Lapangan serta dapat menyelesaikan penulisan laporan dalam tepat waktu dan tanpa halangan yang berarti.

Laporan Praktik Kerja Lapangan ini disusun berdasarkan apa yang telah kami lakukan pada Praktik Kerja Lapangan di PT. Aristi Jasadata yang beralamat di Graha Anabatic Jl. Scientia Boulevard Kav U2 Summarecon Serpong Tangerang, Banten, Indonesia mulai tanggal 22 Juni 2018 hingga 16 Agustus 2018.

Praktik Kerja Lapangan ini merupakan salah satu syarat wajib yang harus ditempuh oleh mahasiswa Program Studi Diploma III Ilmu Komputer dan Sistem Informasi (Komsis). Selain itu, untuk menuntaskan program studi yang kami tempuh, praktik kerja lapangan ini banyak memberikan manfaat baik dari segi akademik maupun non akademik yang tidak dapat kami temukan saat berada di bangku perkuliahan.

Dalam penyusunan laporan hasil Praktik Kerja Lapangan ini kami banyak mendapatkan bantuan dari berbagai pihak baik secara langsung maupun tidak langsung. Oleh sebab itu, kami ingin mengucapkan rasa terima kasih kepada banyak pihak, terutama kepada :

1. Bapak Wikan Sakarinto selaku Direktur Sekolah Vokasi Universitas Gadjah Mada
2. Bapak Muhammad Fakhur Rifqi, S.Kom., M.Cs. selaku dosen pembimbing Praktik Kerja Lapangan yang telah banyak memberikan arahan dan masukan kepada kami dalam melaksanakan Praktik Kerja Lapangan dan juga menyelesaikan laporan praktik kerja lapangan ini.
3. Bapak Agung Coriandri selaku Kepala BaaS PT Aristi Jasadata yang telah memberi kesempatan kepada kami untuk melakukan Praktik Kerja Lapangan di PT Aristi Jasadata.

4. Mas Syaiful Ma'rrup dan Mas Rendra selaku *mentor* kerja praktik di PT Aristi Jasadata yang juga telah banyak memberikan bimbingan dengan baik secara langsung maupun tidak secara langsung sehingga dapat melaksanakan Praktik Kerja Lapangan dengan baik dan lancar.
5. Mbak Mutiara Ika Afrilia dan Mbak Amelia Rebecca selaku HR kami yang sangat baik dan selalu memberikan informasi dan arahan yang kami butuhkan selama masa PKL ini.
6. Mas Sudecanto (karyawan), Mas Patrick Nugroho Hadiwinoto (peserta internship dari ITB), Mas Kristian Aditya Nugraha (peserta internship dari UKSW) telah bergabung dalam tim kami khususnya dalam pembuatan *Tenancy Management* ini.
7. Tak lupa pula kami juga mengucapkan banyak terima kasih kepada pihak-pihak yang terkait lainnya yang telah banyak membantu baik itu untuk pelaksanaan Praktik Kerja Lapangan maupun dalam menyelesaikan laporan Praktik Kerja Lapangan ini yang tidak dapat kami sebutkan satu per satu.

Penulis menyadari bahwa kami tidaklah sempurna, begitu pula dalam penulisan laporan ini, maka apabila nantinya terdapat banyak kesalahan dan kekeliruan dalam penulisan laporan Praktik Kerja Lapangan ini, kami mohon maaf, juga kami sangat mengharapkan adanya saran dan kritik yang membangun kepada kamu. Dengan saran dan kritik itu, semoga kami bisa menjadi lebih baik lagi ke depannya.

Akhir kata, semoga laporan Praktik Kerja Lapangan ini dapat memberikan banyak manfaat bagi kita semua.

Yogyakarta, 27 Agustus 2018

Penulis

## DAFTAR ISI

HALAMAN JUDUL.....	i
KATA PENGANTAR .....	ii
DAFTAR ISI.....	iv
DAFTAR GAMBAR .....	vi
DAFTAR TABEL.....	vii
BAB I PENDAHULUAN .....	1
1.1    Profil Perusahaan.....	1
1.1.1    Visi dan Misi PT. Aristi Jasadata.....	1
1.1.2    Logo PT Aristi Jasadata .....	1
1.1.3    Bidang Usaha .....	1
1.2    Struktur Organisasi.....	3
1.3    Proses Bisnis PT. Aristi Jasadata .....	4
1.4    Latar Belakang Pemilihan .....	5
BAB II PERMASALAHAN .....	6
2.1    Kegiatan Selama PKL .....	6
2.2    Tugas Selama PKL.....	7
2.2.1    Tenancy Management .....	7
2.2.2    Rumusan Masalah .....	8
2.2.3    Batasan Masalah.....	8
2.2.4    Tujuan .....	8
2.2.5    Manfaat .....	9
BAB III METODE PENYELESAIAN MASALAH .....	10
3.1    Penjabaran Penyelesaian Masalah.....	10

3.2	Keterkaitan Dengan Mata Kuliah.....	15
BAB IV HASIL DAN PEMBAHASAN .....		18
4.1	Teknologi yang Digunakan .....	18
4.1.1	Spring Framework.....	18
4.1.2	Angular JS .....	20
4.2	Penjabaran Proyek Tenancy Management .....	22
4.2.1	Overview .....	22
4.2.2	Arsitektur Sistem.....	24
4.2.3	Hierarki Menu .....	25
4.2.4	Diagram Tabel.....	26
4.2.5	Usecase Diagram.....	26
4.2.6	Desain User Interfaces .....	27
4.2.7	Entitas.....	34
BAB V KESIMPULAN DAN SARAN.....		49
5.1	Kesimpulan.....	49
5.2	Hal-hal yang Dapat Dikembangkan di Lokasi PKL.....	49
5.3	Saran .....	50
DAFTAR PUSTAKA .....		51

## DAFTAR GAMBAR

Gambar 1. 1 Logo PT. Aristi Jasadata .....	1
Gambar 4. 1 Arsitektur <i>Tenancy Management</i> .....	24
Gambar 4. 2 Diagram Tabel <i>Tenancy Management (Email Notification)</i> .....	26
Gambar 4. 3 Usecase diagram <i>Tenancy Management (Email Notification)</i> .....	26
Gambar 4. 4 Halaman <i>List Email Sender</i> .....	27
Gambar 4. 5 Halaman <i>Add New Email Sender</i> .....	27
Gambar 4. 6 Halaman <i>Edit Email Sender</i> .....	28
Gambar 4. 7 Halaman <i>List Email Category</i> .....	29
Gambar 4. 8 Halaman <i>Add New Email Category</i> .....	29
Gambar 4. 9 Halaman <i>Edit Email Category</i> .....	30
Gambar 4. 10 Halaman <i>List Email Template</i> .....	30
Gambar 4. 11 Halaman <i>Add New Email Template</i> .....	31
Gambar 4. 12 Halaman <i>Edit Email Template</i> .....	32
Gambar 4. 13 Halaman <i>Edit Email Template (lanj.)</i> .....	32
Gambar 4. 14 Halaman <i>Compose Email</i> .....	33
Gambar 4. 15 Halaman <i>List Email History</i> .....	33
Gambar 4. 16 Halaman <i>Detail Email History</i> .....	34
Gambar 4. 17 <i>Transaction Flow Add Email Sender</i> .....	36
Gambar 4. 18 <i>Transaction Flow Edit Email Sender</i> .....	37
Gambar 4. 19 <i>Transaction flow Add Email Category</i> .....	40
Gambar 4. 20 <i>Transaction flow Edit Email Category</i> .....	41
Gambar 4. 21 <i>Transaction flow Add Email Template</i> .....	43
Gambar 4. 22 <i>Transaction flow Edit Email Template</i> .....	44
Gambar 4. 23 <i>Transaction flow Compose Email (manual)</i> .....	47
Gambar 4. 24 <i>Transaction flow Semi Automatic Mail</i> .....	48

## DAFTAR TABEL

Tabel 4. 1 Tabel contoh <i>Email Category</i> .....	39
---	----





## **BAB I**

### **PENDAHULUAN**

#### **1.1 Profil Perusahaan**

PT. Aristi Jasadata adalah sebuah perusahaan yang bergerak dalam bidang penyedia jasa *software* sebagai solusi dari permasalahan yang dihadapi oleh pelanggan atau biasa disebut *Software as a Service*. Solusi yang ditawarkan oleh PT. Aristi Jasadata adalah solusi yang berbasis pada *Cloud Solution*.

##### **1.1.1 Visi dan Misi PT. Aristi Jasadata**

###### **VISI**

“Menjadi penyedia layanan manajemen jasa nomor satu di Indonesia dan untuk menjadi yang paling diunggulkan oleh pelanggan.”

###### **MISI**

“Memberikan nilai yang tinggi serta produk dan jasa yang terbaik yang mampu meningkatkan keuntungan pelanggan dan mampu berjalan bersama dengan segala *stakeholder*.”

##### **1.1.2 Logo PT Aristi Jasadata**



Gambar 1. 1 Logo PT. Aristi Jasadata

##### **1.1.3 Bidang Usaha**

Berikut adalah beberapa bidang jasa atau solusi yang ditawarkan oleh PT. Aristi Jasadata sebagai solusi permasalahan pelanggan :

###### **1.1.3.1 Facility Management Service (FMS)**

Aristi Jasadata menyediakan layanan *support* untuk *monitoring*, *back up*, dan memperbaiki perangkat lunak dari

berbagai *troubleshoot* yang biasanya ditemukan pada *server*, jaringan, dan pusat aplikasi. Aristi menawarkan proteksi dan manajemen terhadap sistem *client* dalam 24/7. Sebagai konsekuensinya, diharapkan *client* mampu lebih fokus dalam pencapaian laba dan investasi dari bisnis yang mereka jalankan.

Aristi menyediakan layanan *testing* yang intensif dengan beberapa langkah dalam mengatur *test services*, paket pengujian solusi, *functional testing*, *non-functional testing*, dan *load testing*. Aristi menyediakan aplikasi yang menjamin harga efektifitas biaya yang cocok bagi semua ukuran bisnis.

Layanan Aristi FMS di antaranya :

- *Data Center Operation and Network Operation Center 24/7*
- *Managed Service Production Support*
- *Managed Service for Application Performance*
- *Managed Service Data center and DRC IBM iSeries and MIMIX 24/7*
- *Managed Service AIX Platform*

#### **1.1.3.2 Infrastructure as a Service**

Perusahaan Aristi menawarkan solusi kepada pelanggannya untuk mengembangkan efisiensi IT, kecepatan dan reliabilitasnya. Solusi *cloud* yang ditawarkan akan memungkinkan sistem milik *client* untuk menjadi lebih fleksible dalam menghadapi kebutuhan organisasi *client*. Aristi menawarkan segala hal yang diperlukan untuk membangun, mengoperasikan, menyewakan, dan mengatur pusat data pelanggan dengan pelayanan yang mengagumkan. Aristi Jasadata menawarkan sejumlah solusi yang dapat dikustomisasi berdasarkan permintaan *client*.

