PERANCANGAN ARSITEKTUR SISTEM TIKET ELEKTRONIK KERETA API MENGGUNAKAN KERANGKA *SERVICE ORIENTED ENTERPRISE ARCHITECTURE*

(Studi Kasus : PT Railink)

TESIS

Disusun sebagai salah satu syarat untuk
Memperoleh gelar Magister Komputer
dari Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI

Oleh:

IRPAN BUDIANA

NPM: 2016210069

PROGRAM STUDI PASCASARJANA
MAGISTER SISTEM INFORMASI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER LIKMI
BANDUNG

2018

ABSTRAK

PERANCANGAN ARSITEKTUR SISTEM TIKET ELEKTRONIK KERETA API MENGGUNAKAN KERANGKA *SERVICE ORIENTED ENTERPRISE ARCHITECTURE*
(Studi Kasus : PT Railink)

Oleh:
Irpan Budiana
2016210069

Penelitian bertujuan untuk memberikan *guideline* perancangan arsitektur *enterprise* sistem tiket elektronik (*e-ticketing*) kereta api di Indonesia berorientasi *service* dengan menggunakan kerangka *Service Oriented Enterprise Architecture (SOEA)*. Metodologi analis dan desain berorientasi objek digunakan dalam menggambarkan struktur perancangan sistem *e-ticketing* kereta api. Kerangka perancangan arsitektur *enterprise e-ticketing* kereta api berorientasi *service* dimulai dengan menganalisis arsitektur *as-is* yang menganalisis arsitektur yang sedang berjalan dilanjutkan dengan perancangan arsitektur *to-be* untuk merancang arsitektur yang akan dikembangkan dari segi proses bisnis, aplikasi, dan infrastruktur. Perancangan arsitektur *enterprise e-ticketing* kereta api berbasis *service* menggunakan pola arsitektur berbasis *microservice* untuk memenuhi kebutuhan fungsional sistem *e-ticketing* kereta api yang *resilient*, *scalable*, *interoperable*, dan mampu diintegrasikan dengan berbagai *platform* standar. Perancangan arsitektur *to-be* menghasilkan sebuah perancangan aristektur bisnis dan teknologi informasi *to-be* yang akan diimplementasikan ke dalam sistem *e-ticketing* kereta api baru dengan berorientasi *service*.

Kata kunci: perancangan, arsitektur *enterprise*, *e-ticketing, service oriented enterprise architecture, service oriented architecture, microservice, object-oriented analysis and design*.

ABSTRACT

RAILWAY ELECTRONIC TICKETING SYSTEM ARCHITECTURE DESIGN USING SERVICE ORIENTED ENTERPRISE ARCHITECTURE FRAMEWORK (Case Study : PT Railink)

By:
Irpan Budiana
2016210069

The research aims to provide a guideline for designing service oriented enterprise architecture of railway e-ticketing system in Indonesia by using Service Oriented Enterprise Architecture (SOEA) framework. Object-oriented Analysis and Design (OOAD) methodology was used for describing railway e-ticketing system design structure. The enterprise architecture design framework of railway e-ticketing service-oriented consists of as-is architecture analysis that analyzes existing architecture followed by to-be design architecture for designing the architecture will be developed both in terms of business processes, applications, and infrastructures. The enterprise architecture design of railway e-ticketing architecture uses microservice architecture pattern in order to meet the system requirements: resilient, scalability, interoperability, and integrated systems with various standard platforms. To-be business and information system architectural design will be implemented in a new service oriented railway e-ticketing system.

Keywords: system design, enterprise architecture, e-ticketing, service oriented enterprise architecture, service oriented architecture, microservice, object-oriented analysis and design.

KATA PENGANTAR

Bismillahirrahmaanirraahiim

Segala puji dan syukur saya panjatkan ke hadirat Allah SWT yang mana atas berkat dan rahmat-Nya tesis ini dapat diselesaikan sebagai syarat kelulusan program Magister Program Studi Sistem Informasi di Sekolah Tinggi Manajemen Informatika LIKMI Bandung.

Saya mengucapkan terima kasih kepada Bapak Dr. Eng. H. Ana Hadiana yang senantiasa memberikan bimbingan dan arahan sehingga tesis ini dapat diselesaikan dengan sebaik-baiknya.

Selain itu saya menyampaikan terima kasih kepada semua pihak yang telah mendukung selama penyelesaian tesis ini. Saya mengucapkan terima kasih kepada:

1. Bapak Dr. Budi Permana, S.E., Ak., M.Sc., selaku Ketua STMIK LIKMI Bandung sekaligus, atas dukungan yang telah dan semangat yang telah diberikan.
2. Seluruh dosen, petugas tata usaha, akademik, dan pengurus perpustakaan STMIK LIKMI Bandung atas ilmu dan bantuan yang telah diberikan.
3. Seluruh teman-teman Magister Sistem Informasi STMIK LIKMI Bandung angkatan 2016/2017.
4. Keluarga tercinta, atas bantuan, saran, dukungan, pengertian dan do'anya.

Ucapan terima kasih penulis sampaikan kepada semua pihak yang yelah banyak memberikan dukungan kepada penulis. Semoga Allah SWT memberikan berkah dan pahala atas dukungan yang telah diberikan. Aamiin.

Pada tesis ini, mungkin terdapat kekurangan dan kesalahan. Oleh karena itu, penulis mengharapkan adanya kritik dan saran yang bersifat membangun agar kemajuan ilmu dan teknologi dapat dicapai dengan baik.

Terima kasih.

Bandung, 7 Agustus 2018

Irpan Budiana

DAFTAR ISI

halaman

ABSTRAK 1

ABSTRACT ii

KATA PENGANTAR iii

DAFTAR ISI iv

DAFTAR GAMBAR viii

DAFTAR TABEL x

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 2

1.3 Tujuan 3

1.4 Batasan Masalah 3

1.5 Kegunaan Hasil 3

BAB II LANDASAN TEORI 5

2.1 Pengertian Arsitektur *Enterprise* 5

2.2 *Service Oriented Architecture* (SOA) 6

2.2.1 Komponen SOA 8

2.2.2 Keuntungan SOA 9

2.2.3 Prinsip Dasar Service 10

2.2.4 *Web Service* 11

2.3 Arsitektur *Microservice* 16

2.3.1 Definisi *Microservice* 16

2.3.2 Keterbatasan pada aplikasi monolitik 17

2.3.3 Fitur dan Keuntungan pada arsitektur *microservice* 18

2.3.4 *Service Oriented Architecture* (SOA) dan *Microservice* 18

2.3.4.1 *Service Service Oriented Architecture* (SOA) 18

2.3.4.2 *Microservice* 19

2.4 Kerangka *Service Oriented Enterprise Architecture* (SOEA) 21
2.5 *Value Chain* 26
2.5.1 Pengertian *Value Chain* 27
2.5.2 Tujuan *Value Chain* 28
2.5.3 Konsep *Value Chain* 28
2.6 *Unified Modeling Language* (UML) 30
2.6.1 Diagram *Activity* 30
2.6.2 Diagram *Use case* 31
2.6.3 *Domain Class Diagram* 32
2.6.4 *First-Cut Design Class Diagram* 33
2.6.6 *System Sequence Diagram* 33
2.6.7 *First Cut Sequence Diagram* 35
2.6.7 *Updated Class Diagram* 35
2.7 *Electronic Ticketing* (*e-ticketing*) 36
2.7.1 Definisi *e-ticketing* 36
2.7.2 Keuntungan dan Kekurangan *E-ticketing* 37
2.7.3 Infrastruktur *e-ticketing* angkutan massal 38
2.7.4 Pemrosesan Informasi Tiket 39
2.8 Studi Hasil Penelitian Sebelumnya 40
BAB III OBYEK DAN METODOLOGI PENELITIAN 42
3.1 Profil PT Railink 42
3.2 *Milestone* Sejarah PT Railink 43
3.3 Visi, Misi, Tujuan dan Sasaran 44
3.4 Fasilitas & Sarana 45
3.5 Sistem Tiket di PT Railink 46
3.6 Metodologi Penellitian 47
3.6.1 Pemodelan Arsitektur *As-is* 47
3.6.2 Analisis Arsitektur *As-is* 49
3.6.3 Perancangan Arsitektur *To-be* 50

BAB IV ANALISIS DAN PEMBAHASAN 52
4.1 Analisis Arsitektur *As-Is* 52
4.1.1 Analisis Arsitektur Bisnis *As-Is* 52
4.1.1.1 Identifikasi Bisnis Angkutan Penumpang Perusahaan 53
4.1.1.2 Komponen Strategi Bisnis 53
4.1.1.3 *Value Chain* Bisnis Perusahaan 54
4.1.1.4 Struktur Organisasi Bisnis 55
4.1.1.5 *Service* Bisnis *As-is* 57
4.1.2 Analisis Arsitektur TI *As-Is* 59
4.1.2.1 Identifikasi Sumber Daya TI Teknis 60
4.1.2.1 Identifikasi Sistem *E-ticketing As-Is* 60
4.1.2.2 Identifikasi Sumber Daya Infrastruktur 60
4.1.2.2 Identifikasi Sumber Daya TI Non-Teknis 62
4.2 Perancangan Arsitektur *To-be* 63
4.2.1 Proses Bisnis *To-be* 63
4.2.1.1 Diagram *Use case* Bisnis *To-be* 63
4.2.1.2 *Activity Diagram* Proses Bisnis *To-be* 74
4.2.2 *Service* Bisnis *To-be* 96
4.2.3 Perancangan Arsitektur Aplikasi *To-be* 96
4.2.3.1 Diagram *Class* 97
4.2.3.2 Perancangan *Service* Aplikasi *To-be* 104
4.2.3.2.1 *Service Orchestration* 104
4.2.3.2.2 *Service Core* Aplikasi *To-be* 105
4.2.3.2.3 *Service Non-Core* Aplikasi *To-be* 112
4.2.3.2.4 *Service* Eksternal Aplikasi *To-be* 117
4.2.4 Perancangan Arsitektur Infrastruktur *To-be* 118
4.2.4.1 Topologi Jaringan 119
4.2.4.2 Kemanan Jaringan 121
BAB V KESIMPULAN DAN SARAN 122

5.1 Kesimpulan 122

5.2 Saran 122

DAFTAR PUSTAKA

DAFTAR GAMBAR

halaman

Gambar 2.1 Konsep *Service Oriented Architecture* 7

Gambar 2.2 Sistem Aplikasi berkomunikasi dengan API 8

Gambar 2.3 Alur *Request* dari *Service Requestor* ke *Service Provider* 12

Gambar 2.4 SOAP 14

Gambar 2.5 SOP *Request* dan *Response* 15

Gambar 2.6 WSDL 15

Gambar 2.7 Gambaran Arsitektural Monolitik dan *Microservice* 17

Gambar 2.8 Arsitektur SOA 19

Gambar 2.9 Arsitektur *Microservice* 20

Gambar 2.10 *Framework* SOEA 21

Gambar 2.11 Arsitektur Infrastruktur menggunakan pendekatan SOA 23

Gambar 2.12 *The Generic Value Chain* 30

Gambar 2.13 Diagram *Activity* 31

Gambar 2.14 Diagram *Use case* 31

Gambar 2.15 *Use case Description* 32

Gambar 2.16 *Domain Class Diagram* 33

Gambar 2.17 *First Cut Design Class Diagram* 33

Gambar 2.18 *System Sequence Diagram* 34

Gambar 2.19 *First Cut Sequence Diagram* 35

Gambar 2.20 *Updated Class Diagram* 36

Gambar 2.21 Diagram Infrastruktur *E-Ticketing* untuk Sistem Angkutan Massal 38

Gambar 3.1 Tahapan Metodologi Penelitian 47

Gambar 4.1 Konteks Bisnis Perusahaan 53

Gambar 4.2 *Value Chain* Bisnis Perusahaan 54

Gambar 4.3 Struktur Organisasi Kantor Pusat PT Railink 55

Gambar 4.4 Struktur Organisasi PT Railink Cabang Medan 56
Gambar 4.5 Struktur Organisasi PT Railink Cabang Jakarta 56
Gambar 4.6 Diagram *Use Case* Proses Bisnis *E-Ticketing* Kereta Api *As-Is* 58
Gambar 4.7 Topologi Jaringan PT Railink Kantor Pusat dan Cabang Jakarta 61
Gambar 4.8 Topologi Jaringan PT Railink Cabang Medan 62
Gambar 4.9 Diagram *Use Case* Proses Bisnis *E-Ticketing* Kereta Api *To-Be* 64
Gambar 4.10 Diagram *Activity* Pemesanan Melalui *Vending Machine* 75
Gambar 4.11 Diagram *Activity* Pemesanan Melalui *Web* Reservasi 77
Gambar 4.12 Diagram *Activity* Pemesanan Melalui Aplikasi *Mobile* 79
Gambar 4.13 Diagram *Activity* Pembayaran Melalui Vending *Machine* 82
Gambar 4.14 Diagram *Activity* Pembayaran Melalui *Web Reservasi* 84
Gambar 4.15 Diagram *Activity* Pembayaran Melalui Aplikasi *Mobile* 86
Gambar 4.16 Diagram *Activity* Proses *Gate-In* 88
Gambar 4.17 Diagram *Activity* Proses *Gate-Out* 89
Gambar 4.18 Diagram *Activity* Proses Pembatalan Melalui *Vending Machine* 90
Gambar 4.19 Diagram *Activity* Proses Penjadwalan 92
Gambar 4.20 Diagram *Activity* Proses *Setting* Tarif 93
Gambar 4.21 Diagram *Activity* Proses *Web Reporting* 95
Gambar 4.22 Diagram *Deployment* Sistem *E-Ticketing* Kereta Api 96
Gambar 4.23 Diagram *Class* Sistem *E-Ticketing* Kereta Api 98
Gambar 4.24 Diagram *Component Service Orchestration* 104
Gambar 4.25 Diagram Alur Global Operasional Sistem *E-Ticketing* Kereta Api 119
Gambar 4.26 Topologi Jaringan Infrastuktur 120

DAFTAR TABEL

halaman

Tabel 2.1 Fase, Langkah, dan output aktivitas dari *framework* SOEA 24

Tabel 3.1 *Milestone* Sejarah PT Railink 43

Tabel 4.1 Deskripsi dari Proses Bisnis dari *Value Chain* Bisnis Perusahaan 55

Tabel 4.2 Identifikasi *Gap* Kebutuhan Proses Bisnis 59

Tabel 4.3 Identifikasi Infrastruktur Pendukung *E-ticketing* As-Is 60

Tabel 4.4 Identifikasi Kondisi SDM TI Non-Teknis 62

Tabel 4.5 Deskripsi Definisi Aktor Diagram *Use Case To-be* 64

Tabel 4.6 Deskripsi Definisi *Use Case* pada Diagram *Use Case* To-be 65

Tabel 4.7 Deskripsi Diagram *Class* 99

Tabel 4.8 Parameter *request service mandatory* 105

BAB I

PENDAHULUAN

1.1 Latar Belakang

PT Railink sebagai sebuah perusahaan swasta yang merupakan anak perusahaan dari PT Kereta Api Indonesia (Persero) dengan PT Angkasa Pura II (Persero) bertindak selaku operator dari Kereta Api (KA) Bandara. KA Bandara diperuntukkan khusus untuk membawa penumpang yang akan menuju bandara atau sebaliknya. Tahun 2013, PT Railink telah mengoperasikan KA Bandara di Bandara Internasional Kualanamu Sumatera Utara dan direncanakan pada tahun 2018 akan mengoperasikan KA Bandara di Bandara Internasional Soekarno-Hatta Jakarta.

Dalam mengoperasikan KA Bandara dan menunjang bisnis usaha terutama dalam angkutan penumpang, PT Railink melakukan bisnis penjualan tiket yang diterapkan di stasiun-stasiun pemberangkatan maupun *channel-channel* pemesanan dan pembayaran tiket secara elektronik melalui *e-ticketing*. Transaksi penjualan tiket elektronik (*e-ticketing*) diterapkan secara non-tunai dengan penumpang yang akan membeli tiket terlebih sudah terlebih dahulu memesan dan membayar tiket pada melalui aplikasi *web* dan *mobile*, sehingga penumpang sudah memiliki kode pemesanan (*booking*) dan kode pembayaran yang akan digunakan sebagai validasi di stasiun pemberangkatan pada saat penumpang tersebut akan berangkat.

Dalam implementasi sistem *e-ticketing* tersebut dibutuhkan sebuah teknologi yang memungkinkan bentuk akhir dari sebuah program atau aplikasi komputer berupa sebuah *service* atau fungsi yang melakukan sebuah tugas atau proses spesifik yang dikenal dengan istilah *web service*. Dimana dengan adanya teknologi *web service*, memungkinkan perpaduan fungsi-fungsi dalam membangun sebuah program aplikasi tanpa bergantung lagi pada sistem operasi maupun bahasa pemrograman yang digunakan.

Modul-modul *e-ticketing* yang dibangun dan diintegrasikan diantaranya modul pemesanan, pembayaran (antar bank dan *payment gateway)*, *voucher*, dan lain-lain. Dengan adanya kebutuhan integrasi modul, sistem *e-ticketing* yang dibangun memerlukan

dukungan integrasi bisnis berbagai layanan (*service*) dari berbagai sumber informasi dan transaksi, interoperabilitas antar layanan, ketersediaan yang tinggi (*availability*), memenuhi aspek resiliensi supaya dapat beradaptasi serta bertahan terhadap perubahan, kesalahan, dan kerusakan infrastruktur serta memiliki kemampuan menyesuaikan kebutuhan baru yang belum tersedia sebelumnya (*scalability*), maka diperlukan sebuah *pattern* arsitektur pengembangan perangkat lunak yang mampu mengakomodasi kebutuhan dan faktor kualitas dari perangkat lunak tersebut dengan arsitektur sistem berbasis *service (service-oriented)*. Arsitektur ini lebih dikenal dengan nama *microservice*. *Microservice* sendiri merupakan salah satu pengembangan dari pendekatan *Service Oriented Architure* (SOA) yang secara lebih spesifik berhubungan dengan pembangunan layanan bisnis bebas yang dapat dikombinasikan menjadi proses bisnis pada level tinggi dan solusi dalam konteks *enterprise*.

Sebuah perancangan arsitektur *enterprise* untuk sistem *e-ticketing* berbasis *service* diperlukan sebagai sebuah pendekatan perancangan arsitektur yang memodularisasi sistem informasi menjadi *service-service* kecil dengan cara mengelompokkan proses-proses bisnis yang ada pada *e-ticketing*. Dari masalah-masalah ini, peneliti ingin membuat suatu rancangan arsitektur *enterprise* berbasis *microservice* untuk sistem *e-ticketing* kereta api. Kerangka yang digunakan adalah *Service Oriented Enterprise Architecture* (SOEA) yang fokus pada tiga lapisan arsitektur: bisnis, sistem informasi, dan infrastuktur untuk menentukan arsitektur *as-is* dan *to-be* sistem *e-ticketing* kereta api. Maka dari peneliti mengambi judul penelitian “Perancangan Arsitektur Sistem Tiket Elektronik Kereta Api Menggunakan Kerangka *Service Oriented Enterprise Architecture* (Studi Kasus: PT Railink)”.

1.2 Rumusan Masalah

Dari latar belakang masalah di atas, maka dapat dirumuskan beberapa masalah sebagai berikut:

1. Bagaimana menganalisis arsitektur bisnis *as-is e-ticketing* kereta api?
2. Bagaimana menganalisis arsitektur teknologi informasi *as-is e-ticketing* kereta api?

1.3 Tujuan

Tujuan dirancangnya sistem *e-ticketing* kereta api adalah untuk:

1. Menghasilkan rancangan arsitektur bisnis *to-be enterprise e-ticketing* kereta api.
2. Menghasilkan rancangan arsitektur teknologi informasi *to-be enterprise e-ticketing* kereta api.

1.4 Batasan Masalah

Untuk memperjelas permasalahan yang akan dibahas pada tesis ini, maka lingkup masalah dibatasi pada:

1. Sistem *e-ticketing* yang dikembangkan hanya mencakup modul-modul sebagai berikut:
   a. Modul penjadwalan,
   b. Modul pemesanan (reservasi),
   c. Modul pembayaran,
   d. Modul pembatalan,
   e. Modul tarif, dan
   f. Modul *reporting*.
2. Arsitektur *enterprise* yang akan digunakan di dalam penelitian menggunakan arsitektur berbasis *microservice* yang hanya membahas dari segi *layer* arsitektur dan fase *as-is* dan *to-be* dari setiap *layer* arsitektur.
3. Penelitian dilakukan terbatas analisis dan perancangan arsitektur *enterprise*, tidak sampai pada tahap implementasi.
4. Fokus penelitian pada perancangan arsitektur *e-ticketing* jenis kereta api lokal dan jarak jauh.

1.5 Kegunaan Hasil

Hasil yang diharapkan dari perancangan ini adalah:

1. Memberikan kemudahan dalam proses pengembangan *e-ticketing* kereta api khususnya kereta api di Indonesia
2. Memberikan *guideline* untuk membentuk integritas informasi pada sistem *e-ticketing* kereta api.

3. Memberikan *blueprint* arsitektur *enterprise* agar menjadi landasan pengembangan sistem *e-ticketing* kereta api di Indonesia.

BAB II

LANDASAN TEORI

2.1 Pengertian Arsitektur *Enterprise*

Pengertian arsitektur *enterprise* menurut Riverton merupakan sebuah mekanisme yang dilakukan untuk menjamin sumber daya informasi suatu organisasi atau perusahaan digunakan dalam mendukung strategi organisasi tersebut. Sasaran dan tujuan organisasi diuraikan untuk mendapatkan kebutuhan informasi dan komunikasi bagi organisasi tersebut dan kebutuhan ini digunakan sebagai dasar untuk membuat rencana arsitektur *enterprise*, sehingga diharapkan pada tahap akhir waktu perencanaan, tujuan yang diinginkan dapat dicapai dengan baik. Arsitektur *enterprise* merupakan salah satu disiplin sistem informasi yang memiliki definisi sebagai berikut:

1. Pendekatan logis, komprehensif, dan holistik dalam melakukan perancangan dan implementasi sistem dan komponen sistem secara bersama-sama (Parizeau, 2002).
2. Cetak biru pemetaan hubungan antar komponen dan semua orang yang bekerja di dalam perusahaan secara konsisten untuk meningkatkan kerja sama atau kolaborasi, serta koordinasi diantaranya (Ward, John and Peppard, Joe, 2002).
3. *Enterprise Architecture* (EA) adalah deskripsi dari misi *stakeholder* yang didalamnya terdapat informasi, fungsionalitas, lokasi organisasi dan parameter kinerja. EA menggambarkan rencana untuk mengembangkan sebuah sistem atau sekumpulan sistem (Osvalds, 2001).

Dengan memahami definisi di atas, dapat dinyatakan definisi sebagai berikut: arsitektur *enterprise* adalah kumpulan prinsip, metode dan model yang digunakan untuk merancang dan mengimplementasikan struktur organisasi *enterprise*, proses bisnis, sistem informasi dan infrastrukturnya. Dalam pembahasan arsitektur *enterprise* yang akan digunakan dalam mengembangkan sistem tiket elektronik kereta api, kerangka yang akan digunakan adalah kerangka *Service Oriented Enterprise Architecture* (SOEA)

2.2 *Service Oriented Architecture* (SOA)

Menurut Thomas Erl (2005), *Service Oriented Architecture* (SOA) adalah bagian utama dari *service computing platform* yang membawa konsep, teknologi, dan tantangan baru. Menurut Thomas Erl, ada tiga hal penting yang menjadikan sebuah infrastruktur dapat disebut sebagai *service oriented architecture*, yaitu logika bisnis yang dienkapsulasi sebagai *service*, dan proses komunikasi antar *service* dengan menggunakan *message*. Dalam hal ini, *service layer* akan menjembatani hubungan antara *business logic* dan *application logic*.

Menurut Kunal Mittal (2007), *Service Oriented Architecture* adalah sebuah kumpulan yang terdiri atas *tools*, teknologi, *framework*, dan *best practice* yang memudahkan implementasi sebuah *service* secara cepat. Proses dalam mengimplementasi SOA menggunakan metodologi yang mengidentifikasikan *service* yang dapat dipergunakan kembali *(reusable)* dalam aplikasi dan organisasi suatu perusahaan.