#### **1.1.3.3 Software as a Service**

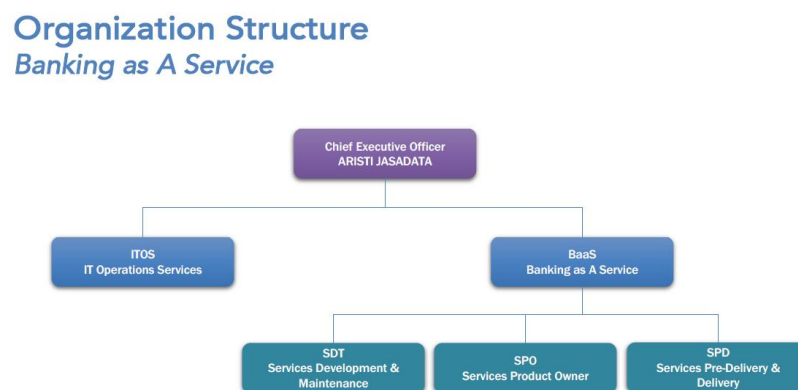
Teknologi Informasi merupakan suatu hal yang esensial dalam pengoperasian suatu bisnis atau bidang usaha. Kualitas yang baik dari suatu layanan teknologi informasi adalah suatu hal yang penting dalam

pertumbuhan suatu perusahaan. Akan tetapi, infrastuktur IT menghadapi batasan dalam hal Sumber Daya Manusia dan Kompetensi.

Ada beberapa produk yang ditawarkan oleh PT. Aristi Jasadata di antaranya, aplikasi *mobile* sebagai solusi di bidang perbankan, aplikasi untuk *e-learning* yang mempelajari berbagai bidang teknologi, dan juga aplikasi yang mampu membantu mengatur aset suatu perusahaan.

## 1.2 Struktur Organisasi

Perusahaan secara keseluruhan dipimpin oleh *Chief Executiver Officer (CEO)*, Ibu Agustini serta Direktur dan Kepala BaaS, yaitu Bapak Agung Coriandri. Dalam kesatuan struktur, BaaS kemudian terbagi menjadi tiga bagian yang lebih detail. BaaS merupakan bagian dari PT. Aristi di mana penulis melangsungkan kerja praktik. Berikut adalah struktur organisasi PT. Aristi Jasadata.



**Gambar 2.1** Struktur Organisasi PT. Aristi Jasadata

Dalam struktur organisasi tersebut BaaS dikepalai oleh seorang *Head of Banking as a Service* yang membawahi tiga bagian lainnya. Bagian yang pertama adalah *Service Development & Maintenance*, di dalamnya terbagi lagi menjadi bagian *Service Development* dan *Application Support*. Bagian kedua adalah bagian *Services Product Owner*, di mana di dalamnya berisi tentang *Product & Service Management* serta *Product&Services Business Research*. Bagian ketiga ialah bagian *Services Pre-Delivery&Delivery (SPD)*. SPD kemudian terbagi menjadi tiga bagian lain di antaranya *Buss Support*,

*Implementation, Pre-Delivery.* Penjelasan mengenai struktur organisasi perusahaan dan proses bisnis didapatkan berdasarkan hasil wawancara dengan direktur PT. Aristi Jasadata.

### **1.3 Proses Bisnis PT. Aristi Jasadata**

Aristi Jasadata merupakan perusahaan yang bergerak dalam bidang layanan perangkat lunak atau biasa disebut *Software as a Service (SaaS)*. Dalam implementasinya, Aristi Jasadata menawarkan solusi-solusi di bidang IT untuk *client* berbasis *cloud*. Ada berbagai macam solusi *cloud* yang ditawarkan diantaranya *Application Assurance, Office Automation*, dan lain-lain.

Dalam implementasinya, pelanggan atau *client* menggunakan aplikasi yang ditawarkan oleh perusahaan dengan model sewa atau *subscription*. Adapun proses bisnis dari PT Aristi Jasadata dijelaskan sebagai berikut :

- a. *Sales* akan melakukan penawaran produk-produk PT. Aristi Jasadata terhadap calon *client*.
- b. Dalam melakukan proses *engagement* dengan *client*, perusahaan melalui *sales* menawarkan produk melalui dokumen penawaran dalam bentuk proposal.
- c. Jika calon *client* menyetujui maka perusahaan mempersiapkan dokumen kontrak sesuai dengan PO yang ada dan status calon *client* berubah menjadi *tenant*.
- d. Setelah dokumen kontrak disetujui dan ditandatangani oleh kedua belah pihak, maka perusahaan mempersiapkan untuk pengimplementasian produk.
- e. Tahap implementasi adalah tahap di mana perusahaan membantu *tenant* untuk mulai menggunakan dan mengelola produk perangkat lunak yang mereka sewa.
- f. *Tenant* akan mengirimkan Berita Acara kepada perusahaan sebagai pemberitahuan kepada perusahaan mengenai pengimplementasian produk perangkat lunak mereka.

- g. Setelah proses implementasi sukses, maka tenant akan mengirimkan Berita Acara bahwa mereka akan Go Live, atau benar-benar menggunakan produk perangkat lunak yang mereka sewa.
- h. Pihak perusahaan akan melakukan verifikasi perjanjian kontrak dan akan mengeluarkan besaran faktur yang harus dibayar oleh *tenant* dalam bentuk *invoice* tiap bulan.
- i. *Tenant* wajib melakukan pembayaran terhadap perusahaan sesuai dengan faktur atau *invoice* yang mereka terima.

Proses bisnis di atas secara garis besar menjelaskan proses pembuatan pesanan dari calon *client*. Aplikasi *Mobile Perbankan* merupakan salah satu produk yang ditawarkan oleh PT. Aristi Jasadata dalam menyelesaikan permasalahan yang dihadapi oleh pelanggannya.

#### **1.4 Latar Belakang Pemilihan**

Ada beberapa alasan kenapa kami memilih PT. Aristi Jasadata menjadi lokasi PKL kami. Yang pertama adalah karena rasa penasaran kami terhadap perusahaan ini. Awalnya kami sama sekali tidak ingin memilih lokasi PKL di luar Jogja, namun setelah mendapatkan sekilas informasi dari dosen kami mengenai perusahaan ini, kami mulai tertarik.

Yang kedua adalah karena profil perusahaan yang cukup bagus dan strategis untuk menjadi tempat mahasiswa mengembangkan diri, apalagi perusahaan ini berada di sekitar kawasan metropolitan (Jabodetabek) yang mana banyak perusahaan besar sejenis bersaing.

Yang ketiga adalah karena fasilitas yang ditawarkan cukup menarik bagi kami sebagai mahasiswa magang.

## BAB II

### PERMASALAHAN

#### 2.1 Kegiatan Selama PKL

Selama kegiatan Praktik Kerja Lapangan di PT Aristi Jasadata, kami melakukan hal-hal sesuai arahan dari pembimbing dan *supervisor* kami yaitu mas Syaiful Ma'rrup. Adapun garis besar kegiatan-kegiatan yang kami lakukan di PT Aristi Jasadata antara lain :

1. Membuat akun Larise (*digital canteen and mart*) , email perusahaan, dan GitAtic untuk mempermudah koordinasi dengan pihak HR.
2. Membuat akun Trello dengan menggunakan email perusahaan yang telah diberikan kepada kami. Kegiatan ini dilakukan oleh seluruh kelompok untuk memudahkan kami dalam berkomunikasi dan melihat perkembangan setiap *sprint* yang kami lakukan.
3. Mendapatkan materi dasar tentang teknologi-teknologi yang digunakan oleh PT Aristi Jasadata untuk mengembangkan aplikasi di Anabatic Group. Materi yang dimaksudkan antara lain *Frontend Development* (AngularJS), *Backend Development* (Spring Framework dan Java EE).
4. Melakukan implementasi dari *Scrum Methodology* yang meliputi :
  - a. *Sprint planning* ketika akan memulai *sprint*. Dimana di proses ini kami berdiskusi mengenai bisnis proses yang akan kami kerjakan nantinya.
  - b. *Sprint review* dilaksanakan ketika masa waktu *sprint* sudah berakhir. Bertujuan untuk membahas *progress* dari *backlog* yang sudah dikerjakan.
  - c. *Sprint retro* dilaksanakan ketika masa dari *sprint* sudah berakhir. Bertujuan untuk membahas kinerja dan cara kerja dari tim yang sedang mengembangkan sistem ataupun aplikasi. *Result* yang diharapkan adalah cara kerja mana yang akan menjadi cara kerja paling efektif dan dapat meningkatkan produktivitas tim.
5. Mempelajari *tools* yang telah dipaparkan sebelumnya. Untuk *backend* mempelajari Spring Framework dengan IDE Eclipse JEE, Postman untuk memeriksa API yang telah dibuat, serta DBeaver untuk mengelola *database* dari sistem yang dibuat. Untuk *frontend* mempelajari penggunaan AngularJS

yang dipadukan dengan HTML5, Sublime sebagai *text editor*, Node.js untuk menjalankan AngularJS, Postman untuk memeriksa API yang telah dibuat oleh *backend*, Eclipse JEE untuk menjalankan *server*, DBeaver untuk memeriksa data di dalam *database*.

6. Mengerjakan modul *Email Notification* sebagai bagian dari *Tenancy Management*.
7. Melakukan *deploy* ke *server* apache Tomcat.

## 2.2 Tugas Selama PKL

### 2.2.1 Tenancy Management

Tugas yang diberikan selama PKL yaitu Tenancy Management (TM). Aplikasi ini dikembangkan dengan mengadopsi *multi tenant software architecture* dengan tujuan utama untuk memudahkan pembuatan *database* baru untuk masing-masing tenant yang telah terdaftar secara otomatis dan tanpa perlu melakukan penyusunan *database* dari awal. Selain itu, dapat digunakan untuk melakukan kontrol terhadap entitas dan parameter yang ada serta mempermudah komunikasi dengan masing-masing *tenant*. Aplikasi ini diharapkan dapat memberikan notifikasi secara otomatis kepada *tenant* mengenai informasi keadaan, tagihan, *reminder* akan *limit* nasabah hingga pada penawaran produk.

#### 2.2.1.1 Latar Belakang Permasalahan

PT. Aristi Jasadata menerapkan konsep *multi tenancy* di dalam menjalankan bisnisnya. *Multitenancy* merupakan konsep dimana sebuah aplikasi dapat digunakan oleh banyak *tenant*, dimana setiap *tenant* mendapatkan perlakuan yang sama. Namun, setiap *template* yang dibangun menyesuaikan dengan atribut yang dimiliki oleh instansi pemilik *tenant*, misal seperti judul dan nama perusahaan, logo, alamat, dan sebagainya.

Di dalam penerapannya untuk membuat sebuah *tenant* baru, admin harus mempersiapkan banyak hal. Selain itu,

admin harus melakukan pengecekan secara berkala dan memberi notifikasi kepada setiap *tenant*. Belum lagi ketika ada suatu kondisi dimana admin harus berkomunikasi secara massal kepada *tenant-tenant* yang terdaftar, Hal tersebut akan menyulitkan dan membuat tidak efektif pekerjaan admin.

Untuk itu, dibuatlah sebuah fitur *Email Notification* yang digunakan untuk melakukan komunikasi antara admin dan salah satu *tenant* maupun semua *tenant-tenant* yang terdaftar baik secara manual maupun otomatis.

#### **2.2.2 Rumusan Masalah**

1. Bagaimana membuat sistem *email notification* yang dapat masuk ke email *tenants*?
2. Bagaimana melakukan pemberitahuan ke *tenants* mengenai tagihan ataupun kendala teknis lainnya?
3. Bagaimana membuat desain *user interface* Tenancy Management dengan meminimalisir penggunaan sistem yang manual?
4. Bagaimana mengintegrasikan email yang menggunakan protokol SMTP ke sistem lokal untuk melakukan pengiriman email?

#### **2.2.3 Batasan Masalah**

1. *Tenancy Management* ini dibuat dengan menggunakan Spring Framework dan Java EE sebagai *backend* sekaligus penyedia API, dan *frontend* menggunakan AngularJS.
2. Sistem yang dibuat adalah fitur *email notification* yang akan memberitahu *tenants* ketika ada pemberitahuan dari admin.
3. Admin harus menginputkan *master parameter* terlebih dahulu untuk dapat melakukan pengiriman email dengan sempurna.

#### **2.2.4 Tujuan**

1. Memudahkan admin untuk menghubungi *tenants* jika terjadi hal-hal yang penting, seperti tagihan, kendala teknis, dll.



2. Memudahkan mengirim email notification ke beberapa *tenants* sekaligus dengan menggunakan template yang telah dibuat.
3. Memudahkan dalam mengirim email yang dibuat secara manual (*manual compose*) sehingga tidak perlu membuka gmail, yahoo, atau sejenisnya untuk melakukan pengiriman langsung di sistem itu juga.

#### **2.2.5 Manfaat**

Dengan dibuatnya sistem *Tenancy Management* ini, sistem pemberitahuan ke *tenants* dapat lebih mudah dimanajemen dan lebih mudah. Admin/marketing staff juga dengan mudah mengetahui jika ada tunggakan pembayaran atau jumlah pengguna yang mencapai limit dari aplikasi yang disewa

### **BAB III**

#### **METODE PENYELESAIAN MASALAH**

PT. Aristi Jasadata menggunakan metode *Scrum* untuk pengembangannya. Karena metode ini berfokus pada kecepatan dan kemampuan beradaptasi untuk proyek yang berbeda, sehingga dapat meningkatkan produktivitas dalam menyelesaikan suatu proyek. Metode *Scrum* yang diterapkan di PT. Aristi Jasadata biasanya berumur 2 minggu dan sudah termasuk *planning* dan *testing* yang kemudian disebut sebagai *sprint*. Dalam 1 *sprint*, pembagian pengerjaan adalah berdasarkan bobot pengerjaan atau biasa disebut *fibonacci* yang telah ditentukan dalam *sprint planning*.

#### **3.1 Penjabaran Penyelesaian Masalah**

##### **Minggu Pertama**

Pada minggu pertama, kami *briefing* terkait aplikasi yang akan dikembangkan, alur bisnis PT. Aristi Jasadata, apa yang akan kita kerjakan, serta pembagian *role*. Kami mengerjakan 1 proyek yang sama, yaitu Tenancy Management. Untuk pembagian *role* dibagi menjadi 2 yaitu *Backend Developer* dan *Frontend Developer*. Adapun, pembagiannya adalah : Paska Anugrah Adil sebagai *Backend Developer*, Erlina Cahyani sebagai *Backend Developer*, Muhammad Rohman Irfanuddin sebagai *Backend Developer*, Mutiara Windi Pranila sebagai *Frontend Developer*, dan Ulfa Intania R. sebagai *Frontend Developer*.

Deskripsi kerja dari *Backend Developer* adalah membuat API yang akan digunakan oleh *Frontend* untuk mengakses data ke *database* beserta *logic* sesuai kebutuhan. *Backend Developer* juga berurusan dengan *database* yang menyesuaikan kebutuhan.

Sedangkan deskripsi kerja dari *Frontend Developer* adalah membuat tampilan yang akan berhubungan langsung dengan *user*, dimana data-data yang ada akan diolah dalam bentuk *json* yang kemudian akan mengakses url dari *backend* sesuai *HTTP Request* yang diperlukan.

##### **Minggu Kedua**

Di Aristi, begitu kami menyebut PT. Aristi Jasadata, masih belajar *framework Java Spring* untuk *backend* dan *AngularJS* untuk *frontend*, yang mana *framework* itulah yang akan digunakan dalam mengerjakan proyek nantinya. Biasanya yang magang di Aristi akan diberikan waktu untuk beradaptasi dulu dan belajar tentang *framework* yang digunakan oleh pihak Aristi.

Di minggu kedua ini, kami memang masih terus belajar konsep dan cara kerja dari *Java Spring* maupun *AngularJS* sendiri. Mencari referensi, bertanya, dan memperhatikan karyawan lain bekerja yang juga sebagai *programmer*.

### **Minggu Ketiga**

Di minggu ketiga ini kami mulai paham *tools* dan *framework* yang akan kami gunakan. Seperti yang telah kami sampaikan, bahwa *programmer* dibagi menjadi sisi *backend* dan *frontend*. Dari *backend* belajar *framework Spring*. *Framework Spring* adalah *framework open source* berbasis Java yang menyediakan infrastruktur yang komprehensif dalam mengembangkan aplikasi Java dengan mudah dan cepat. Dukungan inti dari *framework* ini lebih kepada aplikasi web, manajemen transaksi, akses data, messaging, pengujian dan lain-lain. Spring bisa mengembangkan aplikasi web berbasis MVC dan *web service framework RESTful*. Dukungan JDBC, JPA, dan JMS juga telah tersedia. *Framework* ini dengan requirement minimal, JDK 6+ untuk versi Spring 4.x dan JDK 5+ untuk Spring 3.x. Bagian penting dari Spring antara lain *Entity/Model*, *Mapper*, *Service*, *Controller*. *Entity* berisikan *setter* dan *getter* dari objek. *Mapper* berhubungan dengan *database* dan pengolahan data. *Service* memuat *logic* dari bagian sistem yang akan dibangun. Kemudian, *controller* merupakan proses *routing uri*. Sedangkan untuk *developer frontend* menggunakan *framework AngularJS*. *AngularJS* adalah *frontend framework* untuk *javascript* yang dikembangkan oleh Google. Dengan fitur-fitur *powerful* dari *AngularJS*, proses *development* bisa menjadi jauh lebih singkat.

## Minggu Keempat

Di minggu ini kami mengikuti *sprint review* dan *sprint retro* bersama semua karyawan lantai 4 alias karyawan Aristi. Disini dibahas hasil yang telah didapat selama 1 *sprint* terakhir. Mulai dari hal teknis maupun non-teknis. Setiap BA (*Business Analyst*) menyampaikan capaian yang didapat oleh timnya masing-masing. Hasilnya kemudian diringkas dan dicatat sebagai tolok ukur *backlog* dari *sprint* berikutnya. Selain itu, disini juga membahas kinerja sebagai evaluasi tim. Para karyawan menuliskan saran dan masukan dalam *sticky note* yang kemudian ditempel di papan tulis untuk selanjutnya didiskusikan bersama, apa saja yang harus dipertahankan, dihilangkan, dan yang akan dimulai. Lalu, setelah diskusi berakhir, diambil keputusan yang telah disepakati bersama dan akan dijalankan di *sprint* berikutnya.

Pada minggu ini kami juga sudah mulai membuat *planning* untuk *sprint* kami yang pertama. Dari sisi *backend* sudah membuat rancangan *database*, mulai dari rancangan tabel, atribut, serta relasinya. Dari sisi *frontend* sudah menyiapkan *mockup* untuk nantinya diimplementasikan sebagai tampilan yang akan dibuat dalam *project* nantinya.

Sebelum memulai *sprint*, terlebih dahulu kami berdiskusi bersama *mentor* kami, yaitu Mas Ma'rrup, begitu kami biasa menyapanya. Dalam diskusi ini, kami mempresentasikan apa saja yang telah kami rancang kepada *mentor* kami. Kemudian *mentor* kami memberikan tambahan-tambahan serta masukan-masukan terhadap apa yang telah kami presentasikan.

Setelah melakukan diskusi, di hari berikutnya kami memulai *sprint* kami yang pertama. Kami memulai *sprint* pada hari Kamis dan berakhir 2 minggu kemudian. Dari *front-end* sudah membuat semua *view* (desain *user interface*) dari keseluruhan sistem, namun belum ada fungsi yang ditambahkan. Hari berikutnya, kami sudah menambahkan fungsi *add new email sender*, menampilkan *list email sender*, serta menampilkan *list email*

*history*. Dari backend menyiapkan API yang dibutuhkan oleh sisi *frontend* serta API untuk CRUD (*create, read, update, delete*). API yang dibuat antara lain untuk *email sender, email history, email template category*, serta *email template*.

### **Minggu Kelima**

Minggu ini kami meneruskan *task* yang belum selesai kami kerjakan. Banyak yang kami kerjakan di minggu ini. Minggu kelima merupakan minggu dimana *core task* dikerjakan. Mulai dari halaman *email sender*, untuk *add new email sender, delete email sender, list email sender*, dan *edit email sender* sudah hampir *fix*. Pada halaman ini masih kurang untuk sisi validasi, baik saat *add* maupun saat *edit*. Selanjutnya untuk halaman *email category*, baik dari fungsi *add new email category, delete email category, edit email category*, serta *list email category* sudah dianggap *fix* untuk sementara.

Selain kedua halaman tersebut, adapun halaman yang hampir jadi lainnya yakni *email template, compose email, email history*. Pada halaman *email template*, fungsi *add, delete*, dan menampilkan *list email template* sudah dikerjakan dan hampir *fix*. Di bagian *core* fungsi modul *Email Notification* yaitu *manual compose* dan *semi-automatic* juga sudah 70% berjalan. Pada minggu ini kami sudah bisa melakukan transaksi *email* (mengirim *email* melalui *front-end*). Selain itu, untuk pengiriman dengan menggunakan *email template* juga sudah bisa digunakan. Kemudian untuk halaman *Email History* sudah ditambahkan *pagination* serta halaman untuk melihat *detail* dari *email* yang telah dikirim.

Pada minggu kelima ini, API yang sudah selesai adalah *email sender* untuk *insert, get all(list), get by id, delete, edit*. Selain *email sender*, API untuk *email history* juga sudah selesai untuk *insert, get all(list), get by id*. Untuk *email template category* juga sudah selesai untuk *insert, get all(list), get by id, delete, edit*. Selain API, kami juga mengkonfigurasi email agar dapat digunakan untuk mengirim email. Dalam email terdapat 2 jenis

keamanan yaitu TLS dan SSL. Pihak Aristi dalam *emailing* menggunakan enkripsi TLS. Tetapi kami tetap membuat konfigurasi baik untuk TLS maupun SSL, agar nanti dapat menggunakan *email sender* baik dengan enkripsi TLS atau SSL.

### **Minggu Keenam**

Menurut hitungan, minggu depan seharusnya *sprint* kami akan berakhir dan itu artinya *backlog* yang sudah kami *list* pada saat *planning* harusnya sudah selesai. Nah, di minggu ini kami berusaha menyelesaikan beberapa halaman yang belum *fix* maupun yang belum tersentuh sama sekali. Dari *validation*, *typo*, perubahan API yang digunakan, dan *minor bug* lainnya.

Dari sisi *front-end* sudah menambahkan fungsi untuk *edit email template* dan menampilkan *preview* dari *email template* yang telah dibuat. Lalu mengimplementasikan *selector* untuk *field Recipient* yang ada pada halaman *Compose Email*, *form Add New Email Template*, dan *Edit Email Template*. Untuk validasi-validasi yang belum ditambahkan juga sudah mulai diselesaikan pada minggu ini, namun validasi yang digunakan masih yang berasal dari *front-end*.

Dari sisi *backend* mulai lanjut membuat API untuk *email template* untuk *insert*, *get all(list)*, *get by id*, *delete*, *edit*. Perbaikan dan beberapa penambahan untuk API dari *email sender*, *email template category*, *email history*. Kemudian, *testing send email* dengan konfigurasi *emailing* yang telah dibuat menggunakan *email sender*.

### **Minggu Ketujuh**

Fungsionalitas dari fitur *Email Notification* ini sudah teratasi dan selesai pada minggu kemarin (minggu berakhirnya *sprint* 1), baik dari *front-end* maupun *back-end*. Minggu ini kami manfaatkan untuk melakukan *testing and checking* serta melakukan *deploy* ke *server*.

Setelah pengecekan dilakukan, ternyata ada banyak *bug* yang kami temukan dan akhirnya kami menambah serta memperbaiki *bug* yang ada.