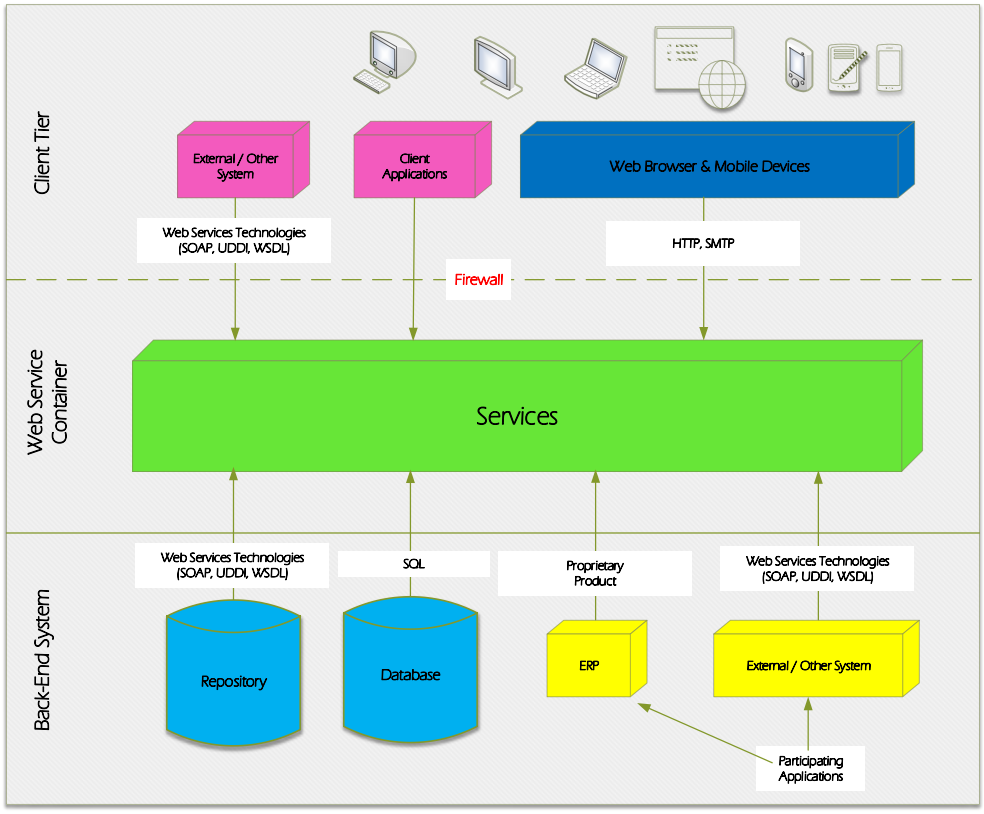
Arsitektur SOA difokuskan untuk mengidentifikasi, membangun, mengubah, dan memelihara proses bisnis suatu perusahaan sebagai sekumpulan *service*. Teknologi yang menggunakan SOA digunakan untuk mengurangi kompleksitas dalam membangun sebuah aplikasi atau *software*. SOA dapat mengantisipasi isu penggunaan *software* terdistribusi, *plarform* yang berbeda-beda, dan integrasi aplikasi.

Dapat disimpulkan bahwa SOA adalah suatu cara mengorganisir *software* sehingga organisasi dapat dengan cepat merespon perubahan kebutuhan. Teknologi tersebut berdasarkan *service* (layanan) yang terdiri dari unit-unit berdasarkan kebutuhan dari *software* yang berjalan pada jaringan.

Service sendiri merupakan komponen umum yang digunakan oleh beberapa sistem aplikasi (*reusable*). *Service* dapat berupa modul program, aplikasi, atau gabungan dari beberapa aplikasi yang berhubungan. SOA merepresentasikan suatu model dimana fungsi-fungsi dibagi menjadi beberapa unit-unit terpisah yang lebih kecil, yang dapat didistribusikan melalui jaringan dan dapat dikombinasikan dan digunakan secara bersama-sama untuk menciptakan aplikasi. *Service-service* tersebut berkomunikasi satu sama lain dengan cara mengirim data dari satu *service* ke *service* lainnya, atau dengan

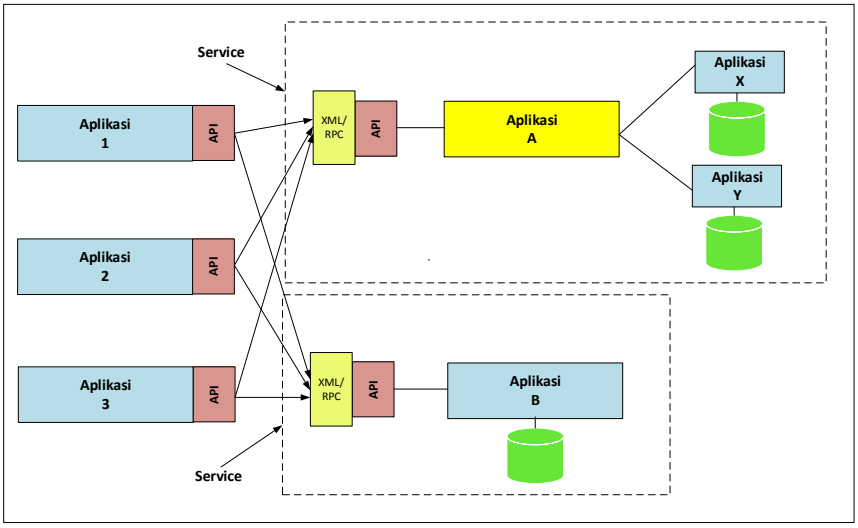
mengkoordinasikan suatu aktivitas antara dua atau lebih *service*. Sehingga SOA memungkinkan *service* yang *interoperable*, yang berarti *service-service* tersebut dapat berkomunikasi satu sama lain, meskipun pada implementasinya dibuat dengan bahasa pemrograman yang berbeda atau diakses melalui *transport protocol* yang berbeda yang memungkinkan pengintegrasian aset-aset sistem aplikasi dari suatu perusahaan.

Ilustrasi konsep dari *Service Oriented Architecture* (SOA) dapat dilihat pada Gambar 2.1:



Gambar 2.1
Konsep *Service Oriented Architecture*

Service merupakan komponen umum yang digunakan oleh beberapa sistem aplikasi (*reusable*). *Service* dapat berupa modul program, aplikasi, atau gabungan dari beberapa aplikasi yang berhubungan. *Service* direalisasikan dengan menambahkan *interface* (*wrapper*) pada satu atau sekelompok sistem aplikasi. Sistem aplikasi berkomunikasi dengan *service interface* melalui *Application Programming interface* (API).



Gambar 2.2
Sistem aplikasi berkomunikasi dengan API

Service merupakan infrastruktur karena:

1. Dipakai bersama oleh struktur di atasnya (sistem-sistem aplikasi).
2. Relatif lebih stabil/statis dibandingkan aplikasi-aplikasi penggunanya.
3. Memiliki *life cycle* tersendiri.
4. Dikelola (dirancang, dikembangkan, dan dipelihara) oleh pihak yang berbeda dengan pengelola aplikasi di atasnya.
5. Dapat di-*outsource*.
6. Menerapkan *Service Level Agreement*.

Dengan menyediakan *service*, keuntungan yang didapat antara lain:

1. Dapat memanfaatkan ulang keahlian.
2. Meningkatkan kecepatan (*agility*) pengembangan sistem aplikasi.
3. Menurunkan kompleksitas pengembangan sistem aplikasi.

2.2.1 Komponen SOA

SOA memiliki tiga komponen penyusun utama dalam fondasinya, yaitu:

1. *Business Architecture*

Business architecture didasarkan pada strategi bisnis, objektif, prioritas, dan proses. Hal ini adalah bagian terpenting dalam kesuksesan implementasi dari SOA. Satu keuntungan utama dari SOA adalah penggunaan kembali bisnis proses yang

menyediakan potensi tingkat keuntungan yang lebih tinggi karena adanya potensi penggunaan ulang dari infrastruktur atau komponen data. Hal ini juga mengikutsertakan proses bisnis sebagai dasar dari implementasi aplikasi bisnis

2. *Infrastructure Architecture*

   Arsitektur ini memfokuskan SOA terhadap semua aspek dari infrastruktur dari jaringan, *server*, *data center*, *firewall*, menjadi infrastruktur aplikasi, *security*, *monitoring*, *middleware*, dan lainnya. Arsitektur ini menjadikan infrastruktur menjadi sebuah *engine* pendukung SOA.

3. *Information* dan *Data Architecture*

   Kesepakatan ini ditandai dengan mengidentifikasikan *Key Performance Indicators* dan informasi yang dibutuhkan untuk mendukung perusahaan. *Data Architecture* berhubungan dengan *data modelling* untuk dilakukan manipulasi data dan penentuan kualitas dari data.

2.2.2 Keuntungan SOA

Perancangan arsitektur berbasis SOA memiliki keuntungan sebagai berikut:

1. Kecepatan

   Dalam SOA, proses bisnis dipecah dan disederhanakan dalam bentuk *service* yang lebih kecil. Ketergantungan yang ada antar *service* harus diminimalisir, sehingga apabila terjadi perubahan pada suatu proses bisnis, cukup *service* yang berkaitan saja yang mengalami perubahan, tidak perlu seluruh sistem. Dengan hal tersebut, sistem dapat merespon perubahan dengan cepat.

2. *Real-time resposive*

   Dalam *service-service* tersimpan *business rules* dan batasan-batasan dalam bisnis. Dan *service-service* ini disimpan dan dikelola dalam sebuah server aplikasi yang disebut *Enterprise Service Bus* (ESB). Sehingga berbagai jenis aplikasi dapat mengakses *business rules* tersebut. Apabila terjadi suatu perubahan terhadap *business rules*, ESB akan mengelolanya secara otomatis. Sehingga *business rules* yang baru akan berlaku saat itu juga.

3. Penghematan

   Meskipun pada awalnya implementasi membutuhkan biaya yang besar, dengan implementasi SOA memungkinkan pengembangan sistem yang terpusat, sehingga ada banyak *resource* yang bisa dikurangi. Dengan demikian juga akan mengurangi biaya.

4. *Channel independent*

   Bisnis berkaitan dengan banyak pihak, baik pelanggan maupun *supplier*. Berbagai pihak yang behubungan dengan organisasi tentu saja memungkinkan adanya berbagai jenis aplikasi yang berbeda-beda. Dengan adanya *service* dan ESB, memungkinkan untuk berbagai aplikasi tersebut untuk mengakses *business rules* yang telah didefinisikan. Sehingga pihak-pihak yang berkaitan dengan organisasi tidak tergantung lagi terhadap suatu aplikasi tertentu.

5. Waktu pengembangan yang lebih singkat

   Dalam SOA, bisnis proses yang dipecah dalam bentuk *service* yang lebih kecil memungkinkan perubahan dan pengembangan pada *service* yang tertentu saja. Karena pengembangan dilakukan secara terfokus, waktu yang dibutuhkan menjadi lebih sedikit.

6. Mengurangi duplikasi

   Service dalam SOA dikelola dalam server aplikasi yang disebut ESB. Karena *service* dikelola secara terpusat, hal ini akan mengurangi kemungkinan adanya duplikasi sistem. Selain itu, bentuk *service* yang memungkinkan *reusability* juga mengurangi adanya fungsi yang sama yang ada di dalam sebuah sistem.

2.2.3 Prinsip Dasar *Service*

Berikut prinsip-prinsip dasar dari *service* dalam SOA:

1. *Services are reusable*

   Service dapat memiliki satu atau banyak *operation*s, tiap tiap *operation*s dapat diakses oleh banyak *service requestor*s.

2. *Service share formal contract*

   Kontrak *service* menyediakan definisi formal dari:

a. *Endpoint* dari *service*
b. *Operation*s yang terdapat pada tiap-tiap *service*
c. Setiap *input* dan *output message* yang didukung oleh tiap-tiap *operation*s
d. Aturan dan karakteristik dari *service* dan *operation-operation* yang terdapat di dalamnya.

3. *Service*s *are loosely coupled*

   Loose coupling adalah kondisi dimana sebuah *service* mendapatkan informasi dari *service* lainnya dalam kondisi independen dari *service* tersebut. Kondisi *loose coupling* dapat diperoleh dari penggunaan kontrak *service* yang memungkinkan *service*s berinteraksi tanpa mendefiniskan parameter-paramenter yang diperlukan terlebih dahulu.

4. *Service*s *abstract underlying logic*

   Service tidak memperlihatkan bagaimana *logic* diimplementasi di dalamnya.

5. *Service*s *are composable*

   Konsep yang mendukung *composability* adalah konsep orkestrasi, yaitu proses *service-oriented* yang dikontrol oleh sebuah induk proses yang terdiri dari beberapa anggota *service*s.

6. *Service are autonomous*

   Service memiliki hak penuh terhadap semua logic yang dienkapsulasi.

7. *Service are stateles*

   Service tidak memiliki status tertentu terkait dengan aktifitas yang dilakukan.

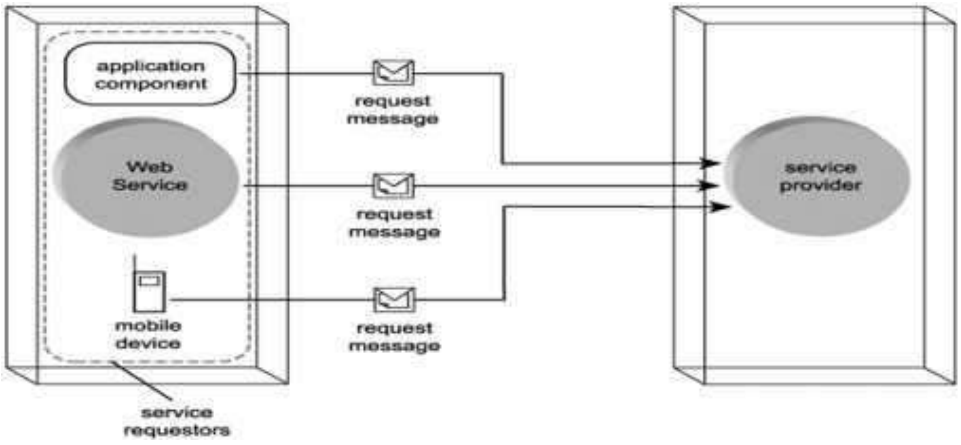
2.2.4 *Web Service*

Sesuai dengan namanya, *web service* hanya menyediakan *service* atau layanan. *Web service* dibuat bukan untuk berinteraksi langsung kepada *client*, tetapi untuk menyediakan layanan yang akan dipanggil oleh aplikasi. *Web service* dapat didefinisikan sebagai aplikasi yang dibuat agar dapat dipanggil atau diakses oleh aplikasi lain melalui internet dengan menggunakan XML sebagai format pengiriman pesan.

Pada kondisi dimana *operating system* dan bahasa pemrograman beraneka ragam jenisnya, timbul masalah dimana pertukaran data antar perangkat yang menggunakan

aplikasi dan *platform* yang berbeda. Inilah alasan utama menggunakan *web service*, *web service* memungkinkan penggunaan *platform* yang berbeda dapat bertukar data dan informasi dengan mudah.

Pada prinsipnya peranan *service* dibagi menjadi dua, yaitu *service provider* dan *service requestor*. *Service provider* adalah *service* yang menyediakan layanan yang akan digunakan dan *service requestor* adalah *service* yang meminta *service* tersebut berjalan. Alokasi *provider* dan *requestor* tidak bersifat mutlak. Sebuah *service provider* dapat berubah menjadi *service requestor*, tergantung dari peran yang diberikan oleh proses bisnis yang terkait.



Gambar 2.3
Alur request dari *Service Requestor* ke *Service Provider*

Requestor dari sebuah *service* tidak hanya berasal dari sebuah *service*, tetapi juga dapat berasal dari aplikasi *native* atau *mobile*. Pada Gambar 2.3 di atas, dijelaskan bahwa *service requestor* meminta *service* kepada *service provider* dengan menggunakan *request message* berbentuk amplop. Amplop tersebut adalah penggambaran dari SOAP yang berupa struktur bahasa dari *Extensible Markup Language* (XML) yang digunakan untuk berkomunikasi oleh *service requestor* dan *service provider*. Gambar 2.3 adalah sebuah implementasi sederhana dari sebuah *service requestor* kepada *service provider*.

Peran *intermediaries* (penghubung) akan sering dijumpai seiring semakin kompleks-nya sebuah sistem. Semakin kompleks sebuah sistem, maka akan semakin banyak *service*s yang digunakan dan hubungan antar *service* akan semakin fleksibel.

Kelebihan dari *web service* adalah:

1. Lintas *platform*

   Dengan menggunakan *web service*, komputer yang memiliki sistem operasi yang berbeda dapat saling berkomunikasi.

2. *Language independent*

   Akses kepada *web service* tidak terbatas pada hanya satu bahasa pemrograman saja. Sebagai contoh, *web service* yang dibangun dengan PHP dapat diakses oleh PHP, JSP, Delphi dan VB.NET.

3. Jembatan penghubung dengan *database*

   Web service dapat menjadi penghubung antara aplikasi dengan *database*. Dengan menggunakan *web service*, aplikasi tidak memerlukan *driver database* dan tidak perlu mengetahui *database* yang digunakan oleh *server* serta bagaimana struktur *database* tersebut jika ingin mengaksesnya.

4. Mempermudah proses pertukaran data

   Pertukaran data antara dua perusahaan yang memiliki *platform* yang berbeda dapat terjadi dengan menggunakan *web service*.

5. Penggunaan kembali komponen aplikasi

   Sebuah layanan dapat melayani sebuah fungsi untuk beberapa pengguna layanan.

Beberapa komponen pendukung *web service* adalah:

1. *Extensilble Markup Languange* (XML)

   Extensilble Markup Languange (XML) diciptakan pada akhir tahun 60-an oleh World Wide *Web* Consortium (W3C). XML menjadi populer pada *e-business* pada akhir 90-an, pada saat traksaksi *online* menggunakan bahasa *server-side scripting* menjadi tren. Arsitektur data yang direpresentasikan XML mencerminkan pemahaman dasar terhadap SOA. Dalam arsitektur data tersebut XML membangun format dan struktur dari *message* yang berjalan antar *service*.

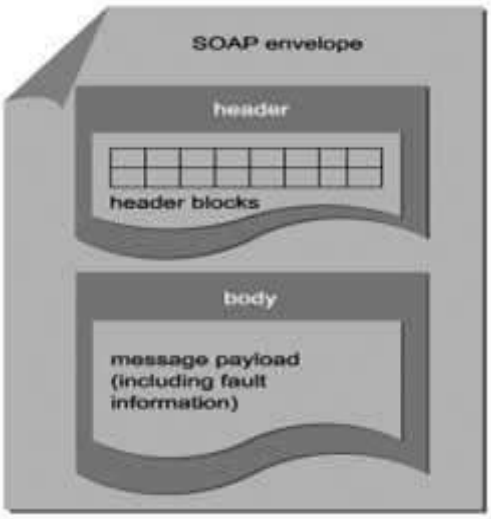
XML bersifat *platform independent*, informasi pada XML dapat diakses oleh *platform* apapun selama aplikasi tersebut dapat menterjemahkan *tag* XML.

2. *Simple Object Access Protocol* (SOAP)

   Simple Object Access Protocol (SOAP) merupakan format standar dokumen yang berbentuk XML yang digunakan untuk melakukan proses *request* dan *response* antar *web service* dengan aplikasi yang memanggilnya (Lucky, 2008). Komunikasi antar *service* yang menggunakan XML harus distandarisasi dengan menggunakan SOAP, sehingga komunikasi antar *service* akan lebih teratur.

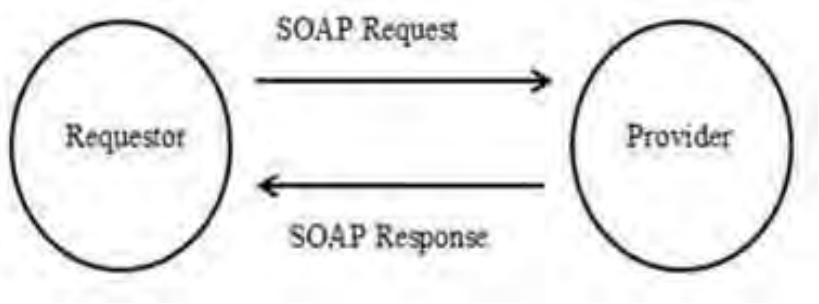
   Tujuan utama penggunaan SOAP adalah melakukan standarisasi format *message* yang berjalan antar *service*. Struktur dari SOAP terdiri dari SOAP *envelope, header* dan *body.*

   Secara arti kata, *envelope* berarti amplop, yaitu dimana disimpan sebuah surat, pada SOAP *envelope* menyimpan seluruh informasi *message* yang mengalir. Informasi penting disimpan pada *body*, dimana terdiri dari data berformat XML. Dan *header* dimana informasi yang bersifat tambahan disimpan.



Gambar 2.4
SOAP (Erl, 2005)

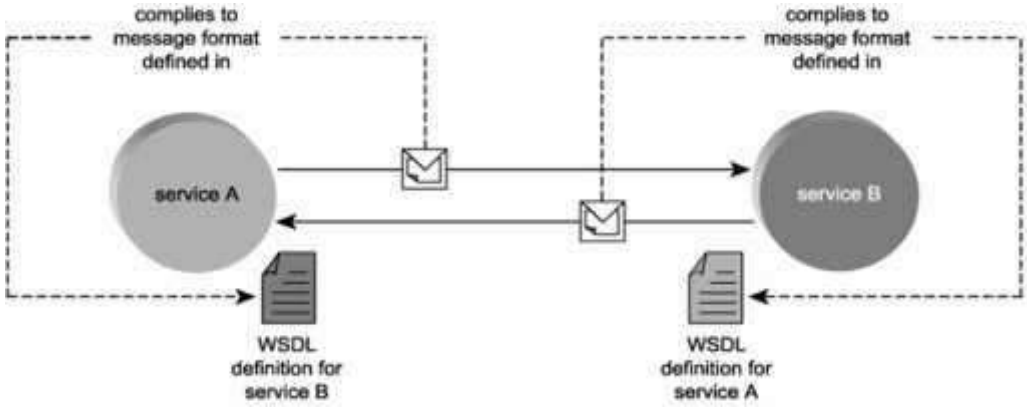
Dokumen SOAP yang digunakan untuk melakukan sebuah permintaan (*request*) oleh *service requestor* disebut SOAP *request* dan dokumen SOAP yang diterima dari *service provider* disebut SOAP *response*



Gambar 2.5
SOAP *Request* dan *Response* (Erl, 2005)

3. *Web Service Description Language* (WSDL)

   Web Service Description Language (WSDL) adalah sebuah dokumen dalam format XML yang isinya menjelaskan informasi detail sebuah *web service*. Di dalam WSDL dijelaskan *method-method* apa saja yang tersedia dalam *web service*, parameter apa saja yang diperlukan untuk memanggil sebuah *method*, dan apa hasil atau tipe data yang dikembalikan oleh *method* yang dipanggil tersebut. (Lucky, 2008)



Gambar 2.6
WSDL (Erl, 2005)

Pada Gambar 2.6, dijelaskan hubungan antara *service* A dan *service* B. pada SOAP *request* yang berasal dari *service* A terdapat WSDL *definition* untuk *service* B dan pada SOAP *response* yang berasal dari *service* B terdapat WSDL *definition* untuk *service* A.

Dalam pengertian sederhana, WSDL dapat berupa sebuah brosur dan formulir sebuah perusahaan. Brosur menjelaskan layanan apa saja yang dapat diberikan oleh perusahaan tersebut dan keuntungan memakai layanan dari perusahaan. Dan formulir

merupakan informasi yang harus diisi oleh pengguna layanan yang akan digunakan oleh perusahaan demi memberikan pelayanan yang maksimal.

2.3 Arsitektur *Microservice*

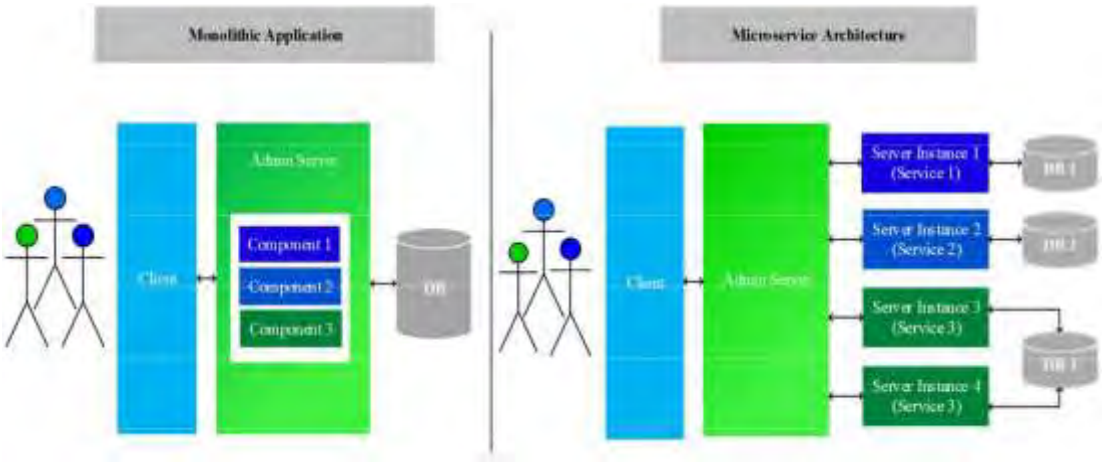
Software dikembangkan sebagai serangkaian komponen otomatis dari suatu aplikasi seperti komponen *web service*, *messaging,* dan sebagainya yang dipasang secara independen pada sisetm yang berbeda-beda. Dengan mengisolasi komponen-komponen fungsional ini ke dalam instalasi mesin yang berbeda-beda, ukuran aplikasi monolitik menjadi berkurang dan menjadikan komunikasi ke aplikasi utama dan *interface* satu sama lain yang *loose couple*, melalui protokol sinkronus *Hypertext Transfer Protocol* (HTTP) atau pesan asinkronus. Pendekatan pengembangan arsitektur *software* seperti inilah yang kemudian dikenal dengan arsitektur *microservice*.