Tidak hanya dari sisi API yang diganti, namun juga dari *view* masing-masing halaman hingga pemunculan *error message* dari *backend*. Disini kami (*front-end*), merubah *label-label* yang dianggap kurang sesuai serta menambahkan *error message* menggunakan *API backend*. Pada kesempatan ini kami juga mencoba menghapus *tag* html yang masih tercantum ketika *email* dikirimkan. Hal ini disebabkan oleh *plugin* yang kami gunakan. Secara *default*, *tinymce* akan memunculkan *tag* html pada setiap *output* yang diberikan. Setelah mencari informasi dari berbagai sumber, akhirnya kami menemukan jalan untuk menghapus *tag* tersebut.

Karena agak *selo*, kami juga melakukan *cleaning code* untuk merapikan *code* yang sebelumnya masih berantakan.

### **Minggu Kedelapan**

Di minggu ke delapan, setelah *sprint review* kemarin kami berencana akan menjalankan 1 *sprint* lagi disini untuk membuat *emailing* yang bersifat *automatic*. Tetapi karena waktu disini tidak mencukupi untuk dilakukan *sprint*. Maka di minggu kedelapan, kami melakukan *merging* dengan tim lain (tim *tenant & customer*) di proyek yang sama. Tetapi karena tim mereka dipindah di proyek lain maka fitur itu kami yang menggabungkan beserta *checking*. Setelah itu kami melakukan *deploy* ke server dan melakukan *final checking*.

### **3.2 Keterkaitan Dengan Mata Kuliah**

Dalam Praktik Kerja Lapangan ini, kami menemukan beberapa keterkaitan dengan mata kuliah yang telah diajarkan selama 4 semester terakhir. Adapun mata kuliah yang terkait dengan pekerjaan yang kami lakukan disini adalah :

#### **a. Pemrograman Berorientasi Objek**

*Backend* dari proyek *Tenancy Management* menggunakan *Framework Spring* yang dikombinasikan dengan Java EE. Konsep-konsep PBO seperti *inheritance* (turunan/pewarisan), *abstraction*

(pengabstrakan), *encapsulation* (pembungkusan), dan *polimorfism* diterapkan untuk keperluan olah data.

b. KL Web I dan Web Lanjut

Proyek *Tenancy Management* adalah proyek yang berbasis web. Yang menggunakan *Framework* Angular JS untuk *Frontend* dan *Spring Framework* untuk *Backend*. Kemampuan yang diajarkan di KL Web I, seperti HTML, CSS, Javascript, AJAX diimplementasikan dalam proyek ini. Kemudian untuk kompetensi yang diajarkan pada KL Web Lanjut seperti penggunaan *framework* untuk manajemen proyek, dan penggunaan javascript di bagian *frontend*.

c. Rekayasa Perangkat Lunak

Pengembangan dari proyek *Tenancy Management* selalu berkaitan dengan mata kuliah Rekayasa Perangkat Lunak (RPL), mulai dari metode pengerjaan yang digunakan, pembuatan ERD dan diagram-diagram lain untuk keperluan *planning* yang kemudian kami bahas bersama *project manager* kami.

d. Sistem Informasi

Penerapan dari mata kuliah Sistem Informasi seperti testing, metode pengembangan yang digunakan, perencanaan, dll diimplementasikan dalam pengembangan proyek *Tenancy Management* ini.

e. Jaringan Komputer

Tidak hanya dari sisi *code* saja kami belajar. Kami juga dituntut untuk bisa *mendevlop* hasil proyek kami ke *server* langsung. *Server* yang digunakan adalah Linux, kami belajar pula bagaimana *server* itu berjalan bagaimana kami mengaksesnya.

f. Pengelolaan Instalasi Komputer



Saat minggu-minggu pertama kami melakukan instalasi beberapa aplikasi yang belum ada di laptop kami, seperti Postman, PostgreSQL, DBeaver, dan beberapa aplikasi untuk *deploy* ke *server*. Instalasi aplikasi ini adalah salah satu implementasi dari mata kuliah Pengelolaan Instalasi Komputer (PIK).

g. Basis Data

Keahlian yang diajarkan sewaktu kuliah seperti Praktikum Basis Data 1 yang mencakup *select*, *update*, *delete*, *insert*, *relational database*, dll juga kami implementasikan dalam *database* di proyek *Tenancy Management*. Kemudian, Praktikum Basis Data 2, seperti *trigger*, *rollback*, dll tetapi dikombinasikan dengan *backend* java untuk keperluan *rollback*, *trigger*. Lalu di PKL ini kami mendapat kompetensi baru untuk menggunakan PostgreSQL untuk mengelola *database* proyek ini. Berbeda dengan yang kami gunakan di perkuliahan yakni MySQL.

h. Technopreneurship

Sewaktu *briefing* ataupun *sprint review* (yang kami lakukan setiap 1 *sprint* selesai) terkadang Pak Agung (*Product Owner*) juga memberikan cerita bagaimana produk ini akan dipasarkan dan bagaimana mendapat keuntungan dari penjualan.

i. Algoritma dan Struktur Data

Baik dari sisi *frontend* dan *backend* terdapat penerapan mata kuliah Algoritma dan Struktur Data, baik dari sisi algoritma, struktur data, ataupun tipe data yang digunakan.

## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Teknologi yang Digunakan

##### 4.1.1 Spring Framework

*Spring framework* merupakan *framework open source* berbasis Java yang menyediakan infrastruktur yang komprehensif dalam mengembangkan aplikasi Java dengan mudah dan cepat. Spring pertama kali ditulis dan dirilis oleh **Rod Johnson** dengan lisensi Apache 2.0 pada bulan Juni 2003. Spring akan membantu pengembangan aplikasi dengan *build* yang sederhana, *portable*, cepat dan sistem berbasis JVM yang fleksibel.

Spring dapat digunakan untuk melakukan pengembangan aplikasi *enterprise* dan web yang dapat berjalan pada JVM manapun. Pengguna bisa *men-deploy* aplikasi yang dibangun dengan mode *standalone*, *app server*, maupun dengan *cloud PaaS*. Spring menyediakan model pemrograman terbuka yang komprehensif, kohesif, mudah dipahami dan dukungannya pun lengkap.

##### 4.1.1.1 Kelebihan Spring Framework

###### a. *Mature*

Spring Framework yang sudah dibuat sejak tahun 2002, sudah banyak berjalan dan digunakan dalam pengembangan aplikasi-aplikasi besar di *production*.

###### b. *Lightweight*

Jika dibandingkan dengan Java Enterprise Edition, Spring Framework sangatlah ringan. *Spring Framework* bisa jalan walaupun hanya di aplikasi web server seperti Tomcat atau Jetty.

###### c. *Supported Framework*

Spring Framework sebenarnya bukanlah framework untuk menggantikan framework-framework yang lain, justru tujuan utama Spring Framework adalah sebagai fondasi

untuk framework-framework yang lain. Spring Framework sendiri mendukung dan didukung oleh framework-framework yang sudah mature lainnya, seperti Hibernate, AspectJ, Bean Validation, JPA, Hystrix, OpenFeign, RxJava dan lain-lain. Oleh karena itu, saat menggunakan Spring Framework sebagai fondasi aplikasi, maka akan banyak sekali framework mature yang bisa digunakan dengan mudah, bahkan juga bisa dengan mudah mengintegrasikan framework lain secara manual.

d. *Good API Design*

e. *Backward Compatible*

Sejak pertama kali dirilis menggunakan konfigurasi XML hingga sekarang yang lebih populer menjadi konfigurasi *Annotation*, *Spring Framework* selalu *backward compatible*. Tidak seperti *Play Framework* yang setiap ganti versi, selalu *rewrite framework*.

#### 4.1.1.2 Struktur Spring Framework

a. *Model/Entity/Domain*

Merupakan *class* yang mereplikasikan tabel yang ada di *database*. *Class* ini berisi atribut-atribut yang merepresentasikan *field-field* yang ada di *database* beserta dengan *setter* dan *getter*-nya.

b. *Controller*

Merupakan pengontrol yang akan mengolah *request* ke sistem dan *response* yang akan diberikan sistem. *Controller* akan memanggil *service* untuk bisa mendapatkan *response*.

c. *Service*

Menampung bisnis proses dan semua *logic* dari sistem. Disini, *service* akan memanggil *mapper* untuk bisa mendapatkan ataupun meng-*inject*-kan data ke *database*.

d. *Mapper/Repository/Dao*

Merupakan *class* yang digunakan untuk memetakan fungsi-fungsi yang berhubungan dengan *database*.

#### 4.1.2 Angular JS

Angular adalah sebuah *framework* Javascript yang memungkinkan kita membuat reaktif *Single Page Application* (SPA). SPA yaitu aplikasi yang berjalan hanya pada satu halaman, tidak membutuhkan *reload page* meskipun nampak di url berpindah halaman.

AngularJS merupakan *framework* Javascript *open source* yang dirilis oleh Google pada tahun 2009. Konsep dari AngularJS adalah meningkatkan fungsi dari HTML untuk membangun *web app*. Bayangkan awalnya HTML hanya digunakan untuk membuat halaman *website* statis dan kini bisa berfungsi untuk membuat *web app* dengan menggunakan Angular JS. AngularJS bukan berupa *library* melainkan *framework* yang solid. Sama seperti *framework* lainnya. AngularJS memiliki konsep MVC (Model, View, Control) meskipun dengan cara yang berbeda.

##### 4.1.2.1 Fitur Umum

1. Angular JS adalah *framework* yang efisien untuk membangun *Rich Internet Applications* (RIA).
2. Angular JS memberikan *developer* pilihan untuk menulis aplikasi pada sisi *client* dengan javascript dan pendekatan MVC (*Model, View, Controller*).
3. Aplikasi dengan menggunakan Angular JS bisa digunakan di segala *web browser*. AngularJS secara otomatis *menghandle* Javascript supaya cocok dengan *web browser* yang digunakan.
4. AngularJS adalah *open source*, gratis. Telah digunakan oleh ribuan *developer* di seluruh dunia.

#### 4.1.2.2 Fitur Core

1. *Data binding* : sinkronisasi data otomatis antara *model* dan *view*
2. *Scope* : objek yang mengacu pada model, sebagai perekat antara *controller* dan *view*.
3. *Controller* : kumpulan fungsi-fungsi dari Javascript.
4. *Services* : AngularJS datang dengan beberapa *built-in services* seperti `$http` untuk membuat sebuah `XMLHttpRequests`.
5. *Filters* : pilihan item dari sebuah *array* dan mengembalikan sebuah *array* baru.
6. *Directives* : *marker* dari DOM elemen seperti *element*, *attribute*, CSS dan lainnya.
7. *Templates* : tampilan yang *dirender* dengan informasi dari *controller* dan *model*.
8. *Routing* : perpindahan tampilan.
9. *Dependency Injection* : terdapat *built in dependency injection* sehingga membantu *developer* untuk membuat dan mencoba aplikasi secara mudah.

#### 4.1.2.3 Kelebihan Angular JS

1. Angular JS menyediakan kemampuan untuk membuat *Single Page Application* secara mudah.
2. Angular JS menyediakan kemampuan *data binding* pada html.
3. Kode Angular JS adalah unit yang bisa dites.
4. Angular JS menggunakan menggunakan *dependency injection*.
5. Angular JS menyediakan komponen yang bisa digunakan kembali.

6. Dengan Angular JS, *developer* bisa membuat fungsionalitas yang lebih baik dengan kode yang lebih ringkas.
7. Dalam AngularJS, *view* adalah html murni, dan *controller* ditulis dalam Javascript sebagai *business processing*.