2.3.1 Definisi *Microservice*

Menurut James Lewis dan Martin Flower, *microservice* dapat didefinisikan sebagai berikut:

”*An approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies*.”
(Lewis and Fowler, 2014)

Arsitektur *microservice* memecah aplikasi yang kompleks ke dalam *service-service* otonom yang kecil. Proses-proses independen saling berkomunikasi melalui berbagai API yang juga independen. *Service-service* tersebut saling terpisah dan fokus pada tugas atau pekerjaan yang kecil, sehingga hal tersebut menjadikannya sebagai sebuah pendekatan modular dalam membangun sebuah sistem. *Service-service* tersebut mudah diganti, dirancang untuk memenuhi kebutuhan bisnis yang spesifik, dan dapat diimplementasikan menggunakan teknologi yang berbeda-beda. Selain itu, karena *microservice* bersifat otonom, *service* tersebut dibangun dan didistribusikan menggunakan metodologi pengembangan *software Continuous Delivery* (CD).



Gambar 2.7
Gambaran Arsitektural Monolitik dan *Microservice*

Microservice dapat melakukan pengelolaan *database*nya sendiri, dan masing-masing *microservice* dapat diperbesar atau diperkecil secara terpisah baik dari sisi aplikasi maupun database, terlepas dari mircroservice yang lainnya. Dengan hal tersebut, memungkinkan perusahaan untuk melakukan peningkatan pada domain dari aplikasi enterprise tertentu yang banyak memakan sumber daya.

Setiap *microservice* berukuran kecil dan bertujuan untuk melakukan fungsi tunggal. *Microservice* bisa memanggil *microservice* lainnya. *Microservice* tersebut bersifat elastis, *resilient*, mudah dikomposisi, minimalis, dan lengkap. *Microservice* dapat dibuka melalui interface Uniform Resource Identifier (URI) yang sederhana, menerima *request* dan menghasilkan *response* dalam format UNIX.

Arsitektur mikroservice berbeda dari arsitektur SOA karena tujuan utama SOA yaitu mengintegrasikan beragam aplikasi bisnis, sedangkan setiap *service* dalam arsitektur *microservice* merupakan bagian dari keseluruhan representasi aplikasi.

2.3.2 Keterbatasan pada aplikasi monolitik

Sebuah aplikasi monolitik merupakan sebuah aplikasi enterprise yang terdiri dari layer presentasi, layer *data-access*, dan layer *server-side*. Dikarenakan arsitektur yang ketat dan kaku ini maka aplikasi monolitik memiliki sejumlah risiko dan keterbatasan sebagai berikut:

1. Skalabilitas

2. Resilien
3. Penguncian *vendor* dan teknologi
4. Masalah implementasi metodologi *agile*
5. *Change management* atau dinamisasi bisnis

2.3.3 Fitur dan Keuntungan pada arsitektur *microservice*

Dengan *microservice*, fitur dan keuntungan yang diperoleh yaitu:

1. Dekomposisi Fungsional
2. Perancangan Failure
3. Evolusioner
4. Kompatibilitas dengan metodologi *agile*
5. *Versioning Service*
6. *Monitoring Service*
7. Otomasi dengan menggunakan *Continuous Delivery*

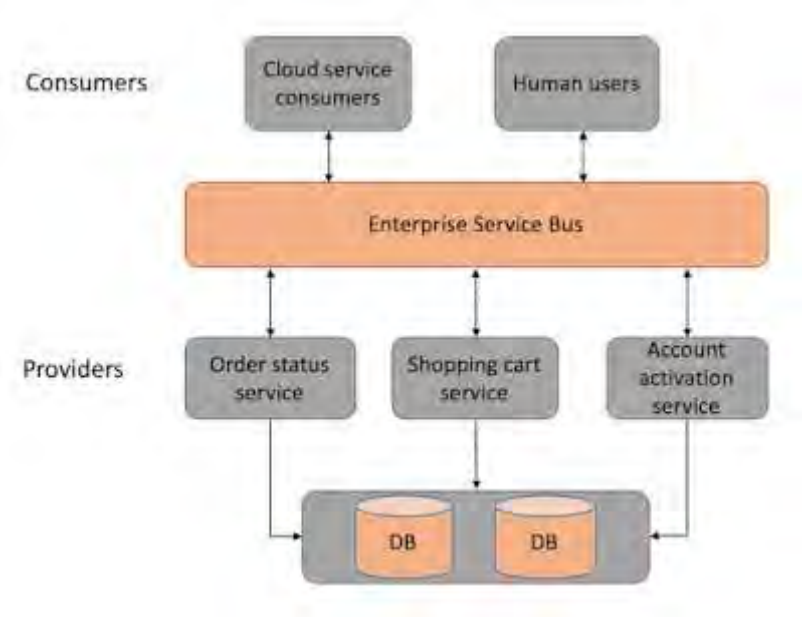
2.3.4 *Service* Oriented Architecture (SOA) dan *Microservice*

Baik arsitektur *Service Oriented Architecture* (SOA) dan arsitektur *microservice* memiliki persamaan maupun perbedaan. Penggunaan antara kedua *pattern* arsitektur tersebut akan dijelaskan pada bahasan berikut.

2.3.4.1 *Service Service* Oriented Architecture (SOA)

Service Oriented Architecture merupakan *pattern* arsitektur *software* dimana komponen-komponen aplikasi memberikan *service-service* komponen lainnya melalui sebuah protokol komunikasi dalam suatu jaringan. Komunikasi tersebut dapat berupa pengiriman data sederhana atau beberapa *service* yang berkoordinasi saling terhubung satu sama lain.

Di dalam SOA terdapat dua peranan, yakni sebagai *service* provider dan *service consumer*. *Consumer Layer* merupakan titik dimana *consumer* (seperti pengguna manusia, *service* lain atau pihak ketiga) berinteraksi dengan SOA dan *Provider Layer* berisi semua *service-service* yang didefinisikan di dalam SOA. Gambar 2.8 menggambarkan arsitektur SOA.



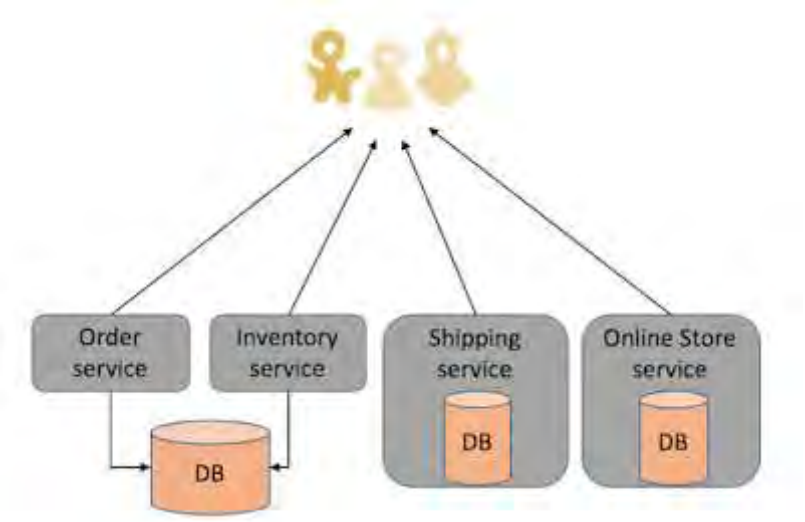
Gambar 2.8
Arsitektur SOA

Enterprise *Service* Bus (ESB) merupakan sebuah teknik arsitektur pengintegrasian yang menyediakan komunikasi dalam sebuah bus komunikasi standar terdiri dari berbagai koneksi point-to-point di antara provider dan *consumer* dan penyimpanan data dibagikan di dalam semua *service* di dalam SOA.

2.3.4.2 *Microservice*

Microservice merupakan *pattern* arsitektur software dimana aplikasi-aplikasi kompleks disusun dari proses-proses kecil, independen dan berkomunikasi satu sama lain mengggunakan API.

Microservice harus merupakan sebuah kebutuhan nyata dalam arsitektur sistem. Dalam arti sebuah *service* harus didistribusikan secara independen, atau dapat dimatikan ketika *service* tersebut tidak diperlukan, dan hal tersebut tidak memiliki dampak terhadap *service* lain. Gambar 2.9 menunjukkan arsitektur dari sebuah *microservice*.



Gambar 2.9
Arsitektur *Microservice*

Kedua arsitektur memiliki kelebihan dan kekurangan yang sama dan terdapat juga beberapa perbedaan. Pada kedua arsitektur tersebut, setiap *service* mempunyai tanggung jawab khusus, sehingga *service-service* yang dikembangkan dapat menggunakan berbagai *technology stack*. Hal ini akan membawa keberanekaragaman teknologi pada tim pengembang. Namun, masing-masing tim perlu mengetahui tentang mekanisme komunikasi standar di dalam SOA.

Di dalam *microservice*, *service-service* dapat beroperasi dan didisribusikan secara independen dari *service* lainnya tidak seperti SOA, sehingga memudahkan dalam pendistribusian *service* versi baru serta perubahan spesifikasi *service* secara mandiri.

Di dalam SOA, ESB menjadi sebuah titik tunggak kegagalan yang akan berdampak pada keseluruhan aplikasi dikarenakan semua *service* berkomunikasi melalui ESB, jika salah satu *service* lambat, dapat mengakibatkan ESB harus ditutup untuk request dari *service* tersebut. Dari hal inilah, *microservice* jauh lebih baik dalam hal *fault tolerance*. Sebagai contoh, jika terdapat sebuah kobocoran memori dalam sebuah *microservice*, maka hanya *microservice* itu saja yang terkena dampak. *Microservice* lainnya akan tetap lanjut menangani request-request. Di dalam SOA, *service-service* berbagi penyimpanan data (Gambar 2.8) sementara dalam *microservice*, setiap *service* dapat mempunyai penyimpanan data tersendiri.

Baik SOA dan *microservice*, pengembangan harus mempertimbangkan kompleksitas arsitektur dan sistem terdistribusi. Pengembang harus mengimplementasikan mekanisme komunikasi antar-*service* di antara *microservice* atau di dalam ESB. Perbedaan utama antara SOA dan *microservice* terletak pada ukuran dan lingkupnya. Ukuran *microservice* secara signifikan lebih kecil dari pada kecenderungan ukuran SOA. Di sisi lain, sebuah SOA dapat berupa sebuah arsitektur monolitik atau dapat juga terdiri dari beberapa *microservice*.

2.4 Kerangka *Service Oriented Enterprise Architecture* (SOEA)

Konsep SOA sudah menginduksi perubahan dalam metodologi EA. Kombinasi SOA dan EA, menghasilkan sebuah kerangka yang dinamakan *Service Oriented Enterprise Architecture* (SOEA), yang menyoroti hubungan sinergi antar keduanya. Dengan pendekatan ini, SOA dan EA saling melengkapi untuk dukungan yang lebih baik dalam bisnis yang *agile*.

Dari perspektif EA, SOA metupakan konsep dasar yang memperkaya pemodelan arsitektur EA pada semua *layer* arsitektur. Akibatnya, SOA dapat berperan sebagai mediator di antara elemen-elemen EA.

Gambar 2.10
Framework SOEA (Kazen, 2010)

Kombinasi EA dan SOA menghasilkan kerangka untuk mendokumentasikan aspek bisnis dan TI dengan mengutilisasi pendekatan berbasis *service*.

Akar *framework* SOEA berasal dari *Federal Enterprise Architecture Framework* (FEAF) dan *Project Management Body of Knowledge* (PMBOK). *Layer* arsitektur yang terdapat dalam SOEA adalah:

1. *Layer* Arsitektur

   Dalam *framework* SOEA, terdapat tiga *layer* arsitektural:

   a. Arsitektur Bisnis

      Arsitektur Bisnis merupakan bagian penting dan *layer* pertama dalam SOEA yang mempengaruhi layer arsitektural lainnya. Di dalam layer ini, dikombinasikan pendekatan *Business Process Management* (BPM) dengan *service orientation*. Tiga pola umum yang dimasukkan ke dalam bisnis proses yaitu:

      1) Level organisasi: level strategis yang berhubungan dengan aktivitas stratejik tingkat tinggi
      2) Interaksi: struktur *peer-to-peer* yang sesuai untuk mendukung kolaborasi dinamis di antara mitra bisnis.
      3) *Value Chain*:

         Primary: infrastruktur perusahaan, *Human Resource Management* (HRM), pelelangan dan pengembangan teknologi

         Support: Logistik internal, operasional, logistik eksternal, pemasaran, penjualan, dan pelayanan.

   b. Arsitektur Aplikasi

      Di dalam arsitektur aplikasi didefinisikan suatu pendekatan komposit aplikasi yang mirip dengan *Component Based Software* (CBS) yang fokus pada pembangunan sistem *software* yang besar dengan mengintegrasikan komponen *software* yang sudah ada.

      Aplikasi komposit mempunyai tiga *layer*:

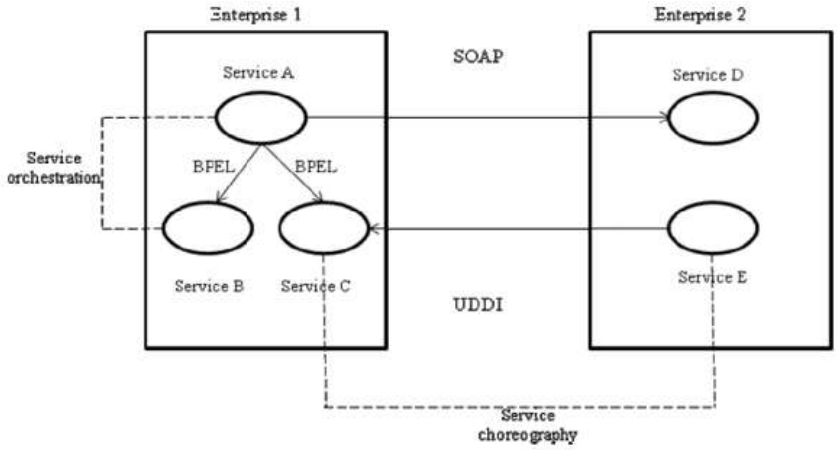
      1) *Layer user interface* dimana *form-form* dan fungsi-fungsi ditampilkan kepada *user.*

2) *Layer Choreography* yang mendefinisikan penyusunan dan pemanggilan *service* ke aplikasi komposit.

3) *Layer Service* yang merepresentasikan *service* ke aplikasi komposit.

c. Arsitektur Infrastruktur

Setelah menentukan aplikasi komposit, di dalam *layer* infrastruktur, diidentifikasi lingkungan dimana *service* aplikasi mendukung semua *service* bisnis.



Gambar 2.11
Arsitektur Infrastruktur menggunakan pendekatan SOA

Layer-layer tersebut menentukan arstitektur adanya (*as-is*) dan menjadi (*to-be*) dari suatu organisasi.

2. Fase Proyek

Tabel 2.1 menggambarkan fase, langkah, aktifitas, dan output yang diharapkan dari sebuah proyek SOEA.

Langkah-langkah dari fase tersebut, dijelaskan sebagai berikut:

a. Perencanaan Proyek

Dalam fase awal ini, semua output, ruang lingkup dan *role* proyek sudah ditentukan. Aktivitas utama dari fase ini adalah:

1) Mengorganisasikan tim proyek

2) Finalisasi ruang lingkup pryek

3) Mengembangkan *Project Managemenet Plan* (PMP)

b. Arsitektur *As-is*

Tujuan utama dari fase ini ialah menentukan kebutuhan untuk mendefinisikan arsitektur *to-be*. Tim mengambil semua informasi yang dibutuhkan tentang aplikasi dan bisnis organisasi. Fase ini terdiri dari dua tahap:

Tabel 2.1 Fase, Langkah, dan output aktivitas dari *framework* SOEA

Fase	**Tahapan**	**Aktivitas**	**Output**
Fase 0: Perencanaan Proyek		• Mengorganisasikan tim proyek • Finalisasi ruang lingkup proyek • Mengembangkan rencana manajemen proyek	• Struktur Organisasi Proyek • Pernyataan Ruang Lingkup • Project Management Plan (PMP)
Fase 1: Arsitektur As-Is	Arsitektur Bisnis As-Is	• Identifikasi keseluruhan struktur bisnis • Mendetailkan komponen-komponen strategi bisnis organisasi • Identifikasi detail struktur bisnis • Mendetailkan *service* bisnis as-is	• Model konseptual bisnis • Komponen-komponen strategi bisnis dan TI • Model fungsi bisnis, model proses dan model data • *Service* Bisnis As-Is
	Arsitektur TI As-Is	• Identifikasi sumber daya teknis TI • Identifikasi sumber daya non-teknis TI	• Aplikasi-aplikasi, Sistem Informasi, integrase, *security*, dan infrastruktur • Sumber daya dan struktur manajemen TI
Fase 2: Analisis Arsitektur As-Is			Analisis Arsitektur Bisnis dan TI As-Is
Fase 3: Arsitektur *To-be*	Arsitektur Bisnis *To-be* Arsitektur TI *To-be*		1. *Service* Bisnis *To-be* (*service* core, non-core, dan eksternal) 2. *Service* Aplikasi 3. *Service* Infrastruktur
Fase 4: Rencana Migrasi		*1. Gap Analysis* 2. Identifikasi dan Prioritas Proyek 3. Mengembangkan rencana evaluasi dan pemutakhiran 4. Pengajuan struktur manajemen TI	1. Laporan *Gap Analysis* 2. Proyek usulan 3. Rencana Evaluasi dan Pemutakhiran 4. Struktur Manajemen TI

1) Arsitektur Bisnis *As-is*

Pada tahap ini, seorang implementor harus menentukan struktur bisnis organisasi dan relasinya dengan *stakeholder.*

Aktivitas utama yang dilakukan ialah:

a) Identifikasi struktur bisnis keseluruhan.

b) Menguraikan komponen strategi bisnis.

c) Mengidentifikasi struktur bisnis mendetail.

d) Menguraikan *service* bisnis *As-Is*.

2) Arsitektur TI *As-is*

Tahapan ini terdiri dari pengidentifikasian di dalam organisasi yang menggambarkan:

a) Sumber daya TI secara teknis

b) Sumber daya TI secara non-teknis

c. Analisis Arsitektur *As-is*

Fase ini mencakup analisis arsitektur bisnis dan TI *as-is*. Tergantung dari proyeknya, aktifitas "*benchmarking*" dapat dilakukan dalam rangka mengidentifikasi langkah dengan *best practice* baik dalam bisnis dan TI.

d. Arsitektur *To-be*

Dengan menggunakan output dari fase sebelumnya sebagai sebuah input, seorang implementor akan mendokumentasikan arsitektur *to-be* dari bisnis dan TI.

1) Arsitektur Bisnis *To-be*

Berdasarkan hasil analisis arsitektur bisnis *as-is* dan *service* bisnis yang dikembangkan pada fase 1, *service* bisnis *to-be* dikembangkan pada tahap ini. Tiga jenis *service* bisnis yang diharapkan:

a) *Service core business*: berkaitan dengan proses bisnis utama (*front office*)

b) *Service non-core business*: berkaitan dengan proses bisnis pendukung (*back office*)

c) *Service external*: berkaitan dengan *stakeholder* dan lingkungan organisasi

2) Arsitektur TI *To-be*

Service TI to-be dikembangkan dalam fase ini. Fase ini terdiri dari pendokumentasian *service* aplikasi dan *service* infrastruktur. *Service* aplikasi akan diidentifikasi berdasarkan *service* bisnis, dan *service* infrastruktur akan ditentukan untuk mendukung *service* aplikasi.

e. Perencanaan migrasi

Pada fase ini, tim proyek akan mengembangakan suatu perencanaan untuk melakukan migrasi dari arsitektur *as-is* menjadi arsitektur *to-be*. Beberapa aktivitasnya adalah:

1) *Gap analysis*: jika terdapat suatu *gap* antara artsitektur *as-is* dan *to-be*, implementor harus menentukan bidang dasar *gap* baik dari sisi TI dan bisnisnya serta mengajukan sebuah *action plan*.
2) Identikasi proyek dan prioritas: dengan terdeteksinya gap, implementor akan mendefinisikan proyek, prioritas dan sekuennya, estimasi waktu, sumber daya yang diperlukan, dan lain-lain.
3) Pengembangan perencangaan evaluasi dan *updating:* dalam aktivitas ini, implementor akan mengajukan indikator performa untuk mengevaluasi progress perencanaan secara keseluruhan.
4) Mengajukan struktur manajemen TI: tim, grup atau departemen yang diperlukan untuk mengelola proyek yang diajukan dan mengevaluasi perencanaan seluruhnya.

2.5 *Value Chain*

Value Chain menggambarkan keseluruhan aktivitas yang dibutuhkan untuk menghasilkan seuatu produk barang atau jasa, mulai dari proses perancangan, input bahan-bahan mentah, proses produksi sampai dengan distribusi ke konsumen akhir serta pelayanan setelah pemasaran.

2.5.1 Pengertian *Value Chain*

Menurut Porter yang dikutip oleh David (2012:225), bisnis sebuah perusahaan paling baik dideskripsikan sebagai rantai nilai (*Value Chain*), dimana total pendapatan dikurangi total biaya semua aktivitas yang dilakukan untuk mengembangkan dan memasarkan produk atau jasa yang dihasilkan nilai. Semua perusahaan di suatu industri memiliki rantai nilai yang serupa, yang mencakup berbagai aktivitas seperti memperoleh bahan mentah, merancang produk, membangun fasilitas manufaktur, mengembangkan perjanjian kerjasama, dan menyediakan layanan konsumen. Sebuah perusahaan akan meraih keuntungan jika total pendapatan melampaui total biaya yang ditimbulkan dari penciptaan dan pengiriman produk atau jasa.

Menurut David (2012:227), analisis rantai nilai (*Value Chain analysis*-VCA) mengacu pada proses yang dengannya perusahaan menentukan biaya yang terkait dengan aktivitas organisasional dari pembelian bahan mentah sampai produksi dan pemasaran produk tersebut.

Menurut Assauri (2011:66), rantai nilai adalah suatu kumpulan yang terkait dengan aktivitas penciptaan nilai, yang dimulai dengan bahan baku dasar, yang datang dari pemasok dan bergerak ke rangkaian aktivitas penambahan nilai (*value added*), yang mencakup produksi dan pemasaran produk, berupa barang atau jasa, dan diakhiri dengan distribusi untuk dapat diterimanya produk oleh konsumen akhir.

Sedangkan menurut Pearce dan Robinson (2007:158), rantai nilai merupakan sebuah perspektif di mana bisnis dipandang sebagai rantai kegiatan dalam mengubah *input* menjadi *output* yang memberikan nilai kepada pelanggan. Sedangkan analisis rantai nilai adalah sebuah analisis yang mencoba untuk memahami bagaimana suatu bisnis dapat menciptakan nilai bagi pelanggan (*customer value*) dengan menguji kontribusi dari kegiatan yang berbeda dalam suatu perusahaan.

Berdasarkan pendapat para ahli mengenai definisi dari *Value Chain*, maka dapat disimpulkan bahwa *Value Chain* merupakan suatu proses perusahaan dalam menentukan biaya yang terkait dengan aktivitas penciptaan nilai perusahaan, dimulai pada proses *input* sampai dengan *output* serta diterimanya produk oleh konsumen akhir.

2.5.2 Tujuan *Value Chain*

Menurut David (2012:227), *Value Chain Analysis* bertujuan untuk mengidentifikasi dimana keunggulan (*advantage*) atau kelemahan (*disadvantage*) biaya rendah yang ada di sepanjang rantai nilai mulai dari bahan mentah sampai aktivitas layanan konsumen.

Menurut Porter, tujuan dari *Value Chain Analysis* adalah untuk mengidentifikasi tahap-tahap *Value Chain* di mana perusahaan dapat meningkatkan *value* untuk pelanggan atau untuk menurunkan biaya. Penurunan biaya atau peningkatan nilai tambah (*value added*) dapat membuat perusahaan lebih kompetitif.

Menurut Wisdaningrum (2012) dalam jurnalnya, *Value Chain Analysis* dapat dipergunakan untuk menentukan pada titik-titik di mana dalam rantai nilai yang dapat mengurangi biaya atau memberikan nilai tambah.

Jadi, dapat disimpulkan bahwa tujuan *Value Chain Analysis* adalah untuk mengidentifikasikan keunggulan dan kelemahan dalam aktivitas pada rantai nilai.

2.5.3 Konsep *Value Chain*

Menurut Porter (1998:39-41), menjelaskan bahwa *Value Chain* terbagi dalam dua jenis aktivitas dan di dalam aktivitas tersebut dibagi pada beberapa kategori yaitu sebagai berikut:

1. Aktivitas Primer (*Primary Activities*)
   a. Logistik ke dalam *(Inbound Logistic)*

      Kegiatan yang berhubungan dengan menerima, menyimpan, dan menyebarkan masukan ke produk, seperti *material handling*, pergudangan, *inventory control*, penjadwalan kendaraan, dan kembali ke pemasok.

   b. Operasi (*Operation*)

      Kegiatan yang berhubungan dengan mengubah *input* menjadi bentuk produk akhir (*output*), seperti mesin, kemasan, perakitan, pemeliharaan peralatan, pengujian, percetakan, dan fasilitas dalam kegiatan operasi.

   c. Logistik ke luar *(Outbound Logistic)*

      Aktivitas yang berhubungan dengan pengumpulan, penyimpanan, dan fisik mendistribusikan produk kepada pembeli, seperti selesai pergudangan

barang, *material handling*, kendaraan operasional pengiriman, pemrosesan pemesanan, dan penjadwalan.

d. Pemasaran dan Penjualan (*Marketing and Sales*)

Kegiatan yang berhubungan dengan menyediakan sarana yang pembeli dapat membeli produk dan mendorong mereka untuk melakukannya, seperti iklan, promosi, *salesforce*, pilihan *channels*, hubungan dengan *channels*, dan harga.

e. Pelayanan *(Service)*

Kegiatan yang berhubungan dengan menyediakan layanan untuk meningkatkan atau mempertahankan nilai produk, seperti instalasi, perbaikan, pelatihan, pasokan suku cadang, dan penyesuaian produk.