#### **4.1.2.4 Kelemahan Angular JS**

1. Tidak aman, autentifikasi dari *server* harus tetap dijaga untuk menjaga keamanan aplikasi.
2. Kebergantungan dengan *web browser*, jika Javascript di-disable dari *web browser* maka semua kode tidak bisa bekerja.

#### **4.1.2.5 Tiga Bagian Utama Angular JS**

1. ng-app : definisi arahan dan *link* dari aplikasi Angular JS ke html.
2. ng-model : arahan data dari aplikasi AngularJS ke input kontrol html.
3. ng-bind : arahan data dari aplikasi Angular JS ke tag html.

## **4.2 Penjabaran Proyek Tenancy Management**

### **4.2.1 Overview**

*Tenancy Management* (TM) merupakan sebuah aplikasi yang ditujukan untuk mendukung pengelolaan tenant yang merupakan customer pelanggan aplikasi-aplikasi yang dimiliki oleh PT Aristi Jasadata. Aplikasi ini dikembangkan dengan mengadopsi *multi tenant* software architecture , dimana aplikasi ini berjalan di atas beberapa server dari banyak tenant . Aplikasi ini dikembangkan dengan tujuan utama untuk memudahkan pembuatan database baru untuk masing-masing tenant yang telah terdaftar secara otomatis dan tanpa perlu melakukan penyusunan database dari awal. Selain itu, aplikasi ini juga berguna untuk melakukan kontrol terhadap entitas dan parameter yang ada serta mempermudah komunikasi dengan masing-masing tenant .

Aplikasi ini ditujukan untuk pihak internal PT Aristi Jasadata sebagai admin yang melakukan pengelolaan aplikasi serta pelanggannya.

*Email notifikasi* merupakan fitur dari Tenancy Management (TM) yang digunakan sebagai media komunikasi antara admin Tenancy Management (TM) dengan *customer*. Fitur ini dapat digunakan untuk mengirim email kepada *contact person* dari *customer* yang terdaftar dan kepada pihak luar. Fitur ini terbagi menjadi tiga macam, yaitu :

1. *Manual*

Pengiriman email secara manual. *User* mengisikan semua yang dibutuhkan untuk mengirim email secara manual, kemudian mengirimnya seperti email pada umumnya.

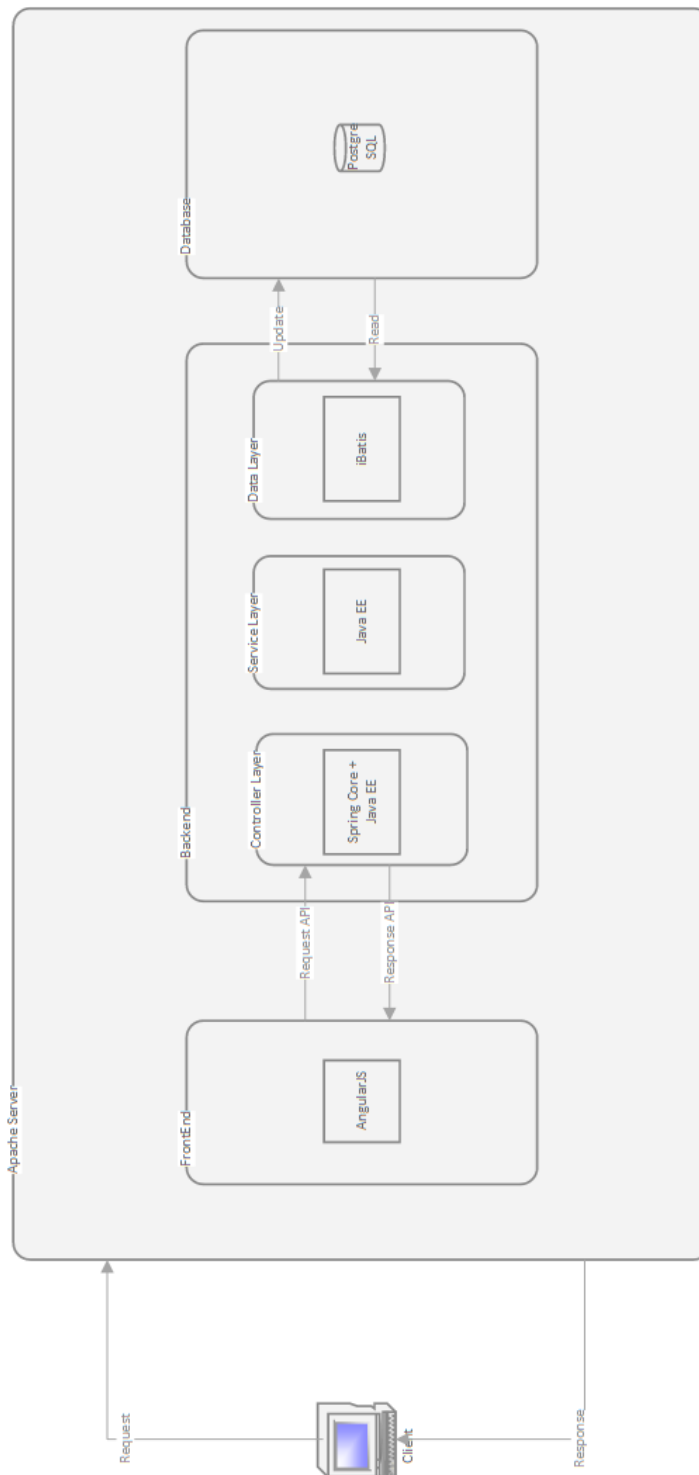
2. *Semi Automatic*

Pengiriman email secara *semi automatic* yaitu pengiriman email melalui *template* email yang telah dibuat sebelumnya.

3. *Automatic*

Pengiriman email *automatic* terbagi menjadi dua, yaitu pengiriman email terjadwal secara *periodic* yang dapat dibuat oleh *user* dan pengiriman email peringatan secara *countdown* (peringatan limit customer dan tagihan pembayaran).

## 4.2.2 Arsitektur Sistem



Gambar 4. 1 Arsitektur *Tenancy Management*

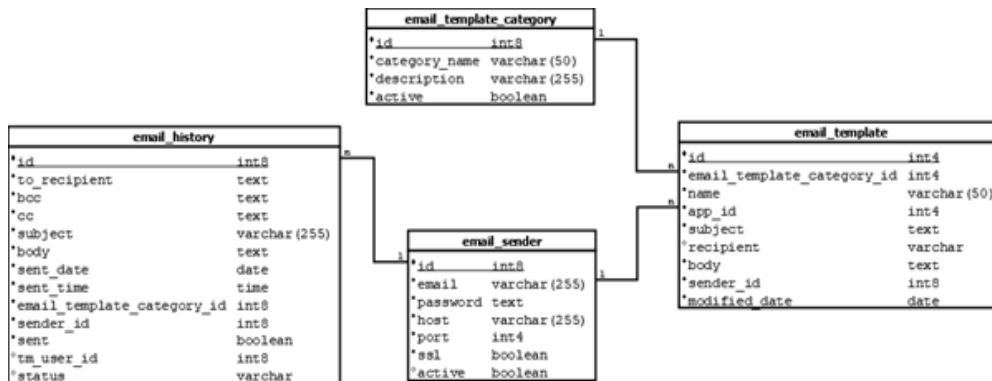


#### 4.2.3 Hierarki Menu

Berikut merupakan hierarki menu dari Tenancy Management :

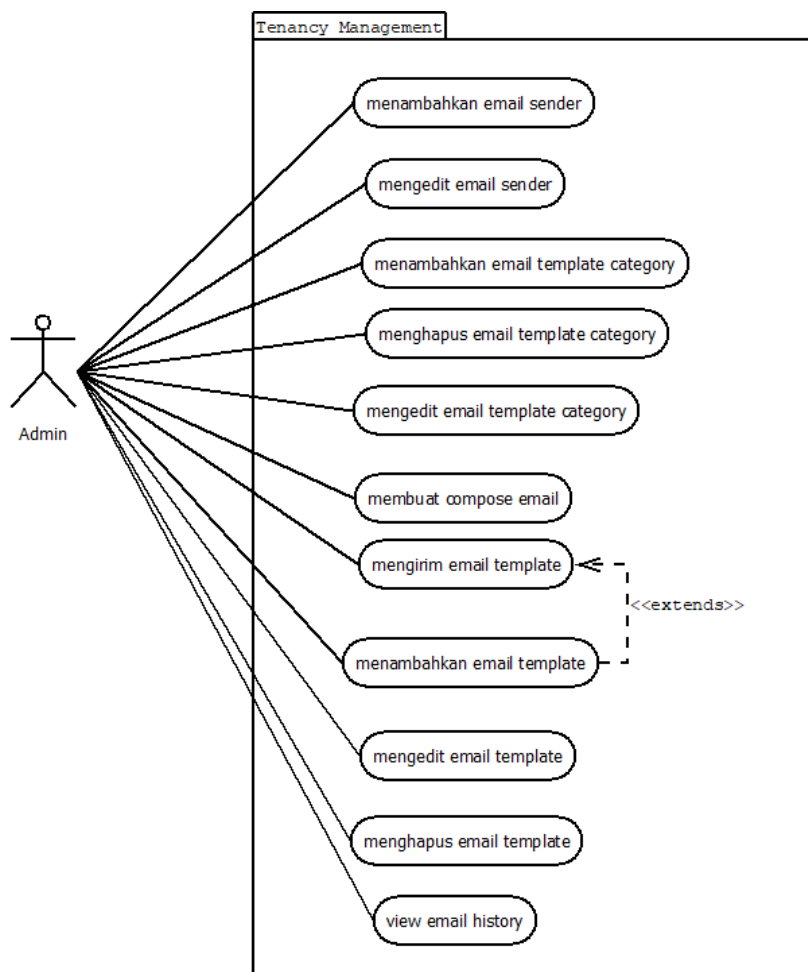
- *Home*
- *Customer*
  - *Add customer*
  - *Edit customer*
  - *Delete customer*
- *Email Notification*
  - *Compose email*
  - *Email History*
- *Master Parameter*
  - *Application*
    - *Add application*
    - *Edit application*
  - *Email Template Category*
    - *Add email template category*
    - *Edit email template category*
    - *Delete email template category*
  - *Email Sender*
    - *Add email sender*
    - *Edit email sender*
  - *Email Template*
    - *Add email template*
    - *Edit email template*
    - *Delete email template*
    - *Send email template*
  - *Tenant Template*

#### 4.2.4 Diagram Tabel



Gambar 4. 2 Diagram Tabel *Tenancy Management (Email Notification)*

#### 4.2.5 Usecase Diagram

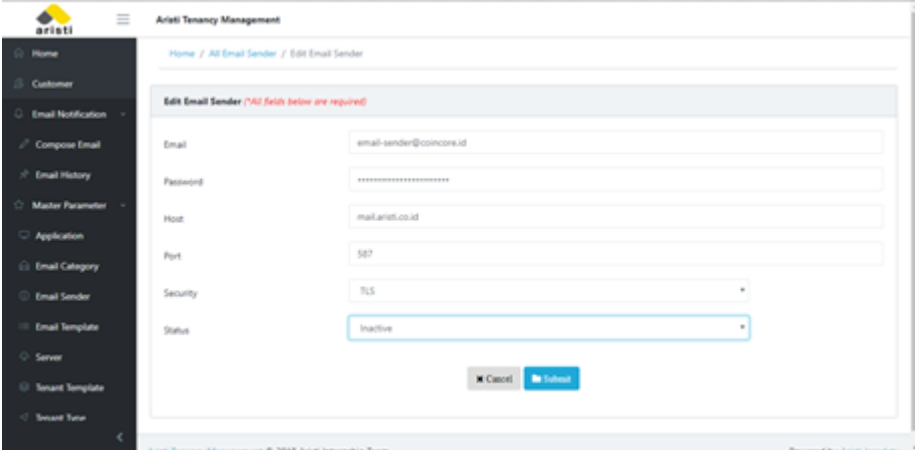


Gambar 4. 3 Usecase diagram *Tenancy Management (Email Notification)*



Halaman ini digunakan untuk menambahkan *email sender* dengan cara mengisi semua *field* yang ada pada *form* tersebut. Ketika menekan *button submit* maka data akan dikirim dan disimpan dalam *database*.

- **Halaman *Edit Email Sender***



The screenshot shows the 'Edit Email Sender' form in the Ariati Tenancy Management system. The form is titled 'Edit Email Sender (\*All fields below are required)'. It contains the following fields: Email (email-sender@cosincore.id), Password (masked with asterisks), Host (mail.ariati.co.id), Port (587), Security (TLS), and Status (Inactive). There are 'Cancel' and 'Submit' buttons at the bottom right. The left sidebar shows the navigation menu with options like Home, Customer, Email Notification, Compose Email, Email History, Master Parameter, Application, Email Category, Email Sender, Email Template, Server, Tenant Template, and Tenant User.

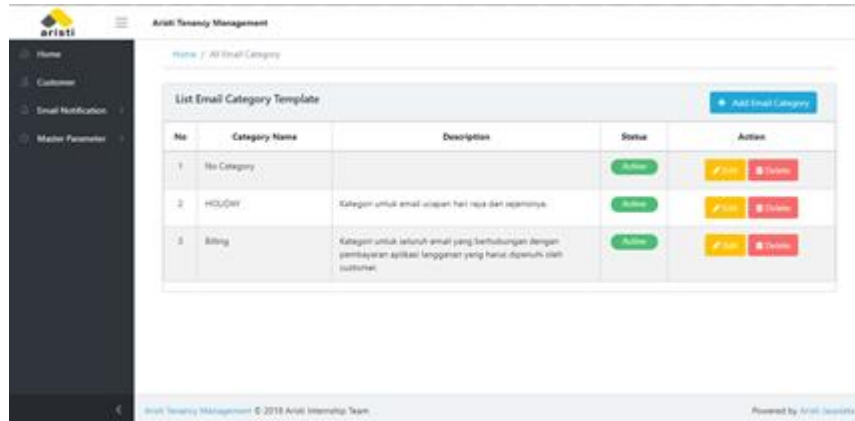
Gambar 4. 6 Halaman *Edit Email Sender*

Halaman ini digunakan untuk mengubah data *email sender* yang telah dibuat sebelumnya.

#### **4.2.6.2 Email Category**

Digunakan untuk menyimpan dan mengolah data mengenai kategori *email template* yang nantinya akan digunakan untuk mengkategorikan email.

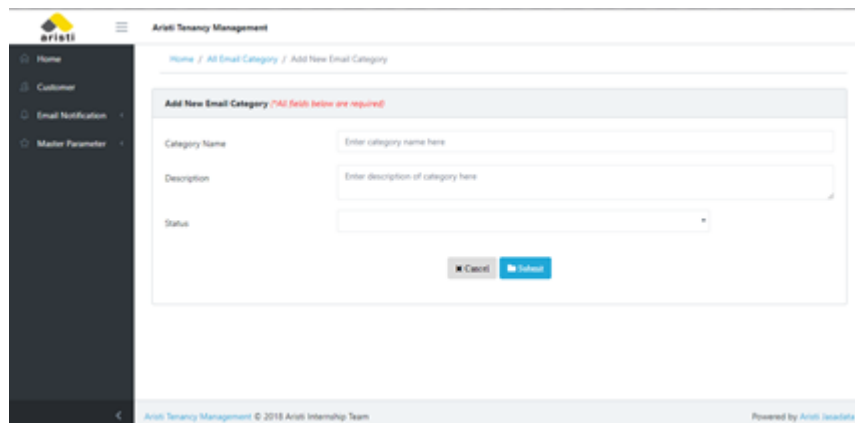
- **Halaman *List Email Category***



Gambar 4. 7 Halaman *List Email Category*

Halaman ini menampilkan *list*/daftar dari keseluruhan *Email Category* yang telah terdaftar dan tersimpan dalam *database* sistem.

- **Halaman *Add New Email Category***



Gambar 4. 8 Halaman *Add New Email Category*

Halaman ini digunakan untuk menambahkan *Email Category* dengan cara mengisi semua *field* yang ada pada form tersebut. Ketika menekan *button submit* maka data akan dikirim dan disimpan dalam *database*.

- **Halaman *Edit Email Category***

The screenshot shows the 'Edit Email Category' page. The left sidebar contains navigation links: Home, Customer, Email Notification, and Master Parameter. The main content area has a breadcrumb 'Home / All Email Category / Edit Email Category'. Below this is a form titled 'Edit Email Category (All fields below are required)'. The form contains three fields: 'Category Name' with the value 'Promotion Day', 'Description' with the value 'Kategori untuk penawaran promo, diskon, dan sebagainya', and 'Status' with a dropdown menu set to 'Active'. At the bottom of the form are 'Cancel' and 'Submit' buttons. The footer of the page includes 'Ariati Tenancy Management © 2018 Ariati Internship Team' and 'Powered by Ariati, Inovadota'.

Gambar 4. 9 Halaman *Edit Email Category*

Halaman ini digunakan untuk mengubah data *Email Category* yang telah dibuat sebelumnya.

#### 4.2.6.3 Email Template

Digunakan untuk menyimpan dan mengolah *email template* yang kemudian dapat dikirim oleh *user* berulang kali. Email tujuan atau penerima dari email ini hanya email-email yang terdaftar di *contact person* dari customer yang telah berlangganan.

- **Halaman *List Email Template***

The screenshot shows the 'List Email Template' page. The left sidebar contains navigation links: Home, Customer, Email Notification, Master Parameter, Application, Email Category, Email Sender, Email Template, Tenant Template, and Timeline. The main content area has a breadcrumb 'Home / All Email Template'. Below this is a table titled 'List Email Template' with a '+ Add New Email Template' button. The table has the following data:

No.	Template Name	Subject	Application	Category	Action
1	Tax Payment Alert	Tax Payment - Alert for July	Coin	Billing	[View] [Edit] [Delete] [Email]
2	Monthly Sale	July Sale Promotion	Coin	No category	[View] [Edit] [Delete] [Email]
3	Name	Subject	Coin	Holiday	[View] [Edit] [Delete] [Email]
4	Nama	00000000	Coin	Holiday	[View] [Edit] [Delete] [Email]

At the bottom of the table is a 'Page 1' indicator. The footer of the page includes 'Ariati Tenancy Management © 2018 Ariati Internship Team' and 'Powered by Ariati, Inovadota'.

Gambar 4. 10 Halaman *List Email Template*

Halaman ini menampilkan list/daftar dari keseluruhan *Email Template* yang telah terdaftar dan tersimpan dalam database sistem.

- **Halaman *Add New Email Template***

The screenshot shows the 'Add New Email Template' page in the Ariati Tenancy Management system. The page has a dark sidebar on the left with navigation links: Home, Customer, Email Notification, Master Parameter, Application, Email Category, Email Sender, Email Template, Tenant Template, Timezone, Application, Email Category, Email Sender, Email Template, Tenant Template, and Timezone. The main content area is titled 'Add New Email Template' and contains a form with the following fields: Template Name (text input), Application (dropdown), Template Category (dropdown), Sender Email (text input), Recipient (text input), Subject (text input), and Body (rich text editor). The Body field has a toolbar with options: File, Edit, Insert, View, Format, and Tools. At the bottom of the form are 'Cancel' and 'Submit' buttons. The footer of the page reads 'Ariati Tenancy Management © 2018 Ariati Internship Team' and 'Powered by Ariati Javadata'.

Gambar 4. 11 Halaman *Add New Email Template*

Halaman ini digunakan untuk menambahkan *Email Template* dengan cara mengisi semua field yang ada pada form tersebut. Ketika menekan button submit maka data akan dikirim dan disimpan dalam database.

- **Halaman *Edit Email Template***

Gambar 4. 12 Halaman *Edit Email Template*

Gambar 4. 13 Halaman *Edit Email Template* (lanj.)

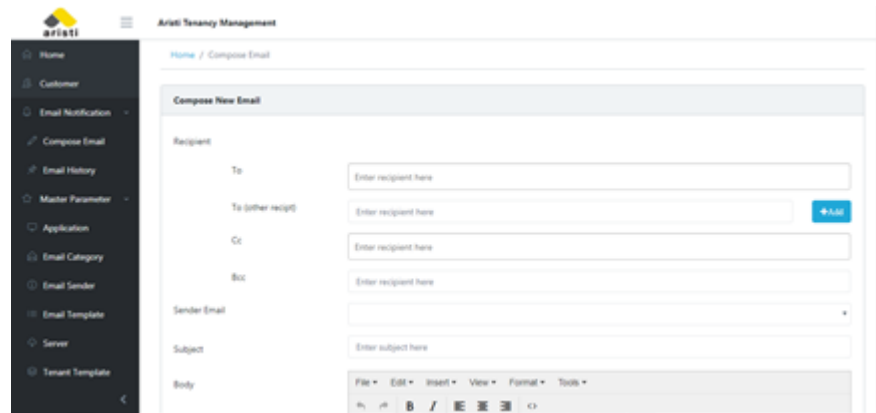
Halaman ini digunakan untuk mengubah data *Email Template* yang telah dibuat sebelumnya.

#### 4.2.6.4 **Compose Email**

*Compose Email* merupakan salah satu fitur dari paket *Email Notification* yang berfungsi untuk menulis *email* manual (*manual compose*). Penggunaannya kurang lebih sama dengan penggunaan *email* pada umumnya.



- **Halaman *Compose Email***



Gambar 4. 14 Halaman *Compose Email*

#### 4.2.6.5 Email History

Merupakan *log* yang digunakan untuk menyimpan email yang telah dikirim oleh *User* baik yang berhasil maupun gagal. Email history tidak dapat dihapus maupun di edit. Pada *email history* ini terdapat status terkirim atau tidaknya email beserta alasannya jika email tidak terkirim.

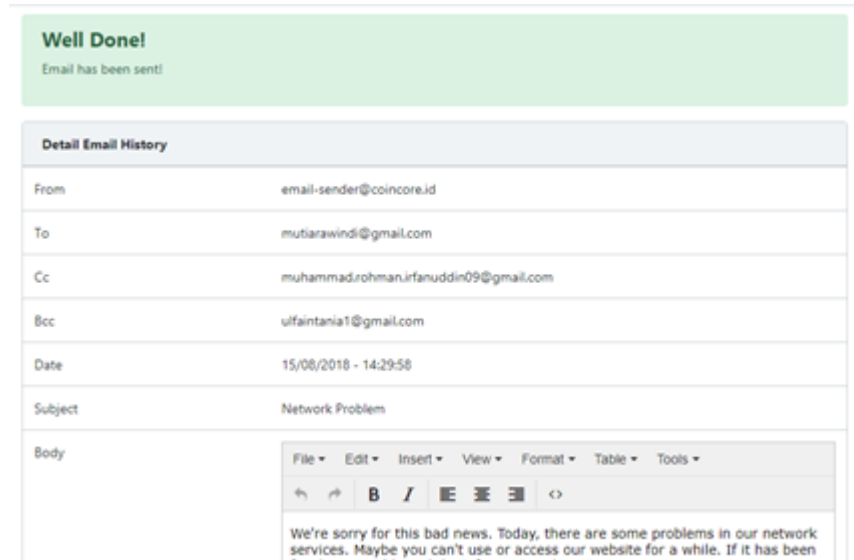
- **Halaman *List Email History***

Email History						
No.	Subject	Recipient	Time	Category	Status	Action
81	July Sale Promotion	mutiarawindi@gmail.com	15/06/2018 - 11:35:08	No category	Success	<a href="#">View</a>
82	Network Problem	mutiarawindi@gmail.com	15/06/2018 - 14:27:06	No category	Failed	<a href="#">View</a>
83	Network Problem	mutiarawindi@gmail.com	15/06/2018 - 14:29:58	No category	Success	<a href="#">View</a>
84	Network Problem	mutiarawindi@gmail.com	15/06/2018 - 14:32:56	No category	Failed	<a href="#">View</a>
<div> <a href="#">Previous</a> Page 9 <a href="#">Next</a> </div>						

Gambar 4. 15 Halaman *List Email History*

Halaman ini menampilkan list/daftar dari keseluruhan *Email History* yang telah terdaftar dan tersimpan dalam database sistem.