2. Aktivitas Sekunder (*Support Activities*)

a. Pengadaan *(Procurement)*

Pengadaan mengacu pada aktivitas-aktivitas yang dilakukan untuk pembelian *input* yang diperlukan dalam kegiatan produksi dalam rantai nilai perusahaan, bukan untuk *input* yang dibeli sendiri.

b. Pengembangan Teknologi (*Technology Development*)

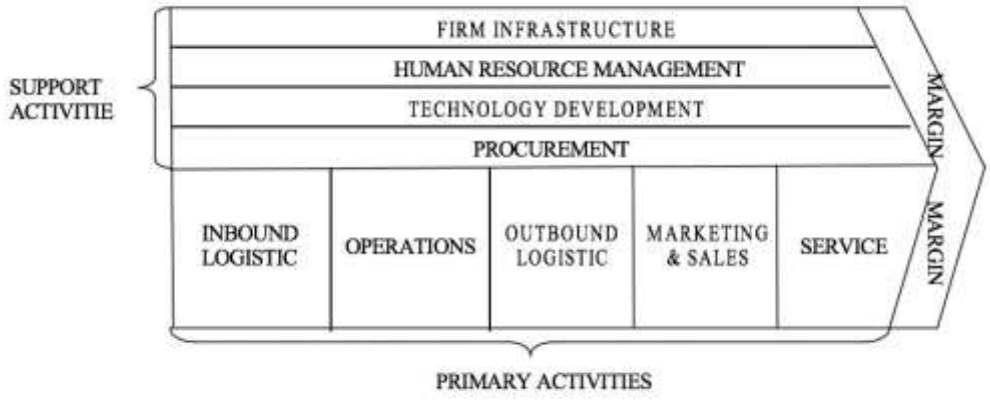
Perkembangan teknologi terdiri dari berbagai kegiatan yang dapat dikelompokkan menjadi upaya untuk meningkatkan produk dan proses yang digunakan perusahaan.

c. Manajemen Sumber Daya Manusia (*Human Resource Management*) Manajemen sumber daya manusia terdiri dari kegiatan yang terlibat dalam merekrut, menyewa, pelatihan, pengembangan, dan kompensasi dari semua jenis personil.

d. Infrastruktur Perusahaan (*Firm Infrastructure*)

Infrastruktur perusahaan terdiri dari sejumlah kegiatan termasuk manajemen umum, perencanaan, keuangan, akuntansi, hukum, urusan pemerintahan, dan manajemen mutu.

Menurut Porter (1998:37), menjelaskan bahwa aktivitas-aktivitas yang dilakukan dalam *Value Chain Analysis* sebagai berikut:



Gambar 2.12
The Generic Value Chain (Porter, 1998)

2.6 *Unified Modeling Language* (UML)

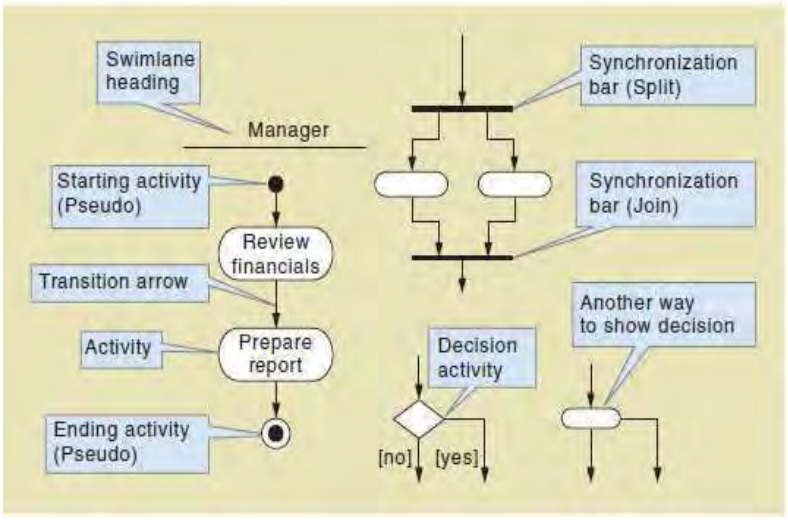
Menurut Satzinger, Jackson, dan Burd (2012), *unified modelling language* (UML) merupakan seperangkat model kontruksi dan notasi yang dibentuk dalam pengembangan sistem berorientasi pada objek. Diagram-diagram bagian dari UML yang dapat dijelaskan sebagai berikut:

2.6.1 Diagram *Activity*

Menurut Satzinger, Jackson, dan Burd (2012), *Activity Diagram* merupakan diagram yang menunjukkan alur kerja atau aktivitas *user* secara berurutan. *Activity Diagram* sendiri terdiri dari beberapa notasi dan fungsi masing–masing. Notasi dan fungsi di dalam *acitivity diagram* dijelaskan sebagai berikut:

1. *Swimlane*, merupakan suatu bentuk persegi yang merepresentasikan aktivitas-aktivitas yang diselesaikan setiap agen.
2. *Synchronization Bar*, merupakan notasi yang berfungsi memisahkan (*split*) atau menyatukan *(join)* urutan jalur aktivitas.
3. *Starting Activity (Pseudo)*, merupakan notasi yang menunjukkan awal dimulainya suatu aktivitas.
4. *Transition Arrow*, merupakan notasi yang berupa anak panah yang mendeskripsikan arah perpindahan suatu aktivitas.
5. *Activity*, merupakan notasi yang mendeskripsikan aktivitas-aktivitas.

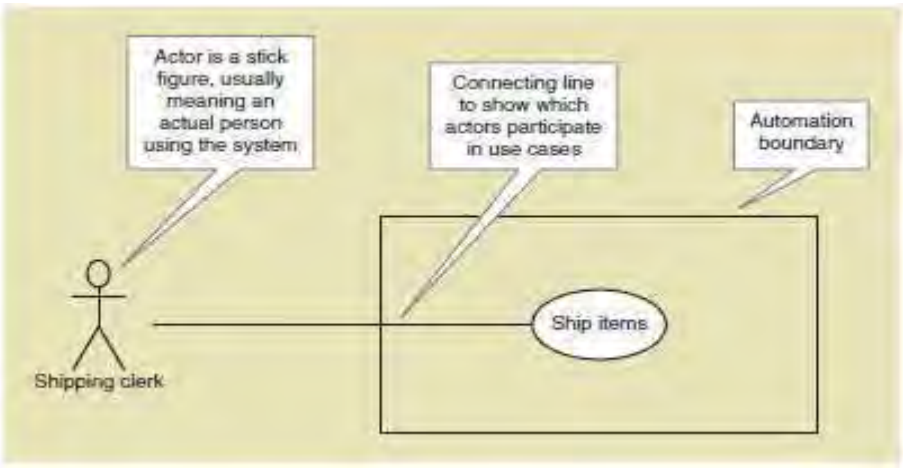
6. *Ending Activity (Pseudo)*, merupakan notasi yang menunjukkan diakhirinya suatu aktivitas.
7. *Decision Activity*, merupakan notasi yang mendeskripsikan kondisi dari suatu aktivitas.



Gambar 2.13
Diagram *Activity* (Satzinger, 2012)

2.6.2 Diagram *Use case*

Menurut Satzinger, Jackson, dan Burd (2012, p78), *use case* adalah aktivitas yang dilakukan oleh sistem berupa respon terhadap permintaan pengguna serta hubungan antara aktor–aktor pengguna tersebut di dalam sistem.



Gambar 2.14
Diagram *Use case* (Satzinger, 2012)

Use case description merupakan penjelasan terperinci mengenai proses dari suatu *use case* atau bisa disebut juga sebagai daftar kasus penggunaan diagram *use case* yang

memberikan gambaran dari semua penggunaan kasus untuk sistem. Informasi rinci tentang setiap kasus penggunaan digambarkan dengan menggunakan deskripsi kasus.

Use case description dibedakan menjadi tiga, yaitu:

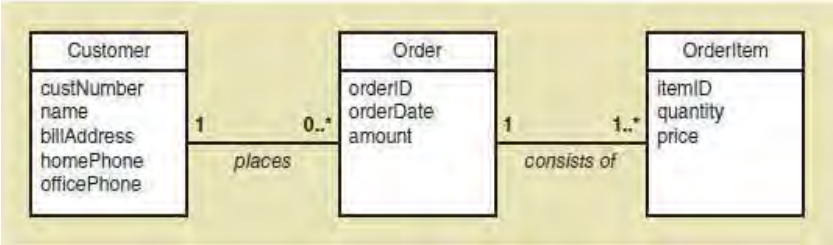
1. *Brief Description* dapat digunakan untuk *use case* yang sederhana, khususnya sistem yang dikembangkan untuk aplikasi kecil yang mudah dimengerti.
2. *Intermediate Description* memperluas penjelasan singkat untuk memuat arus aktivitas internal dari *use case*.
3. *Fully Developed Description* adalah metode paling formal untuk mendokumentasikan *use case* karena menjelaskan secara rinci setiap proses dalam *use case diagram*.

Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information. 2. Customer enters one or more addresses. 3. Customer enters credit/debit card information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses. 2.1 System creates addresses. 2.2 System prompts for credit/debit card. 3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

Gambar 2.15
Use Case Description (Satzinger, 2012)

2.6.3 *Domain Class Diagram*

Domain class diagram adalah sebuah diagram UML yang merepresentasikan kelas-kelas domain, atribut, pekerjaan pengguna serta hubungan antar kelas tersebut. Pada *class diagram,* bentuk kotak menggambarkan *classes* dan garis menunjukkan hubungan antar *class* tersebut. *Domain class diagram* digunakan untuk memahami hubungan antar *class* yang terdiri dari beberapa objek di dalam pengembangan dan perancangan sistem nantinya.

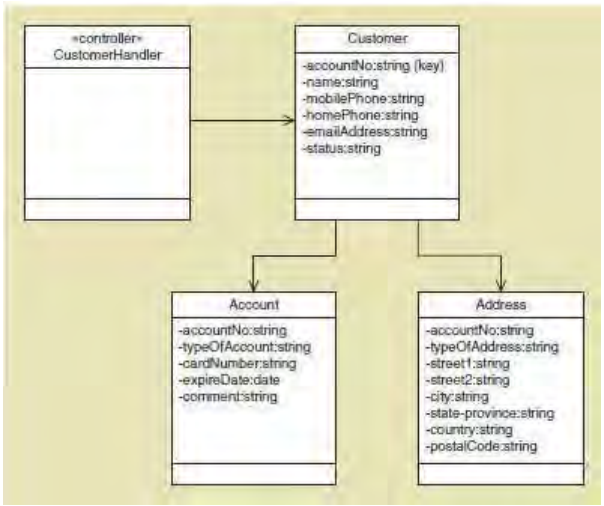


Gambar 2.16
Domain *Class* Diagram (Satzinger, 2012)

2.6.4 First-Cut Design Class Diagram

Pengembangan *design class diagram* dapat dilakukan pada setiap *layer*, dimana dalam *view* dan *data access layer* dilakukan penentuan beberapa *class* baru. Pada *domain layer*, *class* baru yang ditambahkan berfungsi sebagai *use case controller*. Penambahan *method* untuk setiap *class* dalam *updated class diagram* dapat dilakukan, dimana *method* tersebut terdiri dari 3 jenis, yaitu:

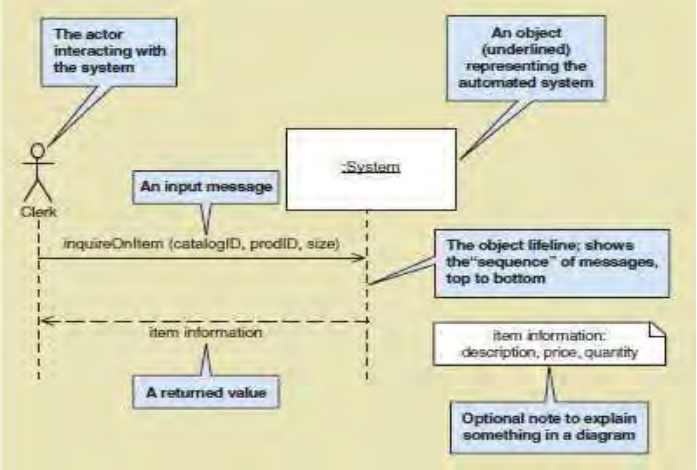
1. *Constructor methods*, merupakan *method* yang membentuk *instance* dari suatu obyek.
2. *Data get and set methods, m*erupakan *method* yang mengambil dan mengubah nilai atribut.
3. *Use case specific methods, m*erupakan *method* yang mewakili *use case* yang ada.



Gambar 2.17
First–Cut Design Class Diagram (Satzinger, 2012)

2.6.6 *System Sequence Diagram*

System sequence diagram merupakan diagram yang menunjukkan urutan pesan antara aktor eksternal dan internal sistem di dalam *use case* atau *scenario* yang sudah dirancang sebelumnya.



Gambar 2.18
System Sequence Diagram (Satzinger, 2012)

System sequence diagram (SSD) terdiri dari beberapa notasi. Berikut merupakan penjelasan dari masing-masing notasi SSD:

1. *Lifeline / Object Lifeline*

   Garis di bawah objek pada SSD menunjukkan berlalunya waktu untuk objek.

2. *Loof Frame*

   Notasi di dalam SSD yang menunjukan pengulangan pesan.

3. *True* / *False Condition*

   Bagian dari pesan diantara obyek yang dievaluasi sebelum ditransmisikan untuk menentukan apakah pesan dapat dikirim atau tidak.

4. *OptFrame*

   Notasi di dalam *sequence diagram* yang menunjukan pesan yang berupa optional atau pilihan.

5. *AltFrame*

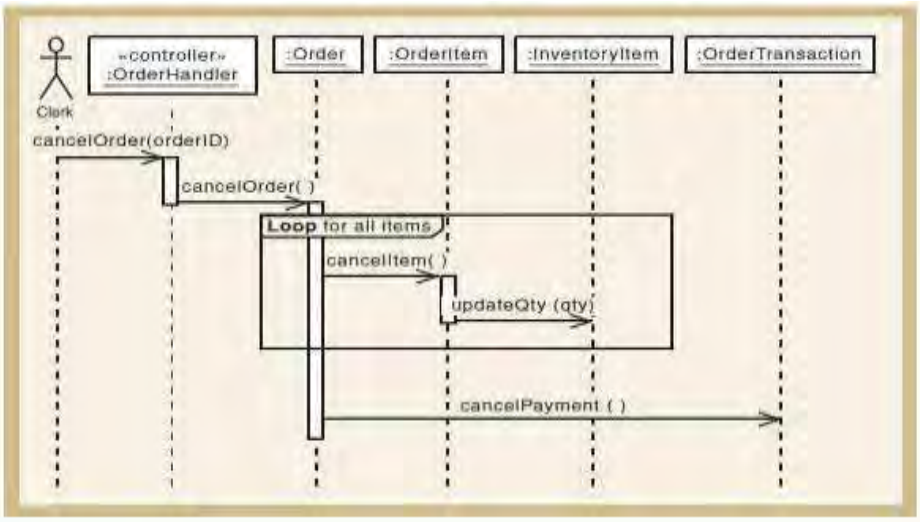
   Notasi di dalam *sequence diagram* yang menunjukan pesan if-else.

Sequence Diagram merupakan sebuah Diagram yang menunjukkan eksekusi operation disebuah objek yang melibatkan pemanggilan operations di objek lain. *Sequence Diagram* dikategorikan menjadi tiga bagian :

1. *First Cut Sequence Diagram*
2. *View Layer*
3. *Data Access Layer*

2.6.7 *First Cut Sequence Diagram*

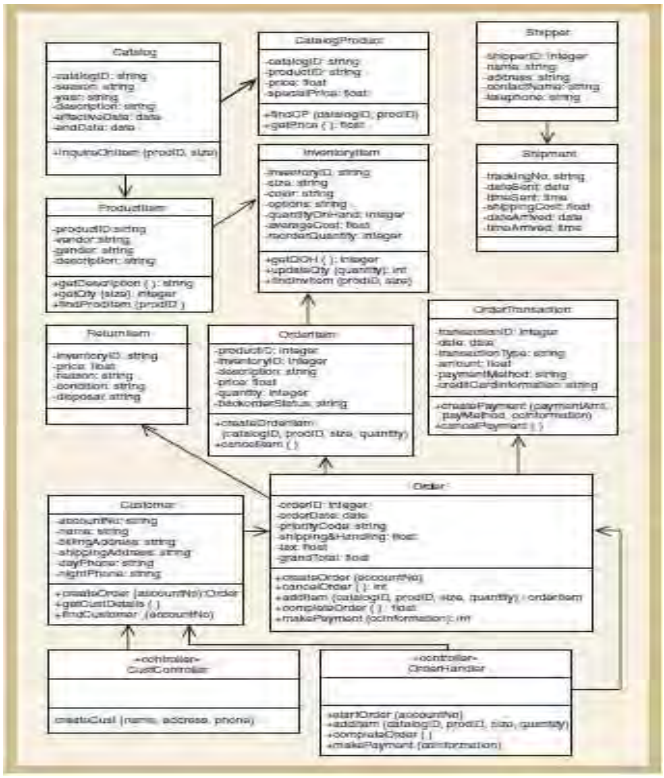
Menggunakan semua elemen yang terdapat pada SSD, perbedaanya hanya pada objek-objek internal dan pesan dalam sistem



Gambar 2.19
First Cut Sequence Diagram (Satzinger, 2012)

2.6.7 *Updated Class Diagram*

Updated Class Diagram merupakan pengembangan dari *first-cut class diagram*. Pada *updated class diagram*, informasi *method* dapat ditambahkan ke dalam *class* dan dilakukan pembaharuan arah panah navigasi yang didapat dari *sequence diagram* yang telah dibuat sebelumnya. Semua *handler* akan menjadi *class* baru.



Gambar 2.20
Updated Class Diagram (Satzinger, 2012)

2.7 *Electronic Ticketing* (*e-ticketing*)

Electronic ticketing atau *e-ticketing* merupakan salah satu bentuk *e-commerce* yang berkembang dalam bidang transportasi. Teknologi ini dikenalkan pertama kali oleh sebuah perusahaan penerbangan di Amerika bernama Value Jet pada Agustus 1993. Semenjak itulah *e-ticketing* perusahaan penerbangan lainnya satu per satu mulai menerapkan teknologi tersebut. Seiring berjalannya waktu, teknologi ini membuka potensi penggunaannya untuk mendukung proses bisnis perusahaan yang bergerak di bidang penyedia jasa layanan transportasi lainnya (seperti: bus, kereta, kapal, dan lain-lain).

2.7.1 Definisi *e-ticketing*

Definisi dari *Electronic Ticketing* (*e-ticketing*) adalah sebuah dokumen elektronik yang banyak digunakan sebagai tiket penumpang moda transportasi. Di sisi lain menyebutkan *e-ticketing* merupakan suatu cara untuk mendokumentasikan proses penjualan dari aktifitas perjalanan pelanggan tanpa harus mengeluarkan dokumen secara fisik (Grace,

2009). Sehingga dapat diartikan bahwa *e-ticketing* merupakan sebuah teknologi yang berguna untuk menggantikan pengolahan dan penggunaan tiket tradisional (*paper ticket*) (Iwuagwu, 2006).

Melihat dari sisi perkeretaapian, sistem *e-ticketing* berpeluang untuk meminimalkan biaya dan mengoptimalkan kenyamanan penumpang. *E-ticketing* mengurangi biaya proses tiket, menghilangkan formulir kertas dan meningkatkan fleksibilitas penumpang dan agen perjalanan. Tujuannya adalah untuk membuat pembelian atau pemesanan tiket lebih mudah (Karami, 2006).

2.7.2 Keuntungan dan Kekurangan *E-ticketing*

E-ticketing menawarkan berbagai keuntungan. Sistem *e-ticketing* menggantikan metode tiket tradisional dengan menghilangkan kebutuhan kertas tiket, pemrosesan dan registrasi data secara manual. Secara garis besarnya, sistem *e-ticketing* mengotomatisasi dasar-dasar dari pemrosesan tiket elektronik, seperti pemesanan tiket, pembayaran tiket, akuntansi pendapatan, pengelolaan informasi kedatangan dan keberangkatan pada penumpang, dan lain-lain. Proses-proses tersebut memerlukan waktu dan sumber daya manusia cukup banyak apabila dilakukan dalam model tiket tradisional.

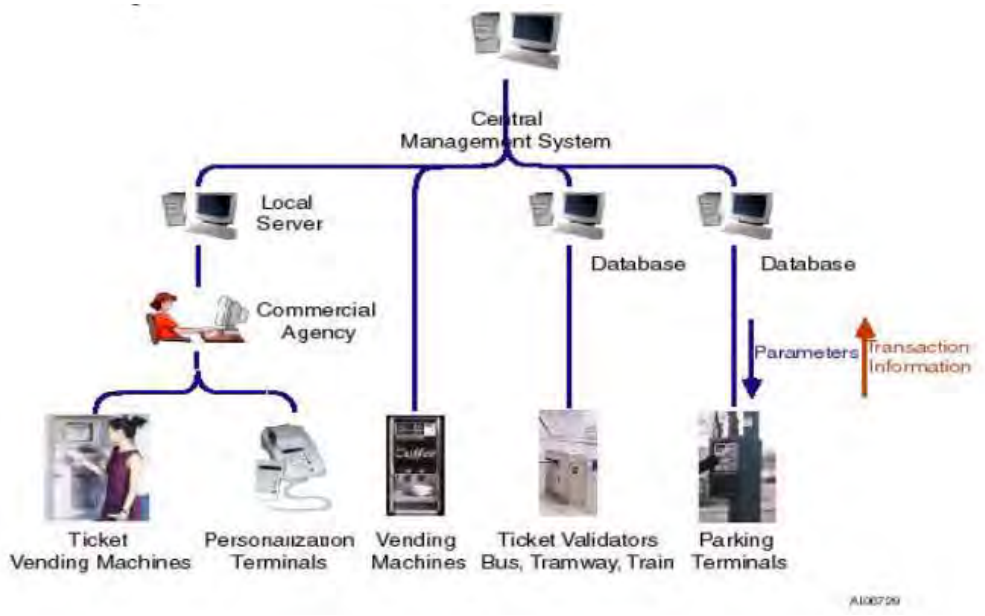
E-ticketing tidak hanya menghemat waktu administrasi dan biaya pada proses-proses yang telah terotomatisasi. *E-ticketing* juga dilengkapi dengan fitur *security* terkini yang menjadi isu penting dalam keamanan transaksi dalam sistem angkutan massal. Selain itu, keuntungan lain yang didapatkan dari penerapan teknologi ini diantaranya yaitu adanya percepatan perolehan pendapatan dan peningkatan efisiensi keseluruhan operasional jaringan transportasi.

Terlepas dari potensi keuntungan yang bisa diraih, masih terdapat risiko dengan diadopsinya sistem *e-ticketing*. Salah satu faktor yang menjadi risiko utama yaitu potensi terjadinya kegagalan sistem yang dapat menyebabkan gangguan dan kerugian finansial bagi operator transportasi.

Selain itu, dalam *e-ticketing*, keamanan komunikasi dan transaksi menjadi faktor paling penting. Pengamanan sistem *e-ticketing* menjadi tantangan yang permanen dan membutuhkan *monitoring*, penilaian, dan inovasi teknologi yang konstan.

2.7.3 Infrastruktur *e-ticketing* angkutan massal

Infrastruktur *e-ticketing* yang lazim digunakan dalam sistem angkutan massal harus bisa menangani transaksi penumpang dalam volume besar, selain itu harus memiliki resistansi perjalanan transportasi yang minimal (Heydt-Benjamin, 2006). Infrastruktur *e-ticketing* mengintegrasikan penjualan tiket, validasi dan data penting lainnya, serta mengolah aktivitas dengan cara yang aman. Inti dari infrastruktur *e-ticketing* yaitu manajemen komputer pusat, yang mengkoordinasikan keseluruhan sistem. Komponen infrastruktur lainnya, meliputi *database*, *server*, validator tiket, *vending machine*, terminal, gate, kartu, dan lain-lain. Infrastruktur *e-ticketing* harus dapat rentang hidup dan stabilitas perangkat keras lebih lama dibandingkan dengan metode tiket tradisional.



Gambar 2.21
Diagram Infrastruktur *e-ticketing* untuk sistem angkutan massal

Infrastruktur *e-ticketing* memiliki dalam pemrosesan yang aman dalam model jaringan komputer terdistribusi. Model jaringan komputer terdistribusi yang terdiri dari *client* dan *server* yang terhubung sedemikian rupa sehingga ada komunikasi dengan sistem lain.

Karakteristik utama dari lingkungan terdistribusi yaitu pemanfaatan komputasi *client*/*server* dan arsitektur *multi-tier*. Dengan demikian, Infrastruktur sistem *e-ticketing*

mendistribusikan pemrosesan dan meringankan beban *server* dari banyak proses. Dan data dapat diakses melalui jaringan kabel maupun nirkabel.