- **Halaman *Detail Email History***



Gambar 4. 16 Halaman *Detail Email History*

Halaman *detail email history* berisi rincian *email* yang telah dikirim.

## 4.2.7 Entitas

### 4.2.7.1 Email Sender

*Email Sender* merupakan fitur untuk menambahkan *sender* (pengirim) yang digunakan dalam pengiriman *email*. *Email Sender* mempunyai 2 fungsi utama yaitu menambahkan dan mengubah *sender*.

#### **Aturan pengisian *email sender* :**

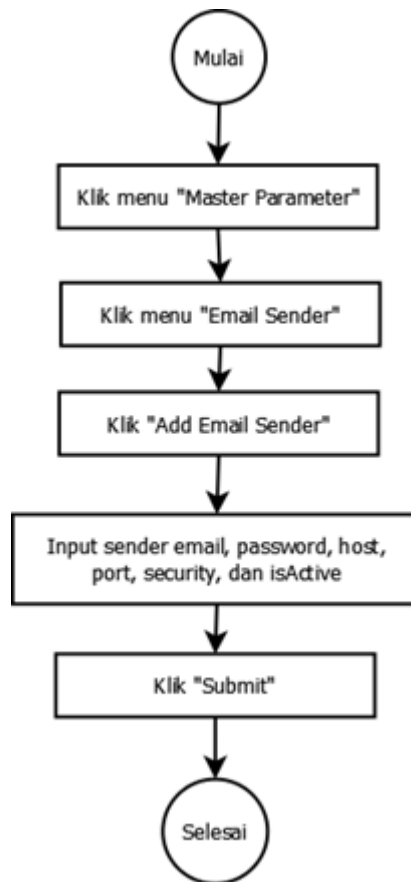
- Semua field wajib diisi.
- Email diisi dengan alamat email asli yang valid dan tidak boleh sama dengan alamat email yang sudah ada sebelumnya (yang sudah digunakan untuk semua jenis *security*) atau dengan kata lain, 1 email dapat menggunakan 2 *security*.
- Password* yang diisikan merupakan *password* asli dari email tersebut dengan minimal 8 karakter.

- d. *Host* berisi *hostname* resmi yang digunakan oleh email terkait, misalnya : email ber-domain gmail.com menggunakan smtp.gmail.com.
- e. *Port* diisi sesuai dengan jenis *security* dengan minimal 1 karakter dan maksimal 4 karakter , jika *security* email tersebut ssl maka port yang diisi adalah 465 dan jika tls maka yang diisi adalah 587.
- f. Status dapat diatur sesuai kebutuhan apakah aktif atau non-aktif.
- g. Satu *sender email* hanya dapat diinputkan sebanyak 2 kali dengan 2 *security* yang berbeda.

### **Fasilitas yang Dibutuhkan**

#### **a. Input Email Sender baru**

Untuk dapat melakukan penginputan email sender, pertama *User* dapat menuju ke menu “**Master Parameter**”. Kemudian masuk ke menu “**Email Sender**” dan pilih “**Add Email Sender**”. *User* dapat mengisikan *sender email*, *password*, *host*, *port* serta memilih *security*nya (TLS atau SSL) sesuai dengan konfigurasinya. *User* juga dapat mengeset apakah email sender tersebut aktif atau tidak. Dimana jika email sender tersebut aktif (*active true*), maka email sender tersebut dapat digunakan untuk mengirim email. Sementara itu, jika email sender tersebut tidak aktif (*active false*), maka email sender tersebut tidak dapat digunakan untuk mengirim email. Setelah semua data terisi dengan benar, *User* dapat melanjutkan penginputan dengan memilih *button* “**Submit**”.



Gambar 4. 17 Transaction Flow Add Email Sender

#### b. Edit email sender

User dapat melakukan pengeditan email sender yang telah ada dengan cara klik *button* “**Edit**” pada kolom “**Action**” di *record* email sender yang akan diedit. Kemudian edit data sesuai kebutuhan dan klik *button* “**Submit**” untuk menyimpan perubahan.



Gambar 4. 18 Transaction Flow Edit Email Sender

### API dan JSON :

#### a. API input email sender

<http://localhost:8080/tenancy-management/api/sender/create>

JSON :

```
{
  "email": "cobapaska@gmail.com",
  "password": "cobapaska123",
  "host": "smtp.gmail.com",
  "port": 465,
  "ssl": true,
  "active": true
}
```

b. *API edit email sender*

<http://localhost:8080/tenancy-management/api/sender/edit>

c. *API delete email sender*

<http://localhost:8080/tenancy-management/api/sender/delete/{emailSenderId}>

d. *API Get All*

<http://localhost:8080/tenancy-management/api/sender/list>

JSON :

```
{
    "id" :1,
    "email": "cobapaska@gmail.com",
    "password": "cobapaska123",
    "host": "smtp.gmail.com",
    "port": 465,
    "ssl": true,
    "active": true
}
```

e. *API Get by Id*

<http://localhost:8080/tenancy-management/api/sender/{emailSenderId}>

f. *API Get by Active*

<http://localhost:8080/tenancy-management/api/sender/active>

#### 4.2.7.2 Email Category

*Email Category* merupakan identitas yang digunakan untuk menyimpan dan mengolah data mengenai kategori *email template* yang nantinya akan digunakan untuk mengkategorikan email.

**Aturan pengisian *email category* :**

- Semua *field* wajib diisi.
- Category *Name* tidak boleh sama dengan yang terdaftar.

### Contoh *Email Category*

No	Kategori	Deskripsi
1	No Category	Untuk mengelompokan email tanpa kategori misalnya jika mengirim email dengan cara <i>compose manual</i> , maka secara <i>default</i> akan menggunakan kategori ini.
2	Billing	Kategori untuk seluruh email yang berhubungan dengan pembayaran aplikasi langganan yang harus dipenuhi oleh <i>customer</i> .
3	Promotion	Kategori untuk penawaran promo, diskon, dan sejenisnya.
4	Holiday	Kategori untuk email ucapan hari raya dan sejenisnya.

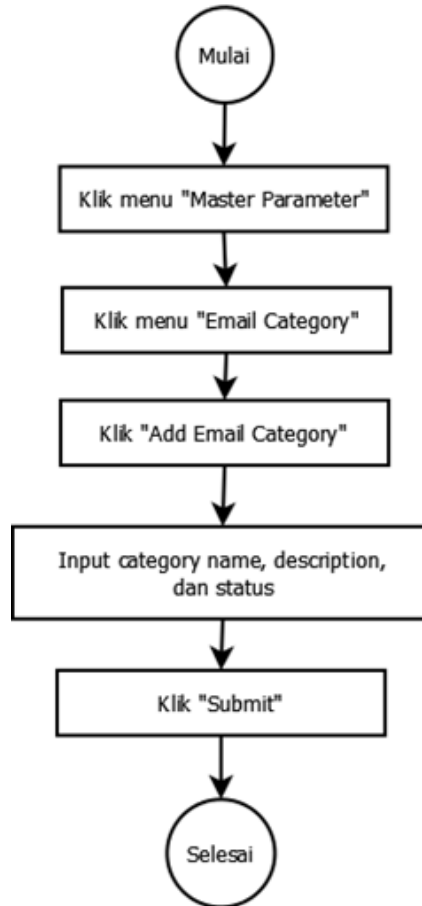
Tabel 4. 1 Tabel contoh *Email Category*

### Fasilitas yang Dibutuhkan

#### a. Input *Email Category*

Untuk melakukan penginputan atau penambahan pada *email template category*, *user* harus masuk ke menu **”Master Parameter”** kemudian masuk ke menu **“Email Category”**. Setelah itu klik **“Add Email Category”** dan inputkan data *category name*, *description* serta status. Dimana jika statusnya *active*, maka kategori tersebut dapat dipakai untuk mengkategorikan email. Sedangkan jika

statusnya *inactive*, maka tidak dapat digunakan. Kemudian, klik *button* “**Submit**” untuk menyimpannya.



Gambar 4. 19 Transaction flow Add Email Category

*b. Edit Email Category*

User dapat melakukan pengeditan kategori yang telah ada dengan cara klik *button* “**Edit**” pada kolom “**Action**” di *record* email category yang akan diedit. Kemudian edit data sesuai kebutuhan dan klik *button* “**Submit**” untuk menyimpan perubahan.





Gambar 4. 20 Transaction flow Edit Email Category

*c. Delete Email Category*

**API dan JSON :**

*a. API Insert email template category*

<http://localhost:8080/tenancy-management/api/email-template-category/create>

JSON :

```

{
  "id":1,
  "categoryName":"Big",
  "description":"besar",
  "active":true
}
  
```

- b. *API edit email template category*  
<http://localhost:8080/tenancy-management/api/email-template-category/edit>
- c. *Delete email template category*  
<http://localhost:8080/tenancy-management/api/email-template-category/delete/{emailTemCategoryId}>
- d. *Get all*  
<http://localhost:8080/tenancy-management/api/email-template-category/list>
- e. *Get by Id*  
<http://localhost:8080/tenancy-management/api/email-template-category/{emailTemCategoryId}>
- f. *Get by Active*  
<http://localhost:8080/tenancy-management/api/email-template-category/active>

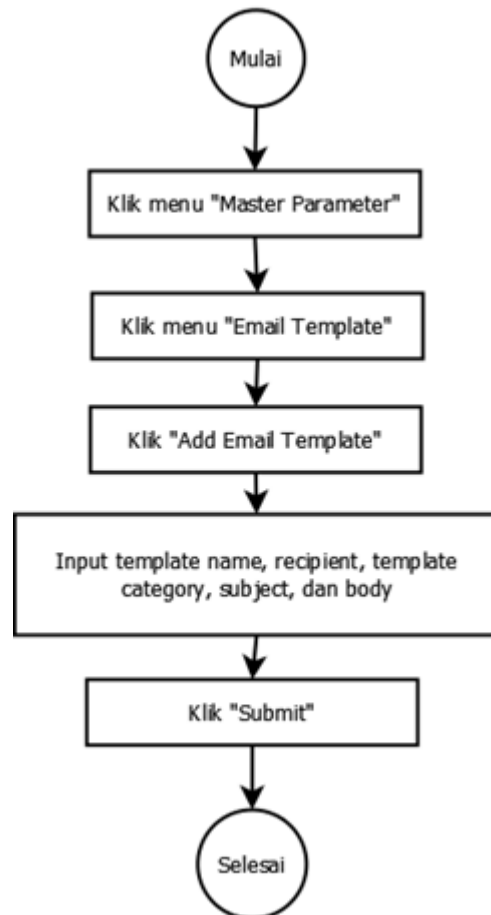
#### 4.2.7.3 **Email Template**

Digunakan untuk menyimpan dan mengolah *email template* yang kemudian dapat dikirim oleh *user* berulang kali. Email tujuan atau penerima dari email ini hanya email-email yang terdaftar di *contact person* dari customer yang telah berlangganan. Fasilitas yang disediakan :

- a. *Input Email Template*

*User* dapat melakukan penambahan atau penginputan email template dengan cara memilih menu “**Master Parameter**” dan masuk ke menu “**Email Template**”. Kemudian pilih menu “**Add New Email Template**”. Setelah masuk ke halaman *Add New Email Template*, masukkan *template name*, *application*, *recipient*, *subject*, *body*, *template category*, dan *sender email*. Kemudian pilih

“**Submit**” untuk menyimpannya. Dimana *recipient* adalah email-email yang terdaftar di dalam *contact person* dari *customer* berdasarkan *application* yang telah dipilih.