Model terdistribusi menawarkan pembiayaan yang rendah dan kapabilitas produk dengan volume yang tinggi, dan dianggap sebagai lingkungan yang cocok untuk sistem *e-ticketing*. Selain itu, model terdistribusi juga memungkinkan pengguna bekerja pada terminal grafis yang terhubung ke *server* yang melakukan semua pemrosesan dan menyimpan data pengguna dalam sebuah pendekatan dikenal sebagai metode *thin client*.

Dimensi penting lainnya pada model terdistribusi adalah teknologi *web*, dimana *server* memberikan akses secara universal terhadap *client* melalui internet. Aspek penting dalam model ini adalah adanya berbagai komputasi *platform*, solusi dan sistem operasi yang tidak menjadi kendala bagi komunikasi dan pertukaran informasi antar berbagai perangkat. Pada bagian model terdistribusi, tidak ada batasan jenis sistem *e-ticketing* yang bisa diimplementasikan.

Sistem *e-ticketing* telah menjadi perhatian secara substansial dalam industri penerbangan, tapi sedikit dieksplorasi dalam konteks dan pengaturan yang lain. Namun, telah terjadi sebuah perubahan, dimana sekarang moda angkutan umum seperti kereta api, trem, bus, kapal feri, juga mulai menerapkan teknologi ini.

2.7.4 Pemrosesan Informasi Tiket

Ada berbagai metode pengolahan informasi dalam *e-ticketing*. Pada sebagian besar sistem *e-ticketing*, pemrosesan informasi dimulai dari pemilihan tiket oleh pelanggan. Pada saat rincian tiket yang dipiliih sudah ditentukan, informasi pribadi dan kartu penumpang dikirim dengan aman ke sistem *e-ticketing* dari operator transportasi. Informasi ini kemudian diproses dan tiket dicetak untuk penumpang.

Semua data transaksi tiket disimpan di dalam server yang digunakan sebgaai referensi dan analisis statistik di masa yang akan datang (jika diperlukan). Sebuah komponen penting dalam pemrosesan informasi tiket, yaitu modul sistem manajemen *e-ticketing*. Modul ini biasanya termasuk modul *vendor*, modul pemrosesan tiket, modul *web*, modul pemesanan tiket, modul *delivery* dan verifikasi tiket. Di dalam sistem *e-ticketing* dimana terdapat *service-service* dari operator yang berbeda-beda yang terintegrasi,

terdapat sebuah sistem kliring terpusat (*clearing house*), yang menjamin pembayaran untuk masing-masing operator dan menyimpan data transaksi tersebut dengan aman.

Dalam beberapa sistem *e-ticketing* berbasis *web*, pemrosesan informasi dimulai ketika penumpang telah melakukan registrasi dalam layanan melalui sebuah *web* portal tiket. Pembayaran tiket dapat dilakukan melalui kartu debit, kartu kredit atau dengan menggunakan dompet virtual dimana uangnya dapat ditransfer secara elektronik. Jika tiket sudah dibayar, tiket dikirimkan ke penumpang.

2.8 Studi Hasil Penelitian Sebelumnya

Kontribusi penelitian ini adalah perancangan arsitektur sistem menggunakan kerangka *service oriented* untuk studi kasus sistem tiket elektronik kereta api. Untuk bisa menjelaskan kontribusi tersebut dan posisi dari rencana penelitian, berikut beberapa penelitian sebelumnya mengenai perancangan arsitektur *enterprise* berbasis *service:*

1. Gregor Engels dan Martin Assman pada tahun 2008 dengan judul penelitian *Service Oriented Enterprise Architectures : Evolution of Concepts and Methods,* memaparkan perubahan perkembangan dari arsitektur *enterprise* berbasis *service*.
2. Mohammed Kazem HAKI dan Maia Wetland Forte pada tahun 2010 dengan judul penellitian *Service Oriented Enterprise Architecture Framework* mengemukakan kerangka arsitektur *enterprise* yang lahir dari kombinasi konsep SOA dan EA yang dinamakan SOEA.
3. Grace Ng-Kruelle dan Paul Anthony Swatman pada tahun 2006 dengan judul *e-ticekting Strategy and Implementation in an Open Access System: The case of Deutsche Bahn* menjelaskan tentang kasus praktis implementasi *e-ticketing* pada sistem "*open access"* pada perusahaan operator nasional kereta api Jerman yang dimiliki oleh Pemerintah Jerman yaitu Deutsche Bahn.
4. Iwuagwu O F pada tahun 2009 dengan penelitian berjudul *Electronic ticketing in public transportation systems the need for standardization* mengemukakan standar ilmiah, teknologi dan aplikasi dalam tiket elektronik menggunakan kerangka dasar dan teoritis *European Interoperability Framework*.

5. Lili Fan, Xinhua Liu, Jianwen Wang tahun 2012 dengan penelitian berjudul *An Evaluation of the Operation of the Railway E-ticketing System* bahwa kinerja alat dan kestabilan sistem berpengaruh sangat besar terhadap peforma sistem *e-ticketing*.

Dari penelitian-penelitian di atas, dapat dilihat belum adanya laporan penelitian untuk perancangan arsitektur sistem tiket elektronik (*e-ticketing)* kereta api yang menggunakan kerangka SOEA.

BAB III

OBYEK DAN METODOLOGI PENELITIAN

3.1 Profil PT Railink

PT Railink merupakan anak perusahaan dari PT Kereta Api Indonesia (Persero) dan PT Angkasa Pura II (Persero) yang beroperasi dalam layanan aksesibilitas bandara. Jika semula aksesibilitas bandara didominasi oleh layanan transportasi berbasis jalan raya yang kian hari kian padat lalu lintasnya dengan segala problematika yang ada, maka PT Railink hadir dengan menyuguhkan warna baru layanan transportasi publik berbasis kereta api bernama Kereta Api Bandara (KA Bandara).

KA Bandara dirancang oleh PT Railink untuk menjadi bagian dari solusi penting atas persoalan yang mendera masyarakat Indonesia selama ini dalam aksesibilitas bandara yang diwarnai dengan kemacetan lalu lintas jalan darat, dan ketidakpastian durasi waktu perjalanan.

Pada tanggal 25 Juli 2013 PT Railink telah berhasil mengoperasikan KA Bandara Kualanamu sebagai KA Bandara pertama di Indonesia. Kelahiran KA Bandara Kualanamu telah menyejajarkan Indonesia dengan negara lain yang telah memiliki KA Bandara sebagai salah satu lambang supremasi transportasi nasional seperti Jepang, Inggris dan lain-lain. Kehadirannya telah mendongkrak *rating* Bandara Kualanamu di mata Internasional berupa sertifikasi bintang empat berdasarkan penilaian lembaga *rating* Skytrack pada KA Bandara yang merupakan manifestasi dari wajah baru transformasi perkeretaapian di Indonesia dan dirancang untuk menghadirkan layanan premium.

Hal ini senada dengan peran bandara itu sendiri sebagai pintu gerbang bagi sebuah negara dimana segala kebaikan dan keunggulan harus disuguhkan untuk membangun kesan positif bagi negara. Dalam mewujudkan layanan premium pada KA Bandara, PT Railink fokus dalam menghadirkan *on time performance* yang tinggi, keramahan khas Indonesia, kecepatan, kenyamanan, keamanan, serta berbagai kemudahan dan fasilitas di stasiun KA Bandara.

Setelah tiga tahun KA Bandara Kualanamu bertumbuh seiring dengan apresiasi masyarakat Indonesia dan menjadi kebanggaan tersendiri bagi masyarakat Sumatera Utara, PT Railink siap menghadirkan KA Bandara baru di Ibu Kota Negara: KA Bandara Soekarno Hatta (BSH).

PT Railink tertantang untuk melanjutkan dedikasinya dalam melayani para pengakses Bandara Internasional Soekarno Hatta, salah satu Bandara tersibuk di Asia dengan pergerakan penumpang airline sekitar 60 juta orang per tahun dari 413 ribu penerbangan dalam dan luar negeri per tahun.

KA Bandara Soekarno-Hatta akan melayani rute dari Manggarai sampai dengan Bandara Soekarno Hatta (pp) sepanjang 36,3 km dengan melewati Stasiun Sudirman Baru sebagai *City Railway Station* (CRS), Stasiun Duri dan Stasiun Batuceper, sebanyak 124 jadwal keberangkatan setiap hari, dengan *headway* tiap 15 menit sekali, jam operasi yang terus diperbaharui dengan perubahan jadwal penerbangan, serta kapasitas angkut 33.728 penumpang per hari.

KA Bandara Seoekarno-Hatta sangat ditunggu kehadirannya oleh masyarakat Indonesia, diharapkan bisa menjadi bagian dari jargon transportasi selama ini bahwa negara yang baik adalah negara yang mampu menghadirkan pilihan transportasi yang bervariasi dan berkualitas bagi masyarakatnya.

3.2 *Milestone* Sejarah PT Railink

Tabel 3.1 menjelaskan *milestone* sejarah PT Railink dari tahun 2006 hingga tahun 2017:

Tabel 3.1 *Milestone* Sejarah PT Railink

Tahun	**Deskripsi**
2006	Pembentukan awal perusahaan PT Railink
2011	1. Pembangunan *City Railways Station* (CRS) Medan dan *Airport Railways Station* (ARS) di Bandara Kualanamu. 2. Pemesanan Kereta Rel Disel (KRD) / *Diesel Multiple Unit* (DMU) ke Woojin.
2013	1. Pengoperasian Pertama Kereta Api Bandara Kualanamu

Tahun	Deskripsi
	2. Pengoperasian Armada Produksi Woojin.
2014	1. Penambahan Frekuensi Kereta Api Bandara Kualanamu
	2. Penandatanganan Kontrak Pembelian Kereta Rel Listrik Kereta Api Bandara Soekarno-Hatta.
2015	1. Sertifikasi ISO 9001. 2. Penandatanganan Kontrak Kredit Pembelian Kereta Rel Listrik Bandara Soekarno-Hatta. 3. Pembebasan Lahan. 4. Penandatanganan Perjanjian Induk dengan Induk Perusahaan untuk Kereta Api Bandara Soekarno-Hatta.
2016	1. Perubahan Logo Perusahaan 2. Adendum Kontrak 3. Pembukaan *Letter of Credit* (LC) Pembelian Kereta Rel Listrik Bandara Soekarno-Hatta 4. Pengembangan *Airport Railways Ticketing System* (ARTS)
2017	1. Pengoperasian *Airport Railways Ticketing System* (ARTS) 2. Pengoperasian Kereta Api Bandara Soekarno-Hatta

3.3 Visi, Misi, Tujuan dan Sasaran

PT Railink sebagai sebuah perusahaan memiliki visi, misi, tujuan sebagai berikut:

1. Visi : Menjadi Pilihan Utama Akses Bandara yang terintegrasi dan bertaraf internasional.
2. Misi: Menyelenggarakan jasa transportasi Kereta Api Bandara serta kegiatan usaha lainnya yang terkait dengan sehat, tumbuh dan berkembang dengan tujuan untuk: mewujudkan kepuasan pelanggan, meningkatkan kesejahteraan karyawan, memberikan nilai tambah kepada pemegang saham, serta memberikan manfaat bagi

komunitas dan pelestarian lingkungan dengan menjalankan tata kelola perusahaan yang baik dan memegang teguh etika bisnis.

3. Tujuan: Melaksanakan dan mendukung kebijaksanaan serta program pemerintah di bidang ekonomi dan pembangunan nasional khususnya di bidang transportasi dengan menyediakan barang-jasa yang bermutu tinggi dan berdaya saing kuat untuk dapat melakukan ekspansi baik di pasar nasional, regional maupun internasional di bidang perkeretaapian yang meliputi usaha pengangkutan orang dengan kereta api, kegiatan perawatan dan pengusahaan prasarana, pengusahaan bisnis properti secara profesional serta pengusahaan bisnis penunjang prasarana dan sarana kereta api secara efektif untuk kemanfaatan umum.

Untuk mencapai tujuan sesuai dengan visi dan misi di atas, PT Railink memiliki kebijakan dan sasaran mutu sebagai berikut:

1. Kebijakan Mutu: Menyelenggarakan layanan akses bandara yang terintegrasi dan bertaraf internasional dengan memenuhi persyaratan mutu pelanggan dan menerapkan perbaikan yang berkesinambungan.
2. Sasaran Mutu:
    a. Tercapainya indeks kepuasan layanan KA bandara.
    b. Tercapainya kehandalan KA sesuai norma kendali.
    c. Tercapainya ketersediaan sarana sesuai Grafik Perjalanan Kereta Api (GAPEKA).
    d. Tercapainya ketepatan waktu KA berangkat dan datang sesuai GAPEKA.

3.4 Fasilitas & Sarana

KA Bandara akan melintas 42 kali perjalanan tiap harinya, dan akan terus dikembangkan di kemudian hari. Meskipun durasi perjalanan cukup singkat, namun KA Bandara tetap memperhatikan sisi kenyamanan bagi para penumpangnya. Sejumlah fasilitas di dalam rangkaian KA Bandara membuat para penumpang merasa nyaman dan terhibur. Di tiap rangkaian kereta terdiri dari 172 kursi, dengan kapasitas seperti itu, KA Bandara siap mengantarkan penumpang hingga 7000 penumpang tiap harinya. Selain itu

para penumpang juga diberikan kenyamanan dengan adanya toilet minimalis yang selalu terjaga kebersihannya. Kenyamanan dari sisi mobilitas juga tercermin dari adanya jaringan intenet nirkabel dan hiburan visual dari *Liquid Crystal Display Television* (LCD TV) yang terpasang di tiap kereta.

Selain menghemat waktu tempuh, layanan yang dimiliki KA Bandara merupakan magnet bagi penumpang angkutan udara yang memilih menggunakan akses transportasi KA Bandara. Stasiun KA Bandara dirancang senyaman mungkin dengan kesejukan merata dan dilengkapi dengan berbagai fasilitas umum seperti toilet, mushala hingga ruang menyusui. Kalaupun ada yang membutuhkan akses data internet, tak perlu risau karena tersedia jaringan internet nirkabel.

Tak hanya bagi penumpang KA Bandara, masyarakat umum juga dapat menikmati fasilitas Stasiun KA Bandara seperti sajian kuliner di restoran/*café* berkualitas ataupun berbelanja *souvenir* di area komersil yang telah disediakan. Untuk mempermudah transaksi keuangan, disediakan galeri ATM sejumlah bank serta tempat penukaran mata uang (*money changer*).

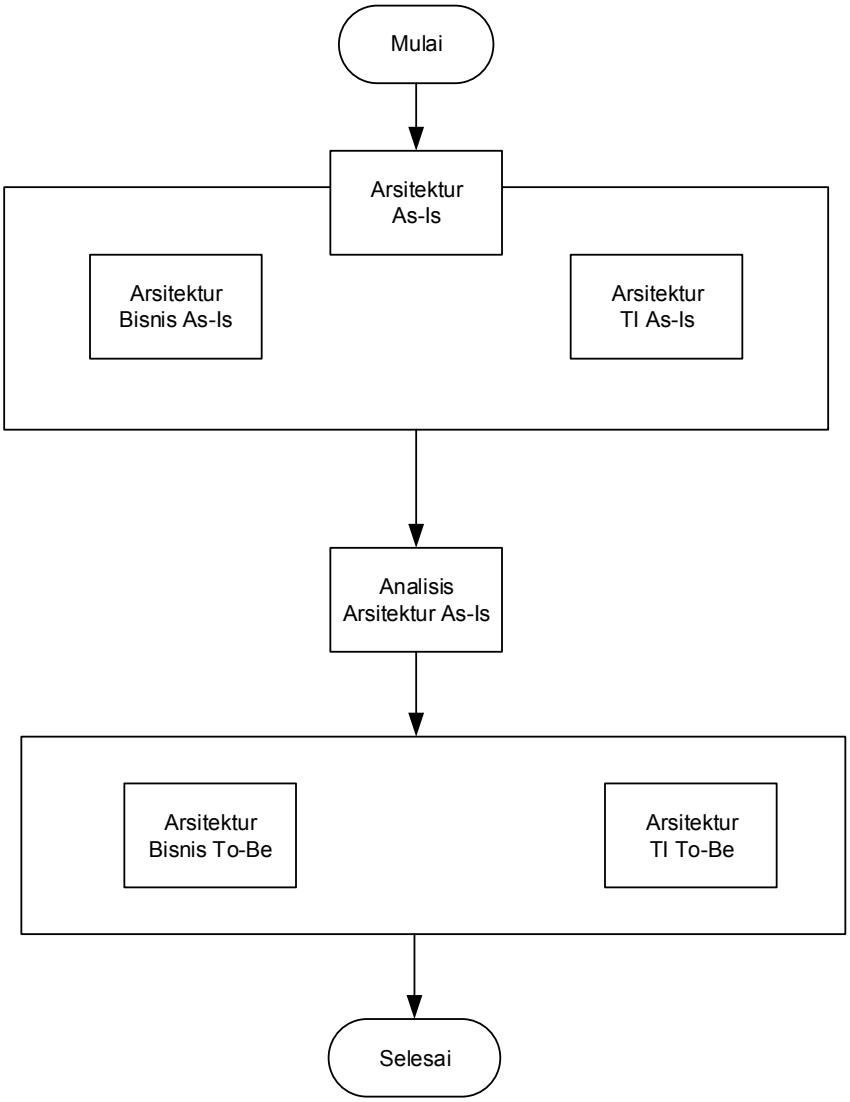
3.5 Sistem Tiket di PT Railink

Penggunaan teknologi informasi tak bisa dipisahkan dalam pengelolaan jasa transportasi untuk menghadirkan kemudahan bagi pelanggan. Untuk itu PT Railink menciptakan terobosan baru dengan membangun sistem ticketing bernama *Airport Railway Ticketing System* (ARTS). ARTS merupakan sistem *ticketing* yang dibangun dengan mengadopsi teknologi terkini dan mampu dikembangkan di kemudian hari dengan memiliki 3 (tiga) nilai inovatif: *cardless* (tanpa penerbitan kartu), *cashless* (tanpa menggunakan uang tunai) dan *manless* (tanpa memerlukan petugas loket).

Dalam pengembangan sisi bisnis dari ARTS, PT Railink bersinergi dengan para mitra yang teruji dan kapabilitasnya di bidang masing-masing (perbankan, *e-commerce*, *payment gateway*, *front-end* serta penyedia infrastruktur jaringan). Dengan ARTS, pelanggan Kereta Api Bandara dapat memperoleh layanan tiket dengan sangat mudah melalui berbagai cara seperti *website*, *mobile application*, *vending machine* dan lain-lain melalui berbagai transaksi non tunai menggunakan kartu.

3.6 Metodologi Penellitian

Berdasarkan permasalahan yang ada pada sistem *e-ticketing* di PT Railink, maka metodologi penelitian yang digunakan dalam perancangan arsitektur sistem tiket elektronik sesuai kerangka SOEA oleh Kazen (2010), seperti gambar berikut,



Gambar 3.1
Tahapan Metodologi Penelitian

3.6.1 Pemodelan Arsitektur *As-is*

Pemodelan Arsitektur *as-is* dari penjualan tiket di PT Railink dibuat berdasarkan penjualan tiket kereta api bandara baik internal dan eksternal. Metode yang digunakan adalah:

1. Pemodelan Arsitektur Bisnis *As-is*

Pemodelan proses bisnis *as-is* dilakukan agar dapat menjelaskan proses bisnis *as-is* secara lebih mudah untuk dipahami. Dalam pemodel proses bisnis digunakan standar *Business Processing Model* (BPM) yang merepresentasikan fungsi-fungsi yang berkaitan dengan kegiatan-kegiatan bisnis meliputi input, proses, output dan sumber daya. Tahapan ini dilakukan untuk identifikasi proses bisnis yang memerlukan perbaikan.

Pemodelan arsitektur bisnis *as-is* dilakukan dengan:

a. Mengidentifikasi struktur bisnis keseluruhan
b. Menguraikan komponen-komponen strategi bisnis *e-ticketing*
c. Mengidentifikasi struktur detail bisnis *e-ticketing*
d. Menguraikan *service-service* bisnis *as-is*

Sehingga dari pemodelan arstitektur bisnis *as-is*, terdapat keluaran sebagai berikut:

a. Model bisnis konseptual
b. Komponen-komponen strategi bisnis dan TI
c. Model fungsi bisnis, model proses, dan model data
d. *Service-service* bisnis *as-is*

2. Pemodelan Arsitektur TI *As-is*

Pemodelan arsitektur TI *as-is* merupakan proses pengembangan spesifikasi TI, model dan *guideline*, menggunakan berbagai notasi *Unified Modelling Language* (UML) dalam kerangka arsitektur TI yang koheren, sesuai dengan kaidah solusi TI baik formal maupun informal, enterprise, dan proses-proses arsitektur infrastrukturnya.

Pemodelan arsitektur TI *as-is* dilakukan dengan:

a. Mengidentifikasi sumberdaya teknis TI
b. Mengidentfikasi sumberdaya-sumberdaya non-teknis TI

Sehingga dari pemodelan arsitektur TI *as-is*, terdapat keluaran sebagai berikut:

a. Sistem informasi, aplikasi-aplikasi, integrasi, *security*, infrastruktur
b. Sumber daya manusia TI dan struktur manajemen

3.6.2 Analisis Arsitektur *As-is*

Analisis arsitektur kondisi *as-is* dari sistem penjualan tiket berdasarkan kondisi internal dan eksternal bisnis serta kondisi internal dan eksternal TI di PT Railink. Metode yang dilakukan adalah:

1. Analisis Arsitektur Bisnis *As-is*

   Analisis fokus pada pengumpulan informasi pada posisi saat ini dari bidang-bidang kegiatan penjualan tiket dan membuat gambaran yang jelas tentang *gap capability* yang bisa menjadi hambatan dalam mencapai potensi pendapatan secara keseluruhan.

   Analisis arsitektur bisnis tiket memetakan gambaran awal tujuan, sasaran, dan capaian bisnis tiket PT Railink secara arsitektural. Untuk mengetahui arsitektur bisnis *as-is* tiket PT Railink, maka perlu mengetahui perkembangan penggunaan dari *e-ticketing as-is* di unit komersial di PT Railink. Arsitektur bisnis *as-is* diketahui berdasarkan dokumen *blueprint* yang dimiliki oleh PT Railink, sedangkan data target pendapatan dan okupansi penjualan tiket didapatkan berdasarkan hasil pengumpulan data dari narasumber di unit komersial PT Railink serta berdasarkan observasi langsung ke lapangan. Analisis arsitektur bisnis juga meliputi faktor eksternal sistem tiket dalam lingkup luas berupa kondisi teknologi dan bisnis tiket yang dimiliki.

2. Analisis Arsitektur TI *As-is*

   Analisis arsitektur TI yang mencakup arsitektur TI dari sistem penjualan tiket saat ini di PT Railink, melakukan *review* terhadap infrastruktur teknologi informasi yang digunakan dan melakukan pengamatan peran sumber daya dan sistem informasi di dalam proses bisnis penjualan tiket.

Secara eksternal, analisis dilakukan untuk mengetahui tren TI yang digunakan oleh operator transportasi yang sejenis.

3.6.3 Perancangan Arsitektur *To-be*

Perancangan Arsitektur *to-be* dari sistem tiket elektronik di PT Railink dibuat dengan menggunakan metode sebagai berikut:

1. Perancangan Arsitektur Bisnis *To-be*

   Perancangan arsitektur bisnis *to-be* meliputi tiga *service* bisnis berikut:

   a. *Service* bisnis *core*

      Service bisnis core terkait dengan proses bisnis utama sistem tiket PT Railink dimulai dari transaksi reservasi (*booking)*, pembayaran, *gate-in* dan *gate-out* serta pembatalan tiket.

   b. *Service* bisnis *non-core*

      Service bisnis *non-core* terkait dengan proses bisnis pendukung. *Service* bisnis non-core meliputi *service* tiket manual dan pengiriman *e-mail*.

   c. *Service* eksternal

      Service eksternal yang diharapkan terdiri dari *service dashboard* laporan eksekutif dan rekonsiliasi pendapatan tiket *payment gateway*.

2. Perancangan Arsitektur TI *To-be*

   Perancangan arsitektur TI *to-be* akan meliputi *service-service* sebagai berikut:

   a. *Service* Aplikasi

      Service aplikasi dibangun berdasarkan *service* bisnisnya. Suatu *service* bisnis dapat memiliki satu atau lebih *service* aplikasi disertai dengan penamaan *service* dan alamat URI. *Service* dapat diakses sesuai kesepakatan standar protokol komunikasi yang menjembatani aplikasi-aplikasi *front-end* dengan *service*nya.

   b. *Service* Infrastruktur

      Service inffrastruktur dirancang untuk mendukung *service* aplikasi baik digunakan secara fisik maupun menggunakan virtualisasi melalui sekumpulan sumber daya

infrastruktur seperti server *web*, server aplikasi, server *database*, server lain, dan media penyimpanan (*storage*).

BAB IV

ANALISIS DAN PEMBAHASAN

4.1 Analisis Arsitektur *As-Is*

Kegiatan analisis *as-is* dimulai dengan fokus pada titik kritis pada lingkup pemodelan analisisnya. Model analisis *as-is* yang dibuat dengan mendetailkan apa yang menjadi gambarn kegiatan suatu perusahaan. Informasi yang harus dimasukkan kedalam modelanalisis as-is meliputi proses dari organisasi atau perusahaan, orang, tujuan, kelemahan, data, sumber daya, pengetahuan, pelanggan, dan sebagainya.

Tujuan analisis *as-is* merupakan proses dari sebuah upaya untuk meningkatkan transparansi tentang proses organisasi, dengan melihat unit bisnis organisasi yang digambarkan dalam struktur organisasi sebuah perusahaan. Selama analisis bisnis *as-is* berlangsung, hal yang lebih penting ialah dengan menyoroti kelemahan dan mengidentifikasi kemungkinan untuk perbaikan. Jika analis bisnis *as-is* yang digunakan untuk memilih solusi perangkat lunak baru, hal ini akan berguna untuk melihat bagaimana alternatif solusi perangkat lunak untuk mendukung proses tertentu. Analis bisnis *as-is* juga dapat memberikan masukan bagi *software developer* yang mengembangkan perangkat lunak baru berdasarkan spesifikasi dan persyaratan proses bisnis yang berlaku.

Analisis kondisi arsitektur saat ini (*as-is*) dilakukan berdasarkan kondisi bisnis serta kondisi TI perusahaan. Analisis tersebut ditujukan terhadap informasi posisi saat ini dan dalam rangka membuat gambaran proses bisnis yang sedang berlangsung disertai *gap capability* yang menjadi hambatan di dalam perancangan sistem *e-ticketing* kereta api sebagai upaya perusahaan dalam hal ini PT Railink mendorong pendapatan dari bisnis transportasi angkutan penumpang kereta api.

4.1.1 Analisis Arsitektur Bisnis As-Is

Dalam analisis arsitektur bisnis *as-is*, didapat dari hasil analisis struktur bisnis perusahaan dan relasinya dengan *stakeholders*. Analisis bisnis *as-is* memetakan gambaran awal tujuan, sasaran, dan capaian bisnis.

4.1.1.1 Identifikasi Bisnis Angkutan Penumpang Perusahaan

Sesuai dengan visi perusahan yaitu menjadikan pilihan utama askes bandara dengan layanan berkualitas internasional sebagaimana telah disebutkan di dalam Bab III, PT Railink memiliki konteks bisnis seperti yang digambarkan dalam Gambar 4.1 sebagai berikut,

Gambar 4.1
Konteks Bisnis Perusahaan

Bisnis utama PT Railink terdiri dari usaha bisnis angkutan penumpang, komersialisasi aset, dan layanan pendukung bisnis lainnya. Faktor kunci keberhasilan untuk mencapai keunggulan operasional yaitu dengan upaya-upaya: penerapan *safety* dengan teknologi yang tepat, peningkatan kualitas SDM, dan peningkatan akses pengguna. Siklus dalam mencapai keunggulan operasional tersebut tidak terlepas dari kegiatan-kegiatan: pengetahuan tentang pasar, adopsi teknologi yang tepat, dan pemanfaatan lahan secara efektif.

Sistem *e-ticketing* kereta api merupakan pilihan utama yang menunjang pendapatan perusahaan dalam bisnis angkutan penumpang.

4.1.1.2 Komponen Strategi Bisnis

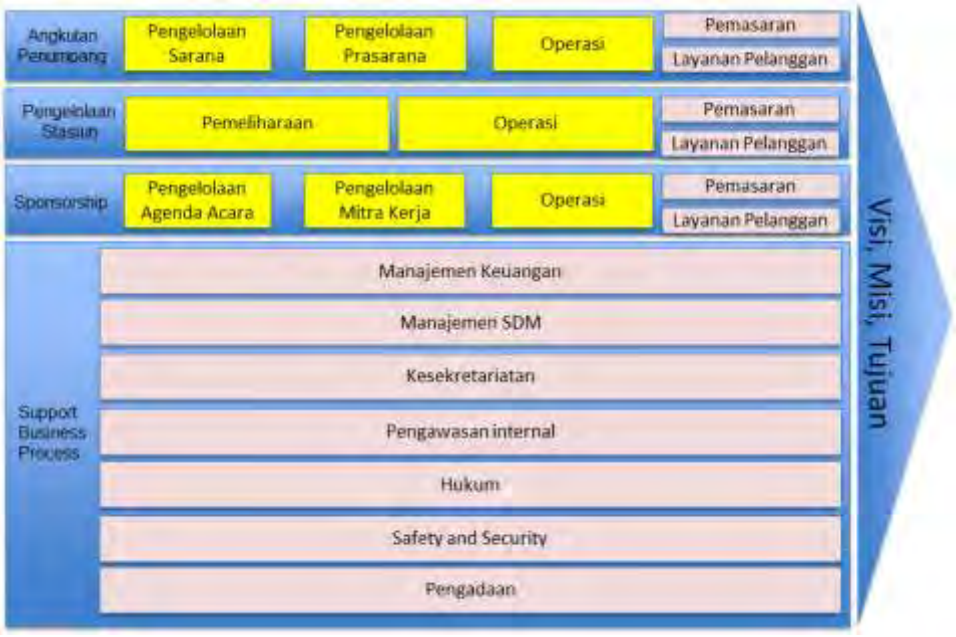
Strategi bisnis PT Railink dapat diuraikan sebagai berikut:

1. Meningkatkan porsi pendapatan secara agresif dari angkutan penumpang dan non penumpang.

2. Mengembangkan angkutan penumpang Kereta Api Bandara di wilayah PT Angkasa Pura I dan PT Angkasa Pura II.
3. Meningkatkan pelayanan angkutan *customer oriented* yang fokus pada kualitas, keamanan, kenyamanan dan nilai tambah bagi pengguna jasa angkutan KA.
4. Menumbuhkembangkan layanan pendukung KA Bandara seperti *City-Checkin*, *baggage handling*, *feeder transport*, dan lain-lain.
5. Meningkatkan *safety level* operasional Kereta Api mencapai *zero accident* dengan teknologi tepat sasaran dan handal.
6. Peningkatan peran sumber daya manusia dan teknologi informasi dalam kerangka efektifitas sistem informasi.

4.1.1.3 *Value Chain* Bisnis Perusahaan

Value Chain perusahaan dapat digambarkan pada Gambar 4.2 di bawah ini:



Gambar 4.2
Value Chain Bisnis Perusahaan

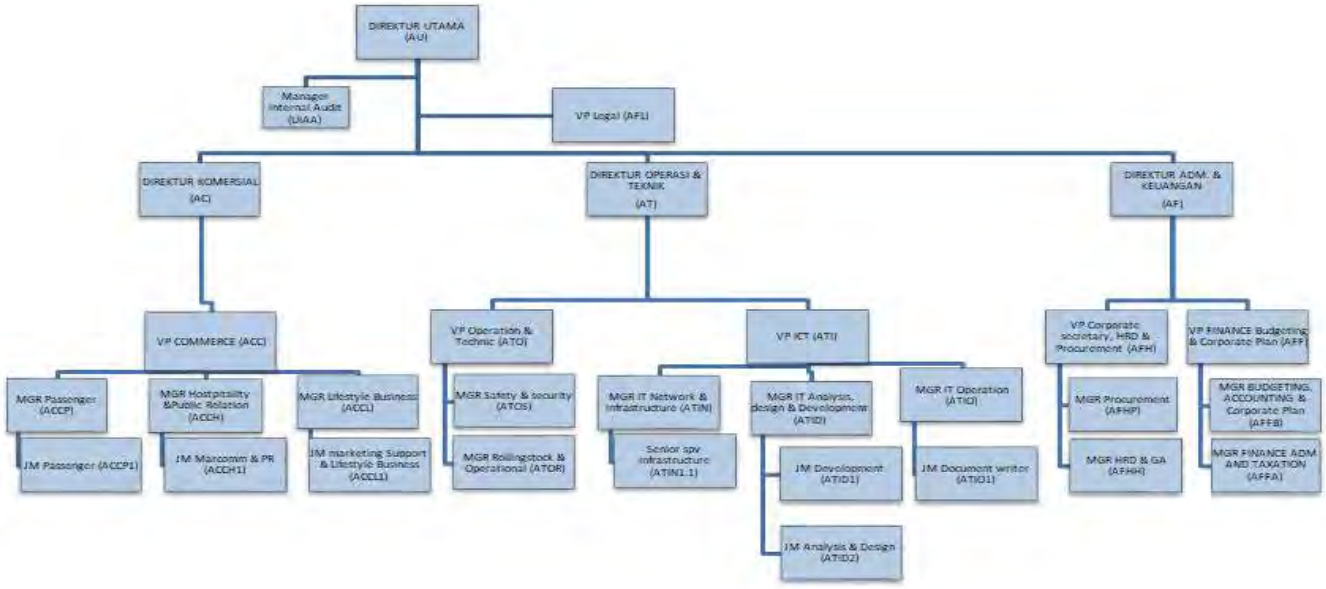
Deskripsi dari setiap proses bisnis berdasarkan *Value Chain* bisnis perusahaan dapat dijelaskan pada Tabel 4.1,

Tabel 4.1 Deskripsi dari Proses Bisnis dari *Value Chain* Bisnis Perusahaan

Prosses Bisnis	**Deskripsi**
Pengelolaan Sarana	Sarana kereta api meliputi kereta dan peralatan khusus. Sedangkan pengelolaan sarana tersebut meliputi perencanaan, penyediaan dan pemeliharaan sarana.
Pengelolaan Prasarana	Prasarana kereta api meliputi jalur stasiun dan fasilitas operasi kereta api. Sedangkan pengelolaan prasarana meliputi perencanaan, pembangunan dan pemeliharaan prasarana.
Operasi	Operasi kereta api meliputi operasi sarana maupun prasarana. Proses dalam operasi kereta tersebut meliputi perencanaan operasi, realisasi, kendali dan evaluasi operasi serta *safety*.
Pemasaran	Pemasaran angkutan penumpang meliputi riset pasar, pengelolaan tarif, pemasaran, penjualan tiket dan retensi/loyalty.
Layanan Pelanggan	Layanan pelanggan dilakukan setelah pelanggan membeli tiket atau menandatangani kontrak. Layanan pelanggan angkutan penumpang dilakukan di stasiun (*Customer Service On Station*), kereta (*Customer Service On Train*) dan melalui *Contact Center*. Sedang proses layanan pelanggan angkutan barang meliputi pemeliharaan kerjasama dan penanganan keluhan pengguna.

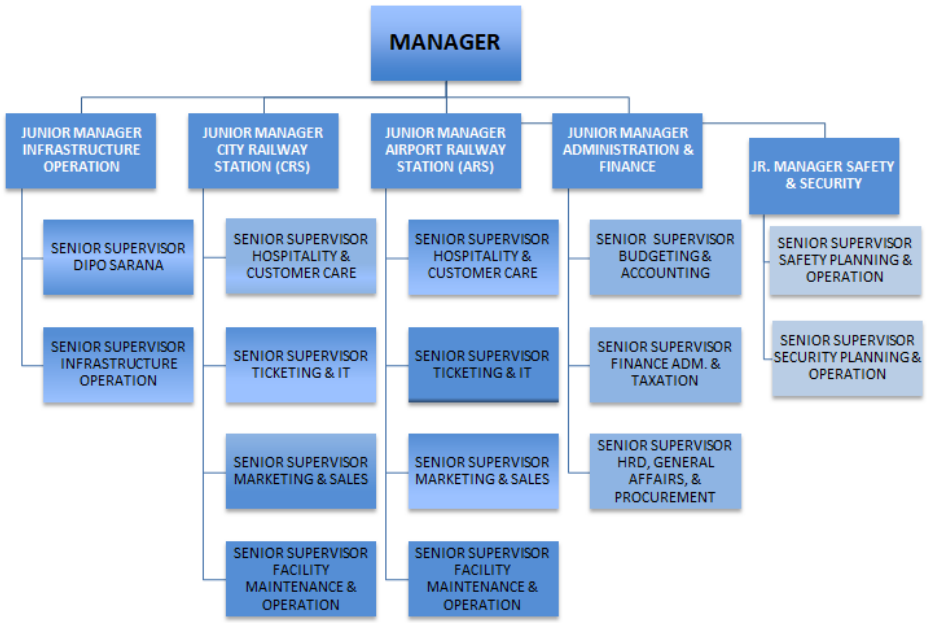
4.1.1.4 Struktur Organisasi Bisnis

PT Railink menggunakan struktur organisasi yang digunakan sebagai wadah operasi kegiatan semua unit di dalam rangka mencapai efektifitas dan efisiensi tujuan bisnis,. Penerapan sistem struktur organisasi bisnis sesuai bidang perkereta-apian bandara pada PT Railink digambarkan pada Gambar 4.3.



Gambar 4.3
Struktur Organisasi Kantor Pusat PT Railink

Kantor Pusat PT Railink memiliki dua Kantor Cabang yaitu Kantor Cabang Medan dan Kantor Cabang Jakarta. Struktur Organisasi untuk Kantor Cabang Medan digambarkan pada Gambar 4.4.



Gambar 4.4
Struktur Organisasi PT Railink Cabang Medan

Kantor Cabang Jakarta memiliki struktur organisasi seperti yang digambarkan pada Gambar 4.5

Gambar 4.5
Struktur Organisasi PT Railink Cabang Medan

4.1.1.5 *Service* Bisnis As-is

Analisis kebutuhan bisnis ditujukan untuk mengidentifikasi dukungan IT potensial atas strategi perusahaan sesuai dengan Rancangan Jangka Panjang Perusahaan (RJPP). Hasil identifikasi dukungan IT potensial akan digunakan untuk menyusun arsitektur IT yang menyeluruh dan terpadu. Analisis difokuskan pada inisiatif strategis atau permasalahan kritikal bidang sarana angkutan penumpang khususnya dalam sistem tiket yang relevan dengan dukungan IT potensial.

Dalam bidang komersial dan angkutan penumpang terdapat inisiatif bisnis atau permasalah kritis sebagai berkut:

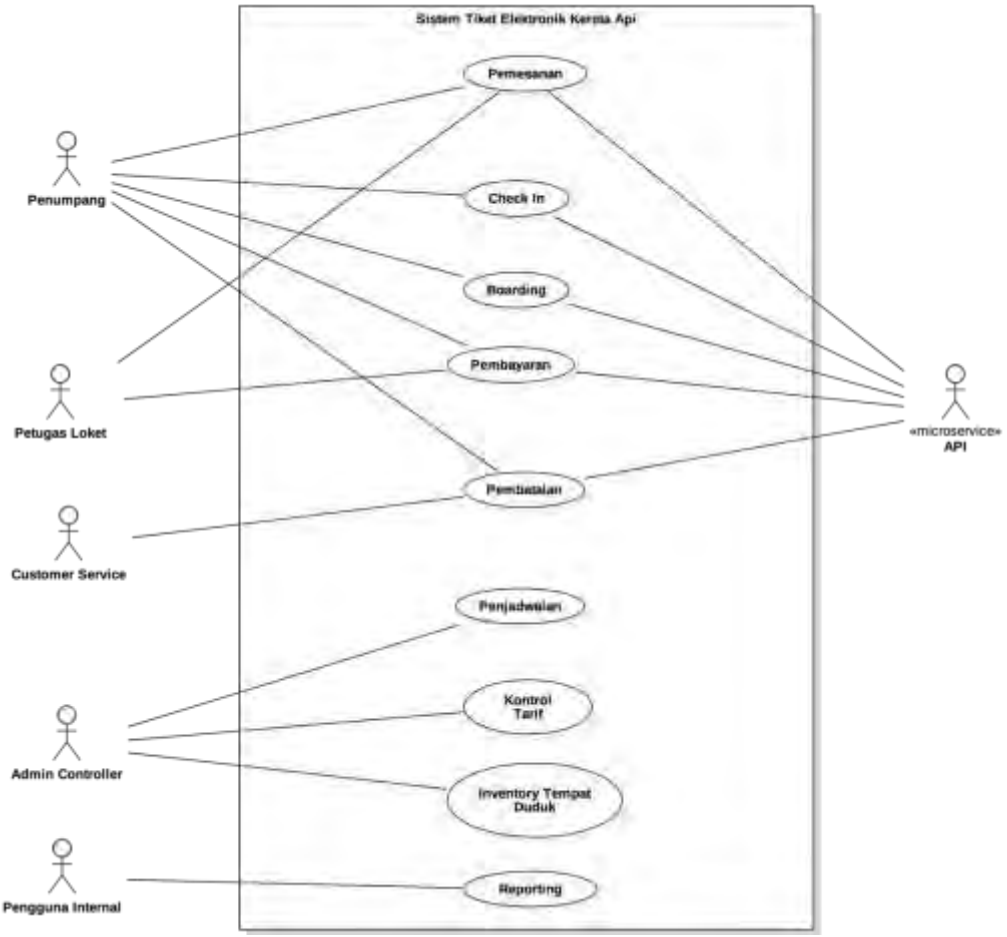
1. Peningkatkan kualitas pelayanan.
2. Peningkatkan penjualan tiket dan aksesibilitas.
3. Strategi pemasaran, sebagai berikut:
    a. Kemudahan mendapatkan tiket.
    b. Keunikan kegiatan eduwisata dengan KA Bandara.
    c. Keramahan petugas.
    d. Ketepatan dan kenyamanan dalam perjalanan.
4. Kampanye pemasaran dengan target pasar baru adalah komunitas dan *traveler*.
5. *Ticketing system*.
6. Mengakomodir kebutuhan tiket pasca keterlambatan penumpang maupun airline.

Dari inisiatif bisnis atau permasalahan kritis maka diperlukan dukungan potensial TI sebagai berikut:

1. *E-ticketing* dengan konsep *manless*, *cashless* dan *everywhere*. Layanan *eticketing* secara mandiri secara non tunai dan dapat dilakukan di Vending Machine, *Internet Booking* (*website*) serta *Mobile Application*.
2. Pembayaran secara non tunai menggunakan *prepaid card*, *debit card* dan *credit card*.
3. Sales analysis yang memungkinkan analisa hasil penjualan dan simulasi pentarifan.
4. Tiket VIP untuk penumpang khusus.

Proses-proses bisnis yang ada pada sistem *e-ticketing* dan persoalan yang sedang berjalan dan akan dideskripsikan dengan menggunakan *Use case Diagram*. Proses-proses

bisnis yang terjadi pada sistem *e-ticketing* kereta api saat ini dapat digambarkan pada Gambar 4.1 dengan *Use case Diagram*.



Gambar 4.6
Diagram Use case Proses Bisnis Sistem *E-ticketing* Kereta Api As-Is

Proses bisnis as-is secara global untuk penjualan tiket memiliki 3 (tiga) kelompok proses bisnis utama sebagai berikut:

a. Pemesanan dan Penjualan Tiket, serta Pembayaran untuk penumpang individu dan rombongan;
b. Pembatalan dan Pengembalian Bea (Refund);
c. Check In dan boarding untuk penumpang individu maupun rombongan;

Proses bisnis as-is untuk pengguna internal dan administrator memiliki 4 (empat) proses bisnis sebagai berikut:

a. Proses penjadwalan (*scheduling*)

b. Proses pentarifan

c. Pengendalian inventory tempat duduk

d. Pembuatan Laporan-laporan

Dari hasil identifikasi terhadap proses bisnis as-is, maka didapat beberapa gap kebutuhan bisnis pada sistem yang akan diimplementasikan sebagaimana dijelaskan pada tabel 4.2:

Tabel 4.2 Identifikasi *Gap* Kebutuhan Proses Bisnis

No.	**Masalah**	**Kondisi *As-Is***	**Kebutuhan TI**	***Gap***
1	Pemesanan dan pembayaran dilayani di Loket	Pemesanan dan Pembayaran yang dilakukan di stasiun yang masih dilayani di loket	Pelayanan pemesanan dan pembayaran yang cepat cukup dilakukan melalui *Vending Machine*.	Waktu pelayanan yang tidak cepat menimbulkan antrian calon penumpang pada jam-jam sibuk. Peniadaan petugas loket
2	Pembayaran tunai pada loket di stasiun	Pembayaran menggunakan uang tunai	Pembayaran dilakukan secara non-tunai untuk mempercepat proses transaksi pembayaran dan minimal kesalahan perhitungan manual uang tunai.	Mesin EDC tunggal untuk akomodasi transaksi non-tunai berbagai bank nasional.
3	Proses Check-in dan Boarding	Penumpang yang sudah memiliki tiket harus check-in dan boarding	Proses *gate-in* dijadikan satu proses baik check-in dan boarding untuk meringkas proses antrian penumpang masuk peron dan penumpang tidak menumpuk di stasiun.	Proses *gate-in* dan *gate-out*

4.1.2 Analisis Arsitektur TI As-Is

Analisis Arsitektur TI As-Is terdiri dari Analisis Arsitektur TI dengan identifkasi sumber daya TI teknis dan non-teknis. Analisis arstiektur TI As-Is mencakup Analis Arsitektur dari sisi sistem informasi dan infrastruktur.

4.1.2.1 Identifikasi Sumber Daya TI Teknis

Identifikasi sumber daya TI secara teknis akan dibagi ke dalam dua bagian, yaitu identifikasi pengembangan sistem As-Is dan identifikasi infrastruktur *as-is*.

4.1.2.1 Identifikasi Sistem *E-ticketing* As-Is

Identifikasi kondisi TI *As-Is* pada pemilik bisnis proses yaitu unit angkutan penumpang dalam sistem *e-ticketing* kereta Api yang dilakukan pada saat identikasi bisnis *As-Is* terdapat kondisi-kondisi sebagai berikut:

1. Sistem yang sedang berjalan dari sisi *back-end* dan *front-end* dibangun oleh pihak ke-tiga *(vendor)* dan sudah diimplementasikan di Kantor Cabang Area Medan.
2. Belum adanya pengembangan sistem tiket elektronik kereta api Bandara yang dikelola secara mandiri oleh perusahaan yang akan diimplementasikan baik di Cabang Medan dan Jakarta maupun Kantor Cabang lainnya.

Dengan kondisi tersebut, menghasilkan sebuah usulan kebutuhan TI yaitu Sistem Tiket Elektronik (*E-ticketing*) Kereta Api yang dibangun dan dikelola secara mandiri dengan adanya gap bahwa sistem *e-ticketing* yang dikembangkan in-house memiliki benefit yang signifikan untuk jangka panjang dan dalam rangka mengurangi ketergantungan dan biaya kepada pihak vendor pengembang pihak ketiga.