Gambar 4. 21 Transaction flow Add Email Template

b. Edit Email Template

Untuk dapat melakukan pengeditan, user harus memilih record atau data mana yang akan diedit kemudian pilih *button* “**Edit**” kemudian lakukan perubahan dan klik *button* “**Submit**”.



Gambar 4. 22 Transaction flow Edit Email Template

c. Delete Email Template

**API dan JSON**

a. API insert email template

<http://localhost:8080/tenancy-management/api/email-template/create>

JSON :

```
{
  "templateName" : "template baru",
  "subject" : "template baru subject",
  "body" : "ceritanya ini body. coba ajalah
ya. Siapa tau beruntung hehe",
  "emailTemplateCategoryId" : 1,
  "senderId" : 3,
```

```
"appId" : 3,  
"recipient" : "paskaanugrah1@gmail.com,  
paskaanugrah123@gmail.com"  
}
```

b. *API send email template*

`http://localhost:8080/tenancy-management/api/email-template/send/{emailTemplateId}`

c. *API edit email template*

`http://localhost:8080/tenancy-management/api/email-template/edit`

d. *API delete email template*

`http://localhost:8080/tenancy-management/api/email-template/delete/{emailTemplateId}`

e. *API get All*

`http://localhost:8080/tenancy-management/api/email-template/list`

f. *API get by Id*

`http://localhost:8080/tenancy-management/api/email-template/{id}`

#### **4.2.7.4 Email History**

Merupakan *log* yang digunakan untuk menyimpan email yang telah dikirim oleh *User* baik yang berhasil maupun gagal. Email history tidak dapat dihapus maupun di edit. Pada *email history* ini terdapat status terkirim atau tidaknya email beserta alasannya jika email tidak terkirim.

#### **API dan JSON :**

a. *API insert atau send email*

`http://localhost:8080/tenancy-management/api/email-history/send`

### JSON :

```
{
  "toRecipient" : "oeyeoy@gmail.com",
  "bcc" : "paskaadil@gmail.com",
  "cc" : "ulfaintania1@gmail.com",
  "subject" : "ini tls - Coba send e-mail",
  "body" : "yeay berhasil!!",
  "senderId" : 3
}
```

b. API *get All*

<http://localhost:8080/tenancy-management/api/email-history/list>

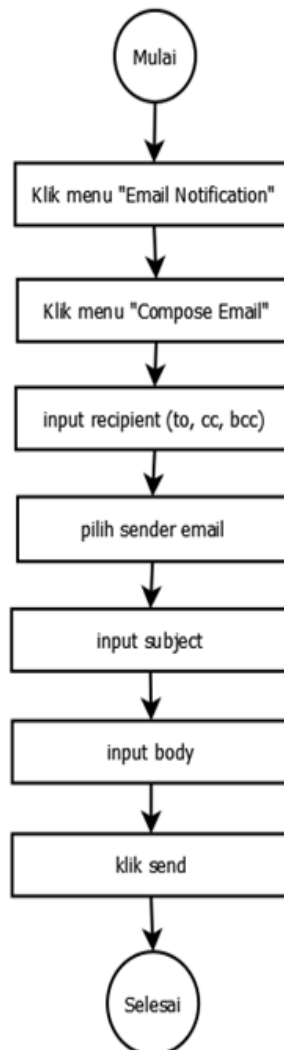
c. API *get by Id*

<http://localhost:8080/tenancy-management/api/email-history/{emailHistoryId}>

#### 4.2.7.5 Sending Email

a. Manual

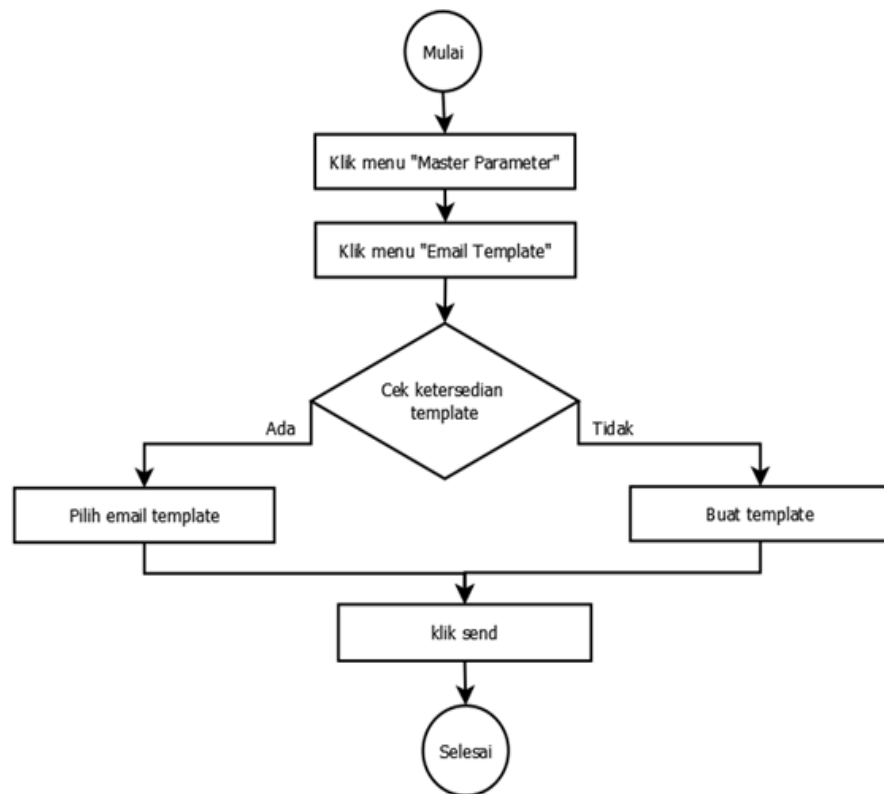
Untuk mengirim email dengan menggunakan metode manual, *user* harus menambahkan *recipient email* pada *field to*, *bcc*, dan *cc*. *Recipient email* pada *field to* merupakan email yang diperoleh dari *contact person* yang terdaftar dan email yang dimasukkan secara manual oleh *user* (email selain yang terdaftar di *contact person* dari *customer*). Untuk *field bcc* dan *cc* merupakan email yang diperoleh dari *contact person* yang terdaftar dari *customer* saja. Selain itu, *user* harus memilih *sender email* aktif yang telah terdaftar, menginputkan *subject*, dan *body* sebelum mengirimkan emailnya.



Gambar 4. 23 Transaction flow Compose Email (manual)

**b. Semi Automatic**

Untuk dapat menggunakan dan mengirim email dengan menggunakan metode semi automatic, pertama *user* harus mengecek ketersediaan *email template* yang dimaksudkan untuk dikirim. Jika ada, maka user hanya perlu mengklik *button* “**Send**”. Jika belum, maka user harus membuat *email template* terlebih dahulu kemudian mengklik *button* “**Send**”.



Gambar 4. 24 *Transaction flow Semi Automatic Mail*



## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Dari proses kami melaksanakan Praktik Kerja Lapangan serta didukung dengan penjelasan-penjelasan yang telah kami sampaikan pada bab-bab sebelumnya, dapat kami simpulkan bahwa:

- a. *Tenancy Management* sangat membantu pengelolaan *tenants* yang menyewa aplikasi PT Aristi Jasadata.
- b. Fitur notifikasi email dari *Tenancy Management* bisa mempermudah pekerjaan admin dalam menghubungi pelanggan ketika terjadi tagihan, kendala teknis, atau pemberitahuan lainnya.
- c. Dalam fitur notifikasi email, terdapat 2 sub-fitur, dimana admin bisa membuat email manual dan email template.
- d. Dalam pelaksanaan Praktik Kerja Lapangan ini kami mendapatkan sangat banyak pengalaman yang berharga yang tidak dapat kami dapatkan dalam pembelajaran mata kuliah.

#### **5.2 Hal-hal yang Dapat Dikembangkan di Lokasi PKL**

Setelah dilaksanakannya kegiatan Praktik Kerja Lapangan, terdapat hal-hal yang dapat kami peroleh sebagai bentuk dari proses pengembangan diri, diantaranya :

- a. Pengalaman bagaimana bekerja dalam sebuah tim. Dalam pengerjaan sebuah *team project*, diperlukan koordinasi antar anggotanya dan tanggung jawab masing-masing anggota terhadap pekerjaan yang diemban agar selesai tepat pada waktunya.
- b. Bertambah *skill* dan informasi yang berkaitan dengan dunia IT di industri. Bagaimana manajemen proyek yang besar, bagaimana melakukan evaluasi, dan bagaimana proyek itu dikerjakan.
- c. Membuat pola pikir baru mengenai bagaimana menyelesaikan masalah yang dihadapi.

- d. Mengetahui *tools* yang bermanfaat dalam pengerjaan proyek, diantaranya *Git*, *Sourcetree*, *Postman*, *DBever*, *Visual Studio Code*, *Sublime Text 3*, *Eclipse*. Ada pula aplikasi berbasis web yang bernama *trello.com* untuk berkolaborasi dan melihat *progress* dari setiap personil yang ada di proyek itu.
- e. Mempelajari framework yang digunakan oleh perusahaan untuk membuat aplikasi *enterprise* seperti, Spring Framework, NodeJS, AngularJS, API (*Application Program Interface*).

### 5.3 Saran

Kegiatan Praktik Kerja Lapangan ini merupakan kegiatan yang sangat penting dan bermanfaat untuk diploma, utamanya Prodi Komputer dan Sistem Informasi UGM. Melalui kegiatan ini mahasiswa akan dapat merasakan implementasi dari ilmu-ilmu yang didapatkannya di perkuliahan dalam dunia kerja. Sehingga untuk tenaga vokasi hal seperti ini juga membuka wawasan baru untuk mahasiswa memilih dan memperdalam ilmu yang diinginkannya. Seperti kata pepatah bahwa pengalaman merupakan guru terbaik, melalui pengalaman kami, akan kami sampaikan beberapa saran untuk pengembangan program Praktik Kerja Lapangan ini dan untuk pengembangan Program Studi Komputer dan Sistem Informasi.

Sistem proyek SIM (Sistem Informasi dan Multimedia) yang sekarang bisa diadopsi ke PKL yang selanjutnya, dimana pihak kampus dapat menjalin relasi ke beberapa perusahaan untuk bekerja sama, kemudian saat *briefing* PKL bisa dilakukan semacam bursa tempat PKL yang mana nantinya setiap kelompok yang ingin ke perusahaan tersebut melakukan tes sesuai kebijakan perusahaan masing-masing.

### **DAFTAR PUSTAKA**

- Micheline, K. H. (2006). *Data Mining : Concept and Techniques*. Chicago: Morgan Kaufmann.
- Ortiz, E. G. (2007). *A Scalable and Efficient Outlier Detection Strategy for Categorical Data* . Florida: University of Central Florida.
- Pamungkas, A. (2008). *Deteksi Outlier pada Data Kategorik Menggunakan Algoritma LSA*. Bandung: Telkom University.