4.1.2.2 Identifikasi Sumber Daya Infrastruktur

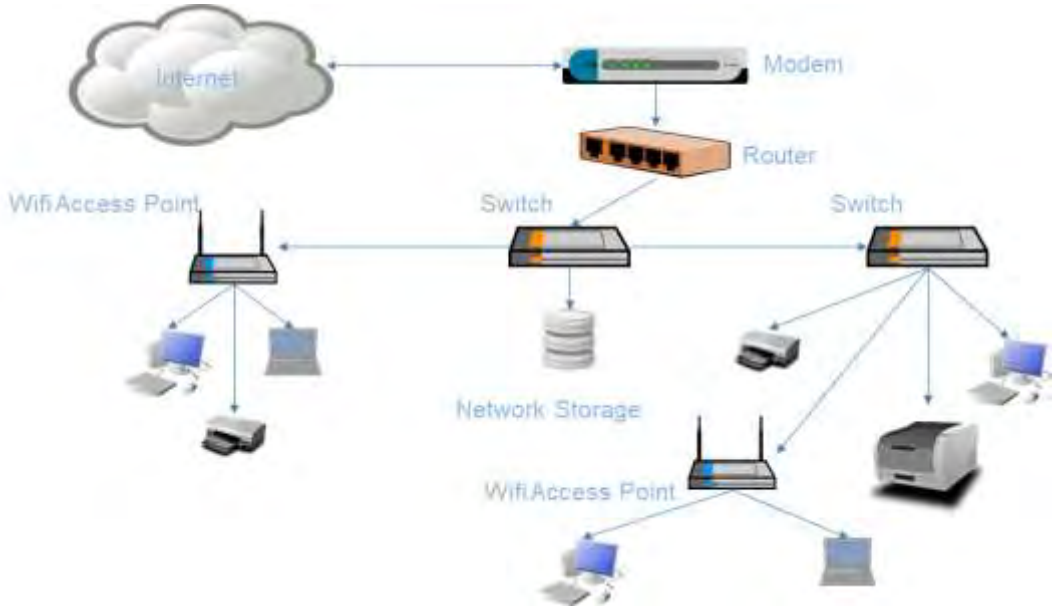
Identifikasi Kondisi TI dari sisi teknologi infrastruktur yang digunakan sebagaimana pada tabel 4.3:

Tabel 4.3 Identifikasi Infrastruktur Pendukung *E-ticketing As-Is*

Teknologi	**Usulan Kebutuhan TI**	**As-Is**	***Gap***
DC/DRC	*Full control*	Menggunakan resource induk Data Center PT Kereta Api Indonesia (Persero)	SDM TI
Server	*Full control*	Menggunakan resource induk Data Center PT Kereta Api Indonesia (Persero)	SDM TI

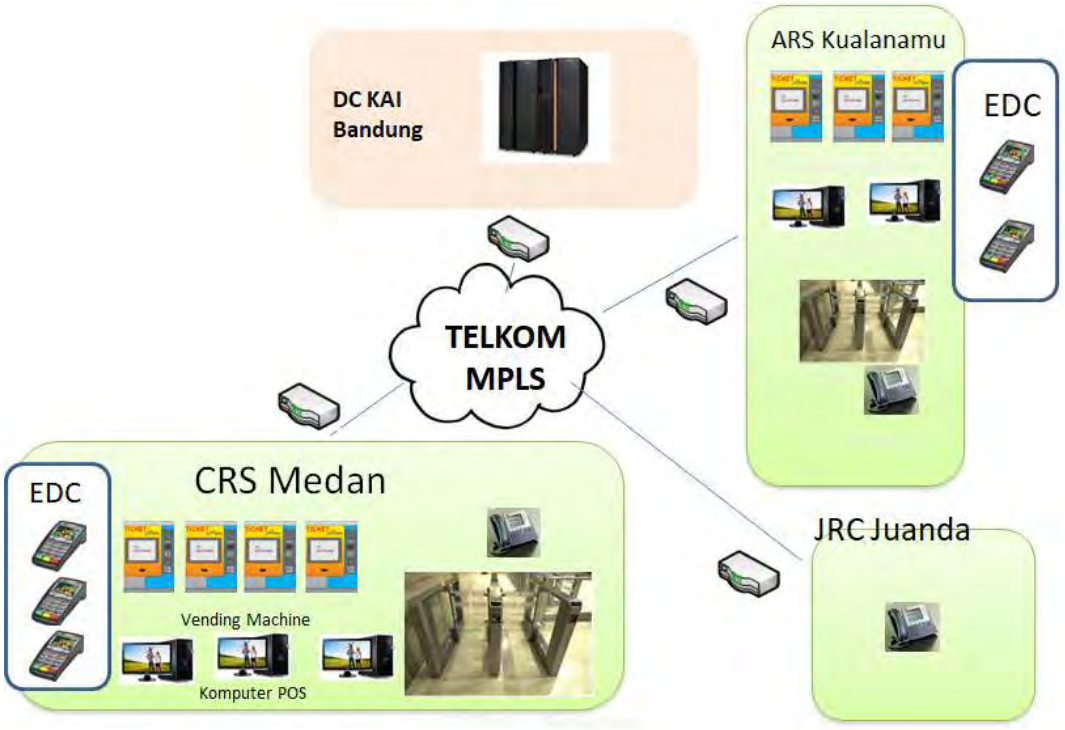
Teknologi	**Usulan Kebutuhan TI**	**As-Is**	***Gap***
LAN	a. *Unified wireless* secara menyeluruh. b. Gigabit device. c. Peningkatan *throughput*	Masih ada device 10/100 Mbps	Implementasi baru sesuai dengan perluasan kondisi organisasi/perusahaan.
WAN	*a. Redundant* *b. Support tenant*	Sebagian belum ada backup	a. Penambahan *link* b. Penambahan *bandwidth*
Manajemen Jaringan dan Infrastruktur	Tool *monitoring* infrastruktur dan jaringan		*Monitoring* infrastruktur dan jaringan.
Personal Computing	*Seat Management*	*Internal Asset*	Pengadaan *Seat Management*
IT Security	a. Perlindungan terhadap akses fisik maupun remote yang lebih baik. b. Antivirus *Corporate*	Mikrotik, Fortinet, antivirus retail	a. Tambahan antivirus corporate. b. Pengaman akses ke ruang server.

Topologi jaringan infrastruktur As-Is pada Kantor Pusat Jakarta digambarkan pada Gambar 4.7



Gambar 4.7
Topologi Jaringan Kantor Pusat dan Cabang Jakarta

Topologi jaringan infrastruktur As-Is pada Kantor Cabang Medan digambarkan pada Gambar 4.8



Gambar 4.8
Topologi Jaringan Cabang Medan

4.1.2.2 Identifikasi Sumber Daya TI Non-Teknis

Identifikasi Kondisi TI pada organisasi dan sumber daya manusia IT dijelaskan pada tabel 4.4 sebagai berikut:

Tabel 4.4 Identifikasi kondisi SDM TI Non-Teknis

Item	As-Is	Gap
Struktur Organisasi	Tersentralisasi untuk kantor pusat	Koordinasi dengan Kantor Cabang
Job Description	Job Description tiap unit	Dokumentasi job description untuk tiap bagian
Jumlah SDM	Beberapa bagian belum ada SDM	Pemenuhan SDM untuk tiap bagian
Kompetensi dan skill	Kompetensi yang dipetakan sesuai job description	-
Program pengembangan SDM TI		Program pengembangan SDM TI sesuai Job Description
KP SDM TI	Berdasarkan target penyelesaian proyek	KPI semua SDM TI sesuai KPI perusahaan.

4.2 Perancangan Arsitektur *To-be*

Perancangan arsitektur *to-be* dari sistem tiket elektronik kereta api pada PT Railink terdiri dari perancangan arsitektur bisnis *to-be* dan perancangan arsitektur TI *to-be*. Perancangan arsitektur *to-be* tersebut dijelaskan sebagai berikut:

4.2.1 Perancangan Arsitektur Bisnis *To-be*

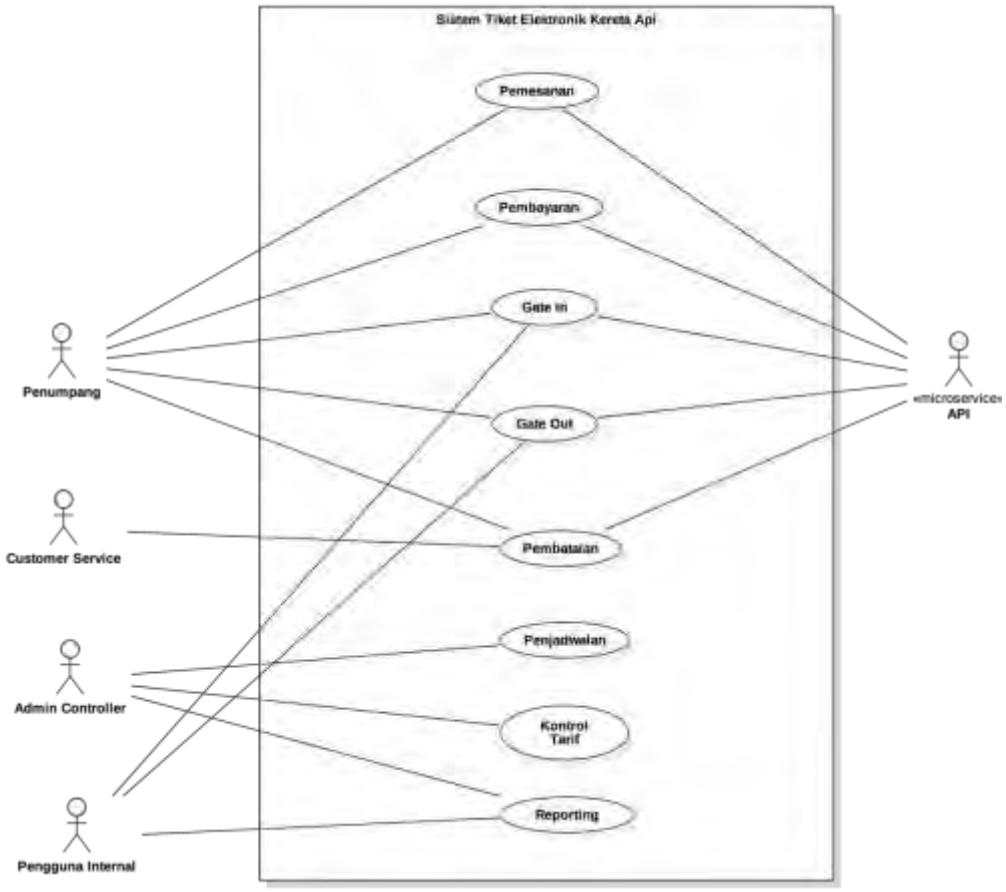
Perancangan arsitektur bisnis yang akan dijelaskan terdiri dari *service* bisnis *core*, *service* bisnis *non-core*, dan *service* bisnis eksternal. Sebelum masuk ke dalam perancangan arsitektur *service* terlebih dahulu akan dibahas mengenasi proses bisnis *to-be* yang akan digambarkan dalam diagram *use case* dan diagram *Activity*.

4.2.1 Proses Bisnis *To-be*

Di dalam perancangan proses binis *to-be*, untuk mengakomodasi kebutuhan sistem *e-ticketing* kereta api sebagaimana telah dijelaskan pada sub bab 4.1.2.1 mengenai identifikasi sistem *e-ticketing* as-is maka terdapat beberapa proses yang dihilangkan seperti proses boarding untuk meminimalisir waktu antrian penumpang. Pada proses bisnis *to-be* juga sudah menghilangkan peran petugas loket yang digantikan secara penuh oleh *Vending Machine*. Gambaran detil dari aktor, model fungsi bisnis, dan aktifitas dari masing-masing fungsi bisnis dijelaskan seperti berikut:

4.2.1.1 Diagram *Use case* Bisnis *To-be*

Diagram *use case* bisnis *to-be* diperlukan untuk memodelkan perilaku bisnis dari sistem *e-ticketing* kereta api. Diagram *use case* akan digunakan untuk mengetahu fungsi bisnis dan siapa yang berhak menggunakan fungsi tersebut. Dalam Gambar 4.9 dideskripsikan diagram *use case* bisnis *to-be* dari sistem *e-ticketing* kereta api.



Gambar 4.9
Diagram *Use Case* Proses Bisnis *E-ticketing* Kereta Api *To-be*

Pada tabel 4.5 dideskripsikan pendefinisian aktor pada sistem *e-ticketing* kereta api:

Tabel 4.5 Deskripsi definisi aktor diagram *use case to-be*

No.	Aktor	Deskripsi
1	Penumpang	Orang yang menumpang kereta api dan telah membayar tiket tetapi tidak termasuk mengoperasikan dan melayani di dalam kereta api
2	*Customer Service*	Petugas di stasiun keberangkatan kereta api yang bertugas melayani dan membantu penumpang dalam masalah tiket kereta api
3	Admin Controller	Petugas administrator yang bertugas merencanakan, menjadwal, mengendalikan, dan memastikan semua data yang dibutuhkan pada sistem *e-ticketing* tersedia
4	Pengguna Internal	Orang yang berkepentingan pada bagian tertentu di dalam perusahaan yang memiliki wewenang untuk mengakses fitur tertentu pada sistem *e-ticketing* kereta api
5	API Microservice	Protokol antarmuka pemrograman aplikasi standar di dalam *microservice* yang digunakan untuk mengintegrasikan beberapa bagian *service* aplikasi dan komponen eksternal serta dapat dihubungkan dalam berbagai keperluan *multi-platfofm*.

Pada tabel 4.6 dideskripsikan pendefinisian *use case* pada sistem *e-ticketing* kereta api:

Tabel 4.6 Deskripsi definisi *use case* pada diagram *use case* t*o-be*

No.	*Use case*	Deskripsi
1	Pemesanan	proses pemesanan tiket kereta api tertentu untuk keberangkatan pada masa tertentu yang diatur batas minimal dan batas maksimalnya melalui suatu parameter waktu. Pemesanan terdiri dari pemesanan untuk individu maupun rombongan. Kategori pemesanan dimasukkan ke dalam pemesanan rombongan apabila calon penumpang memesan untuk jumlah penumpang 3 (tiga) orang lebih. Pemesanan diatur sebagai berikut: 1. Kereta Api yang dapat dipesan adalah KA-KA yang ditetapkan dapat dilakukan pemesanan. 2. Parameter batasan waktu KA yang dapat dipesan dalam satuan jam a. Batas minimal 2 jam; b. Batas maksimal 2160 jam; c. Batas minimal dan maksimal diatur melalui variabel yang dapat diubah sesuai kebutuhan. 3. Proses pemesanan dari awal sampai dengan akhir dibatasi waktu, dengan parameter pengaturan waktu dalam satuan menit untuk setiap jenis pemesanan. Pilihan perjalanan terdiri dari : 1. Perjalanan sekali jalan 2. Perjalanan pergi pulang
2	Pembayaran	proses pemindahan sejumlah uang (secara non tunai) sebagai tanda bukti bayar tiket kereta api menggunakan media kartu untuk transaksi pada *vending machine*, serta *mobile* dan *internet banking* pada transaksi online.
3	*Gate-in*	proses penumpang atau petugas melakukan *tap-in* tiket atau kartu pada *gate* keberangkatan KA sebagai tanda bahwa penumpang atau petugas akan masuk ke peron untuk naik ke dalam KA
4	*Gate-out*	proses penumpang atau petugas melakukan tap-out tiket atau kartu pada *gate* kedatangan KA sebagai tanda bahwa penumpang atau petugas keluar dari KA untuk keluar dari peron
5	Pembatalan	proses pembatalan tiket atas permintaan penumpang. Pembatalan terdiri dari dua jenis, yaitu: 1. Refund, yaitu pengembalian bea tiket yang dibatalkan atas permintaan pembeli. Proses refund mengurangi pendapatan. 2. Reschedule, perubahan jadwal keberangkatan pada kereta api yang sama ataupun kereta api yang berbeda.
6	Penjadwalan	Proses yang dilakukan petugas admin controller dalam membuat alokasi: 1. Nomor, nama dan jenis kereta api,

No.	*Use case*	Deskripsi
		2. Sarana (gerbong penumpang) yang akan digunakan 3. Subklas dari sarana yang akan dijual 4. Blok jumlah tempat duduk dan pengalokasian tempat duduk sesuai subklasnya 5. Tarif setiap subklas 6. Daftar waktu dari perjalanan kereta api 7. Melakukan operasi jalan atau batal jadwal perjalanan kereta api yang sudah disusun sesuai instruksi warta maklumat KA yang dikeluarkan oleh unit operasional KA.
7	Kontrol Tarif	Merupakan proses pengunggahan (*upload*) tarif berdasarkan: 1. Tanggal awal dan tanggal akhir perjalanan KA 2. Tanggal awal dan tanggal akhir penjualan KA 3. Nomor KA 4. Subklas sarana
9	*Reporting*	Merupakan proses hasil dari pengolahan *query* transaksi-transaksi pemesananan pembelian tiket yang dapat dicetak maupun diekspor ke dalam bentuk file lain.

Berikut adalah skenario masing-masing *use case* yang telah didefiniskan sebelumnya:

1. *Use case* Pemesanan

   a. Use case Pemesanan melalui *Vending Machine*

   Skenario:

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. *Customer* memilih menu Pemesanan	
	2. Menampilkan Halaman Pemesanan Tiket
3. *Customer* menu pembelian tiket	
	4. Tampil Form Pembelian Tiket
5. Input Jumlah Penumpang dan Stasiun Tujuan	
	6. Submit data jumlah penumpang dan stasiun tujuan 7. Tampilkan jadwal, ketersediaan tempat duduk, dan tarif
8. Memilih jadwal keberangkatan	
	9. Melakukan operasi *booking*
	10. Menampilkan Form Data Penumpang
11. Input Data Pembeli	

Aksi Aktor	**Reaksi Sistem**
	12. Menampilkan konfirmasi pembelian tiket
13. Konfirmasi pembelian tiket	
	14. *Submit* untuk dilakukan pembayaran tiket

b. Use case Pemesanan melalui *Web* Reservasi

Skenario:

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Memilih menu reservasi	
	2. Menampilkan menu pemesanan
3. Memilih jenis perjalanan sekali jalan 4. Menginput stasiun asal, stasiun tujuan, tanggal keberangkatan, dan jumlah penumpang	
	5. *Submit* data
	6. Menampilkan jadwal, ketersediaan tempat duduk dan tarif
7. Memilih jadwal	
	8. Menampilkan Form Data Penumpang
9. Menginput nama, nomor handphone, dan email	
	10. Melakukan proses *booking*
	11. Menampilkan notifikasi *booking* sukses
12. Memilih tipe pembayaran	
	13. *Submit* untuk lanjut ke proses pembayaran
Skenario Alternatif	
3. Memilih jenis perjalanan pergi pulang 4. Menginput stasiun asal, stasiun tujuan, tanggal keberangkatan (tanggal pergi dan tanggal pulang) dan jumlah penumpang	
	5. *Submit* data

Aksi Aktor	Reaksi Sistem
	6. Menampilkan jadwal, ketersediaan tempat duduk dan tarif
7. Memilih jadwal	
	8. Menampilkan Form Data Penumpang
9. Menginput nama, nomor handphone, dan email	
	10. Melakukan proses *booking*
	11. Menampilkan notifikasi *booking* sukses
12. Memilih tipe pembayaran	
	13. Submit untuk lanjut ke proses pembayaran

c. *Use case* Pemesanan melalui Aplikasi Mobile

Skenario:

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Buka halaman login	
	2. Menampilkan Form Login
3. Input username dan password	
	4. Validasi login
	5. Menu Pilihan pemesanan
6. Memilih menu pemesanan tiket	
	7. Menampilkan form pemesanan
8. Memilih perjalanan sekali jalan	
9. Menginput stasiun asal, stasiun tujuan, tanggal keberangkatan, dan jumlah penumpang	
	10. *Submit* data
	11. Menampilkan jadwal keberangkatan, ketersediaan tempat duduk, dan tarif
12. Memilih jadwal keberangkatan	
	13. Menampilkan Form Data Penumpang

Aksi Aktor	Reaksi Sistem
14. Menginput nama, nomor handphone, dan email	
	15. Melakukan proses *Booking*
	16. Menampilkan notifikasi *booking* sukses
17. Memilih tipe pembayaran	
	18. *Submit* untuk lanjut ke proses pembayaran
Skenario Alternatif	
8. Memilih perjalanan sekali jalan	
9. Menginput stasiun asal, stasiun tujuan, tanggal keberangkatan, dan jumlah penumpang	
	10. *Submit* data
	11. Menampilkan jadwal keberangkatan, ketersediaan tempat duduk, dan tarif
12. Memilih jadwal keberangkatan	
	13. Menampilkan Form Data Penumpang
14. Menginput nama, nomor handphone, dan email	
	15. Melakukan proses *Booking*
	16. Menampilkan notifikasi *booking* sukses
17. Memilih tipe pembayaran	
	18. *Submit* untuk lanjut ke proses pembayaran

14. *Use case* Pembayaran

    a. *Use case* Pembayaran melalui Vending Machine

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Melakukan proses pemesanan melalui *vending machine*	
	2. Menampilkan menu pembayaran
3. Konfirmasi Pembayaran	
	4. Menampilkan info jumlah yang harus dibayar

Aksi Aktor	**Reaksi Sistem**
5. Memilih menu bayar	
	6. *Submit* data bayar ke mesin EDC
	7. EDC menerima jumlah bayar dari Vending Machine
8. *Swipe*/*Tap* kartu perbankan pada EDC	
	9. Proses Pembayaran
	10. Cetak tiket
Skenario Alternatif	
	10. Tiket tidak tercetak
11. Memilih menu cetak ulang tiket	
12. Validasi dengan menginput kode booking dan nomor handphone	
	13. Memvalidasi dan mencetak ulang tiket

b. *Use case* Pembayaran melalui *Web* Reservasi

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Melakukan proses pemesanan melalaui *web* reservasi	
	2. Menampilkan menu pembayaran
3. Konfirmasi Pembayaran	
	4. Menampilkan info jumlah yang harus dibayar
5. Memilih Pembayaran melalui e-commerce	
	6. *Submit* data pembayaran e-commerce
7. Menginput data pembayaran e-commerce	
	8. *Submit* data pembayaran
	9. Menampilkan notifikasi sukses pembayaran
	10. Membuat email bukti pembayaran dan barcode tiket
11. Menerima email notifikasi pembayaran dan barcode tiket	
Skenario Alternatif	

Aksi Aktor	**Reaksi Sistem**
5. Memilih Pembayaran melalui *channel* eksternal	
	6. Pembayaran diterima
	7. Membuat email bukti pembayaran dan barcode tiket
8. Menerima email notifikasi pembayaran dan barcode tiket	

c. *Use case* Pembayaran melalui Aplikasi Mobile

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Melakukan proses pemesanan melalaui aplikasi mobile	
	2. Menampilkan menu pembayaran
3. Konfirmasi Pembayaran	
	4. Menampilkan info jumlah yang harus dibayar
5. Memilih Pembayaran melalui e-commerce	
	6. *Submit* data pembayaran e-commerce
7. Menginput data pembayaran e-commerce	
	8. *Submit* data pembayaran
	9. Menampilkan notifikasi sukses pembayaran
	10. Membuat email bukti pembayaran dan barcode tiket
11. Menerima email notifikasi pembayaran dan barcode tiket	
Skenario Alternatif	
12. Memilih Pembayaran melalui *channel* eksternal	
	13. Pembayaran diterima
	14. Membuat email bukti pembayaran dan barcode tiket
15. Menerima email notifikasi pembayaran dan barcode tiket	

15. *Use case Gate-in*

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Menempelkan barcode tiket pada gate	
	2. Gate terbuka
3. Masuk peron	
Skenario Alternatif	
	2. Menampilkan pesan error
	3. Gate tidak terbuka

16. *Use case Gate-out*

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
4. Menempelkan barcode tiket pada gate	
	5. Gate terbuka
6. Keluar peron	
Skenario Alternatif	
	4. Menampilkan pesan error
	5. Gate tidak terbuka

17. *Use case* Pembatalan

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Membuka menu pembatalan	
	2. Menampilkan form pembatalan
3. Menginput nomor handphone dan nomor tiket	
	4. *Submit* data nomor handphone dan nomor tiket
	5. Menghitung bea pembatalan

Aksi Aktor	**Reaksi Sistem**
	6. Menampilkan *summary* perhitungan bea dan form pembatalan
7. Input data pembatalan	
	8. Submit data pembatalan
	9. Mencetak bukti pembatalan
	10. Mengirim email bukti pembatalan
11. Menerima email bukti pembatalan	

18. *Use case* Kontrol tarif

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Membuka menu *setting* tarif	
	2. Menampilkan *list* tarif
3. Memilih *download template* tarif	
	4. *Download template* tarif
Skenario Alternatif	
3. Memilih *upload* tarif	
	4. Menampilkan Form *Upload* Tarif
5. Melakukan *upload file* tarif	
	6. *Submit* file tarif
	7. Menampilkan notifikasi hasil *upload file* tarif

19. *Use case Reporting*

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Memilih menu *reporting*	
	2. Menampilkan Filter Pencarian *Report*

Aksi Aktor	**Reaksi Sistem**
3. Memilih Filter Pencarian *Report*	
	4. Submit filter pencarian *report*
	5. Melakuan *query* berdasarkan parameter pencarian
	6. Menampilkan *report* hasil *query*
7. Melakukan export hasil *report*	

4.2.1.2 *Activity Diagram* Proses Bisnis *To-be*

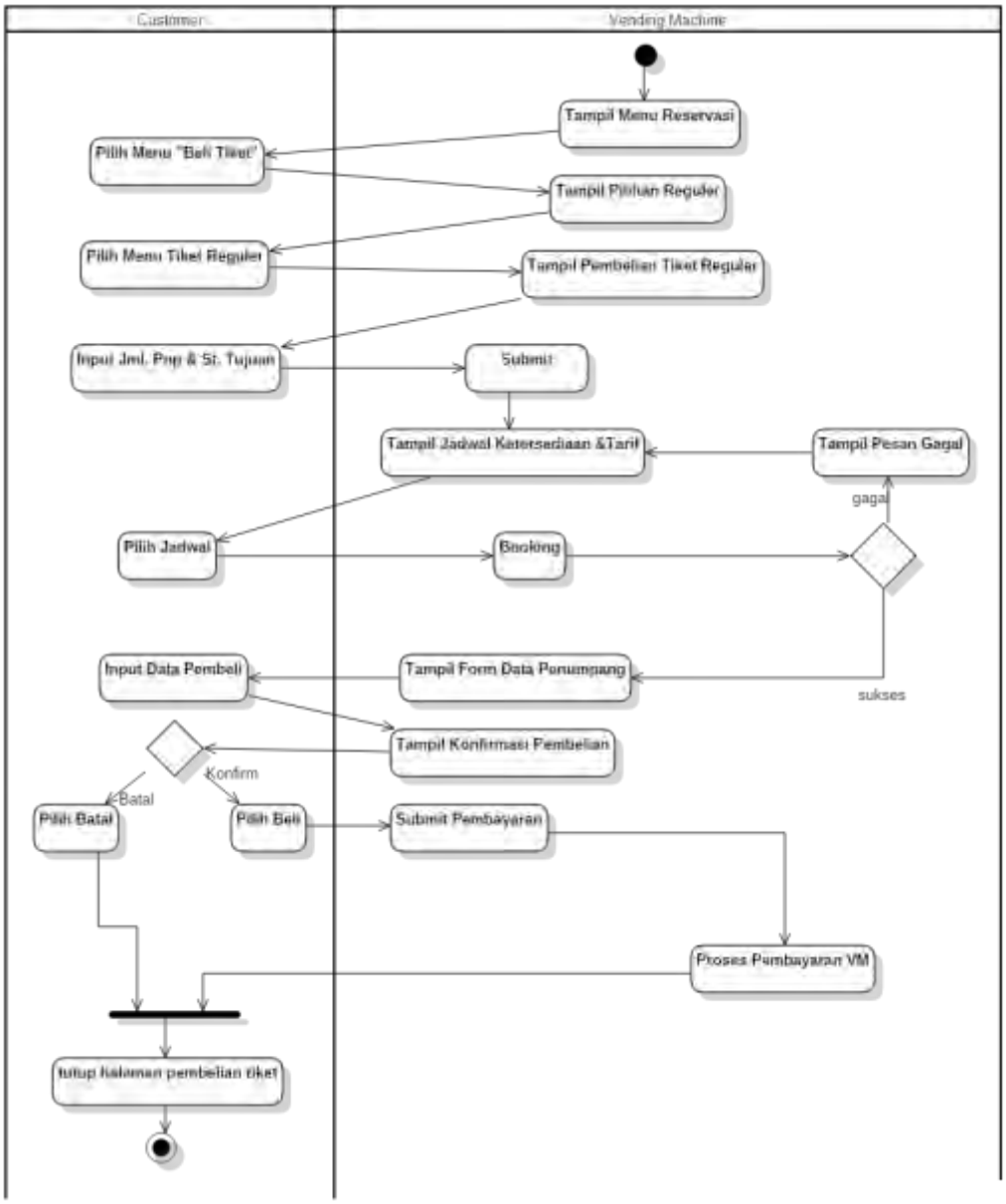
Dari fungsi-fungsi bisnis yang terdapat dalam Diagram *Use case To-be* Sistem *E-ticketing* Kereta Api (Gambar 4.8)

1. Diagram *Activity* Proses Pemesanan

   Proses pemesanan tiket dapat dilakukan pada vending machine, *web* reservasi, dan aplikasi mobile. Berikut diagram *Activity* dari masing-masing proses pemesanan.

   a. Diagram *Activity* pemesanan melalui *vending machine*

      Customer selaku pengguna kereta api bandara terlebih dahulu melakukan pemesanan secara langsung yang akan membeli tiket pada hari itu juga menggunakan vending machine yang tersedia di stasiun-stasiun keberangkatan. Diagram *Activity* dari proses pemesanan melalui vending machine dapat dilihat pada gambar 4.10



Gambar 4.10
Diagram *Activity* Pemesanan melalui Vending Machine

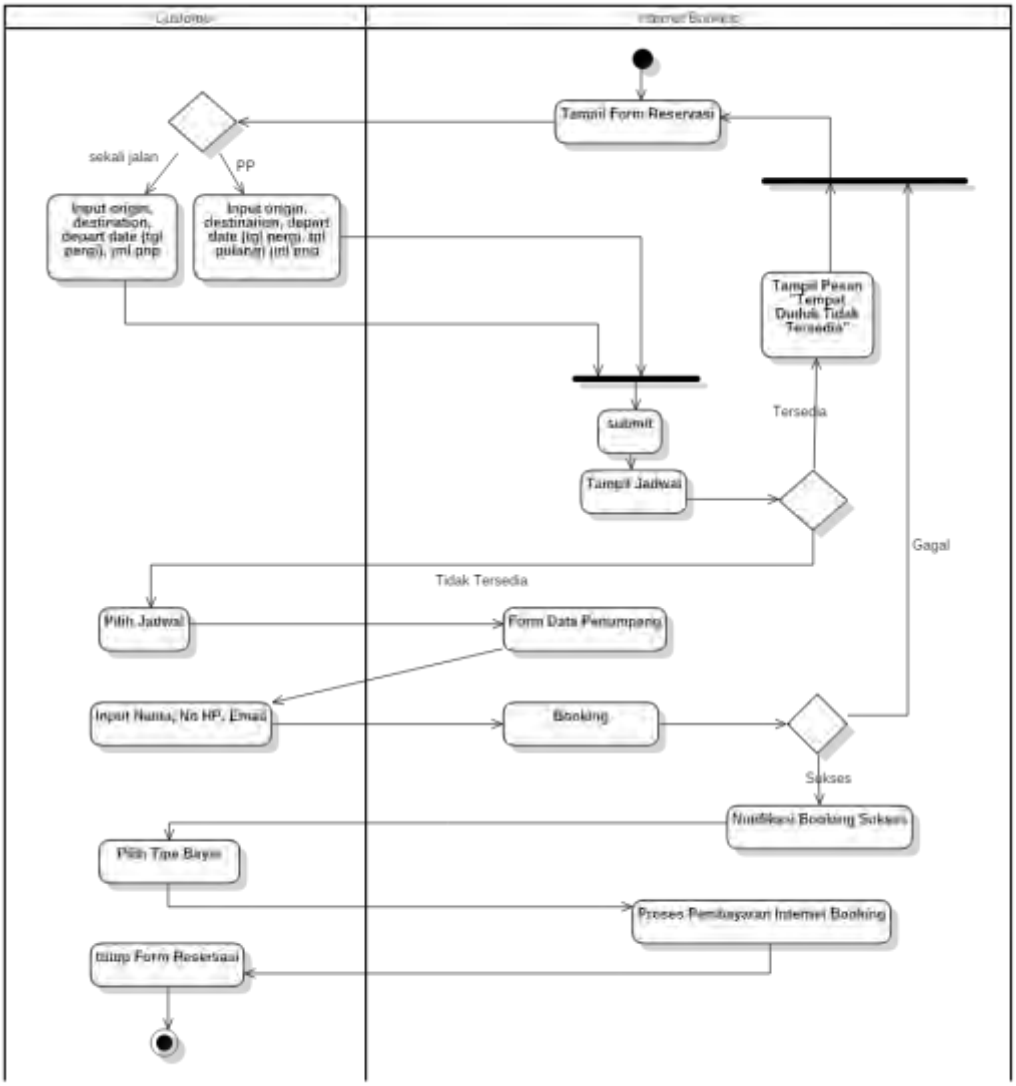
Aliran aktivitas dari pemesanan melalui vending machine tersebut dapat dijelaskan sebagai berikut:

1) Sistem menampilkan menu reservasi
2) *Customer* memilih menu pembelian tiket
3) Sistem menampilkan pilihan jenis tiket reguler
4) *Customer* memilih jenis tiket reguler

5) Sistem menampilkan form pembelian tiket reguler
6) *Customer* menginput jumlah penumpang dan stasiun yang dituju
7) Sistem melakukan submit data jumlah penumpang dan stasiun yang dituju
8) Sistem menampilkan jadwal yan tersedia, ketersediaan tempat duduk, dan tarif
9) *Customer* memilih jadwal keberangkatan
10) Sistem melakukan proses *booking*
11) Jika proses booking berhasil, sistem menampilkan form isian data penumpang. Jika proses *booking* gagal, sistem menampilkan proses *booking* gagal.
12) *Customer* melakukan input data penumpang
13) Submit data penumpang
14) Sistem menampilkan form konfirmasi data pemesanan
15) Jika *customer* mengkonfirmasi data pemesanan, sistem mengirimkan data pemesanan yang dilanjutkan dengan proses pembayaran. Jika *customer* membatalkan data pemesanan, halaman pemesanan tiket ditutup.

b. Diagram *Activity* pemesanan via *web* reservasi (*internet booking*)

Pemesanan yang dilakukan melalui *web* secara *online* dapat dilakukan melaui sebuah alamat *web* pemesanan resmi yang telah ditentukan. Pemesanan tiket *online* sebagai fasilitas yang diberikan untuk meminimalkan adanya antrian pembelian tiket secara langsung di stasiun keberangkatan (menggunakan *vending machine*). Diagram *Activity* dari proses pemesanan melalui *web* reservasi dapat dilihat pada gambar 4.11



Gambar 4.11
Diagram *Activity* Pemesanan melalui Web Reservasi

Aliran aktivitas dari pemesanan melalui *web* reservasi dapat dijelaskan sebagai berikut:

1) Sistem menampilkan Form Reservasi
2) *Customer* dapat memilih jenis perjalanan untuk sekali jalan atau perjalanan pergi pulang (pp). Jika *customer* memilih jenis perjalanan sekali jalan, maka *customer* melakukan input data stasiun asal, stasiun tujuan, tanggal keberangkatan, dan jumlah penumpang. Dan jika *customer* memilih jenis perjalanan pergi pulang, maka *customer* menginput data

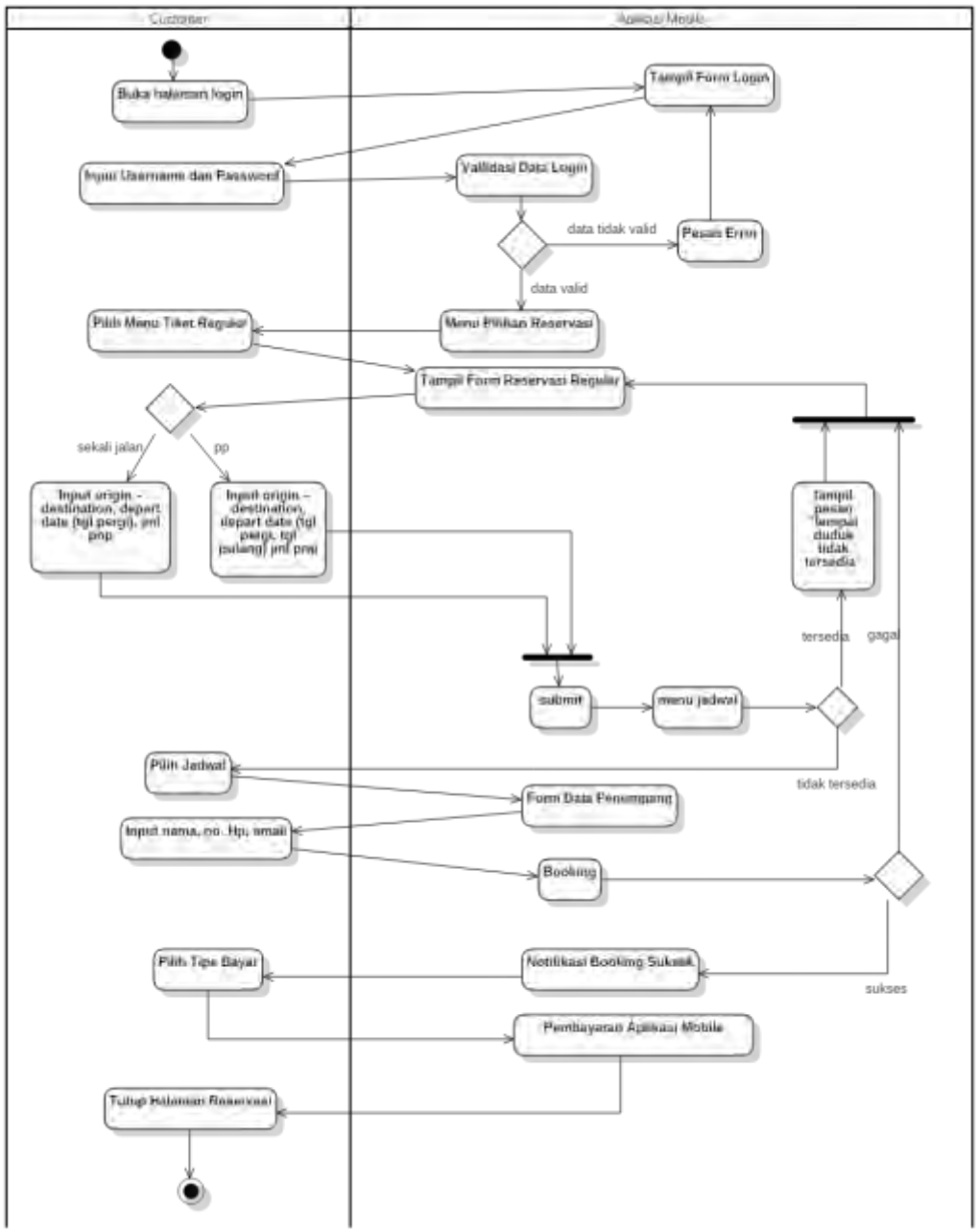
stasiun asal, stasiun tujuan, tanggal keberangkatan, tanggal kepulangan, dan jumlah penumpang.

3) Sistem melakukan *submit* data jenis perjalanan, stasiun asal dan tujuan, tanggal keberangkatan dan atau tanggal kepulangan, serta jumlah penumpang.
4) Jika jadwal tidak tersedia atau tempat duduk tersedia, sistem menampilkan pesan "jadwal/tempat duduk tidak tersedia. Jika jadwal tersedia, sistem akan menampilkan jadwal yang tersedia, ketersediaan kursi dan tarif.
5) *Customer* menginput jumlah penumpang dan stasiun yang dituju
6) Sistem melakukan submit data jumlah penumpang dan stasiun yang dituju
7) Sistem menampilkan jadwal yan tersedia, ketersediaan tempat duduk, dan tarif
8) *Customer* memilih jadwal keberangkatan
9) Sistem melakukan proses *booking*
10) Jika proses booking berhasil, sistem menampilkan form isian data penumpang. Jika proses *booking* gagal, sistem menampilkan proses *booking* gagal.
11) *Customer* melakukan input data penumpang
12) Submit data penumpang
13) Sistem menampilkan form konfirmasi data pemesanan
14) Jika *customer* mengkonfirmasi data pemesanan, sistem mengirimkan data pemesanan yang dilanjutkan dengan proses pembayaran. Jika *customer* membatalkan data pemesanan, halaman pemesanan tiket ditutup.

c. Diagram *Activity* pemesanan via aplikasi mobile

Pemesanan yang dilakukan melalui aplikasi *mobile* dilakukan melalui aplikasi mobile yang didistribusikan dalam paket aplikasi android dan ios. Pemesanan tiket *mobile* sebagai fasilitas yang diberikan untuk meminimalkan adanya antrian pembelian tiket secara langsung di stasiun keberangkatan

(menggunakan *vending machine*). Diagram *Activity* dari proses pemesanan melalui aplikasi *mobile* dapat dilihat pada gambar 4.12



Gambar 4.12
Diagram *Activity* Pemesanan melalui Aplikasi Mobile

Aliran aktivitas dari pemesanan melalui aplikasi mobile dapat dijelaskan sebagai berikut:

1) *Customer* membuka halaman login

2) Sistem menampilkan form login
3) User menginputkan username dan password
4) Sistem melakukan validasi data login
5) Jika data login tidak valid, sistem menampilkan pesan error validasi login. Jika data login valid, maka ditampilkan menu reservasi
6) User memilih menu tiket reguler
7) Sistem menampilkan form reservasi reguler
8) *Customer* dapat memilih jenis perjalanan untuk sekali jalan atau perjalanan pergi pulang (pp). Jika *customer* memilih jenis perjalanan sekali jalan, maka *customer* melakukan input data stasiun asal, stasiun tujuan, tanggal keberangkatan, dan jumlah penumpang. Dan jika *customer* memilih jenis perjalanan pergi pulang, maka *customer* menginput data stasiun asal, stasiun tujuan, tanggal keberangkatan, tanggal kepulangan, dan jumlah penumpang.
9) Sistem melakukan *submit* data jenis perjalanan, stasiun asal dan tujuan, tanggal keberangkatan dan atau tanggal kepulangan, serta jumlah penumpang.
10) Jika jadwal tidak tersedia atau tempat duduk tersedia, sistem menampilkan pesan “jadwal/tempat duduk tidak tersedia. Jika jadwal tersedia, sistem akan menampilkan jadwal yang tersedia, ketersediaan kursi dan tarif.
11) *Customer* menginput jumlah penumpang dan stasiun yang dituju
12) Sistem melakukan submit data jumlah penumpang dan stasiun yang dituju
13) Sistem menampilkan jadwal yan tersedia, ketersediaan tempat duduk, dan tarif
14) *Customer* memilih jadwal keberangkatan
15) Sistem melakukan proses *booking*
16) Jika proses booking berhasil, sistem menampilkan form isian data penumpang. Jika proses *booking* gagal, sistem menampilkan proses *booking* gagal.

17) *Customer* melakukan input data penumpang

18) Submit data penumpang

19) Sistem menampilkan form konfirmasi data pemesanan

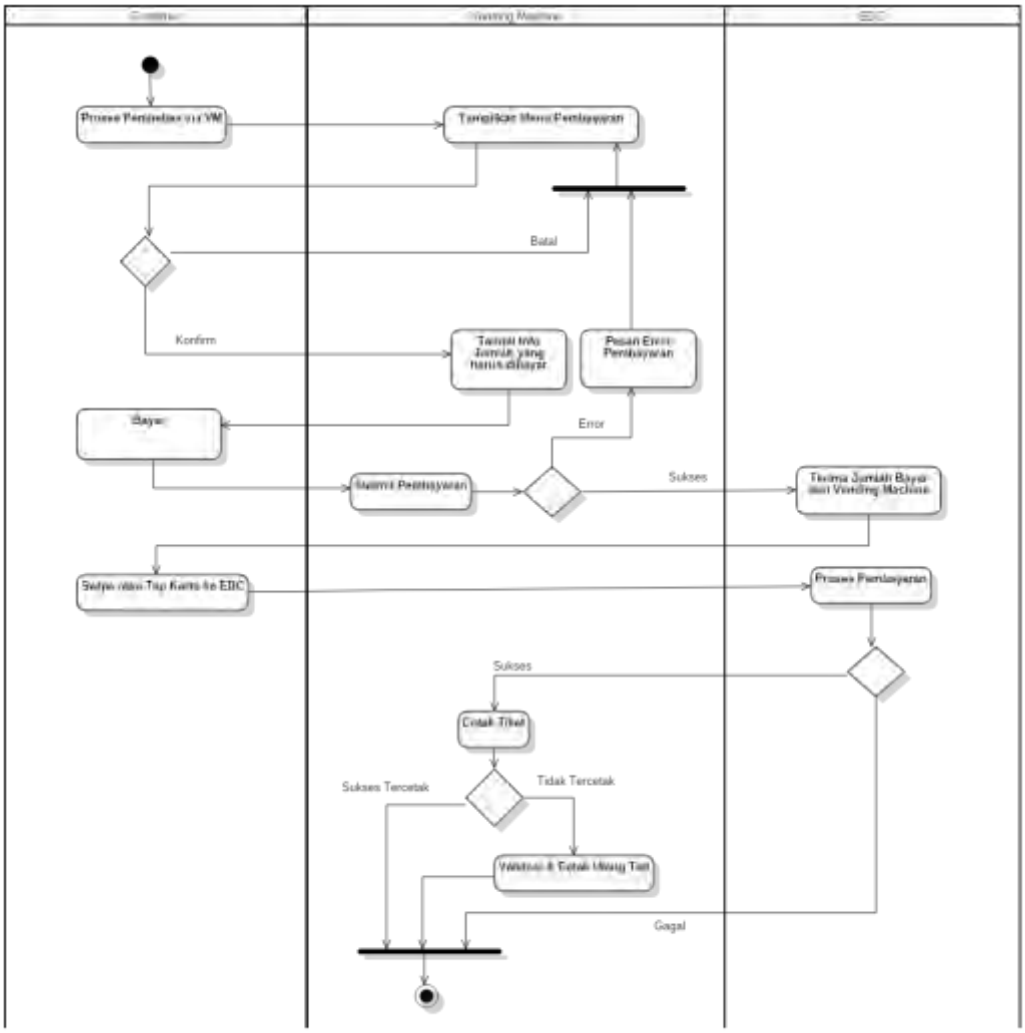
20) Jika *customer* mengkonfirmasi data pemesanan, sistem mengirimkan data pemesanan yang dilanjutkan dengan proses pembayaran. Jika *customer* membatalkan data pemesanan, halaman pemesanan tiket ditutup.

2. Diagram *Activity* Proses Pembayaran

Dari setiap proses pemesanan baik menggunakan vending machine, *web* reservasi dan aplikasi *mobile*, setiap pemesanan tiket yang akan dibeli harus dilakukan proses pembayaran. Berikut diagram *Activity* proses pembayaran untuk masing-masing proses pemesanan tersebut:

a. Diagram *Activity* Pembayaran menggunakan *Vending Machine*

Pembayaran tiket yang dilakukan pada vending machine dilakukan saat itu juga setelah *customer* melakukan pemesanan tiket. Diagram *Activity* dari proses pembayaran melalui *vending machine* dapat dilihat pada gambar 4.13



Gambar 4.13
Diagram *Activity* Pembayaran melalui Vending Machine

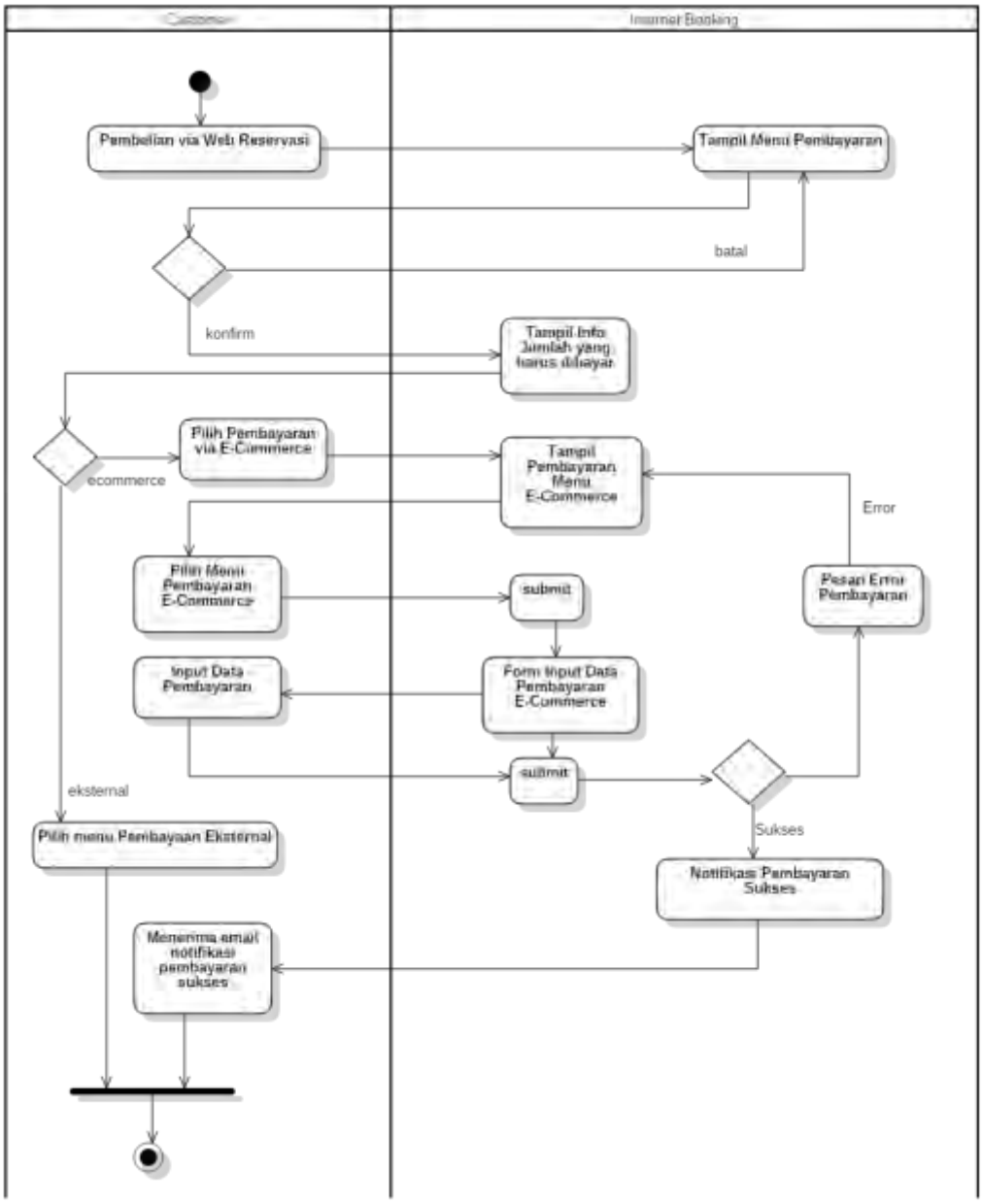
Aliran aktivitas dari pembayaran melalui aplikasi mobile dapat dijelaskan sebagai berikut:

1) *Customer* melakukan pemesanan tiket melalui *vending machine*
2) Sistem menampilkan menu pembayaran tiket
3) Jika *customer* membatalkan pembayaran, sistem akan mengembalikan tampilan ke menu pembayaran. Jika *customer* mengkonfirmasi pembelian tiket, sistem akan menampilkan nominal jumlah yang harus dibayar.
4) *Customer* memilih menu bayar tiket
5) Sistem melakukan *submit* data pembayaran

6) Jika hasil *submit* gagal, sistem menampilkan pesan error pembayaran. Jika hasil *submit* sukses, mesin EDC akan menerima jumlah bayar dari *vending machine*.
7) *Customer* melakukan *swipe* untuk kartu *magnetic* atau memasukkan kartu untuk kartu *chip* pada mesin EDC. Kartu yang dapat digunakan berupa kartu debit, kredit maupun prabayar dari bank.
8) EDC memproses pembayaran tiket
9) Jika proses pembayaran oleh EDC gagal, sistem menampilkan pesan pembayaran gagal. Jika proses pembayaran oleh EDC sukses, maka *vending machine* melakukan proses cetak tiket.
10) Jika tiket tidak tercetak, dilakukan proses validasi dan cetak ulang tiket.

b. Diagram *Activity* Pembayaran melalui *Web* Reservasi

Pembayaran tiket yang dilakukan pada *web* reservasi dilakukan setelah *customer* melakukan pemesanan tiket melalui *web* reservasi pada alamat *web* yang telah ditentukan. Diagram *Activity* dari proses pembayaran melalui *web* reservasi dapat dilihat pada gambar 4.14



Gambar 4.14
Diagram *Activity* Pembayaran melalui Web Reservasi

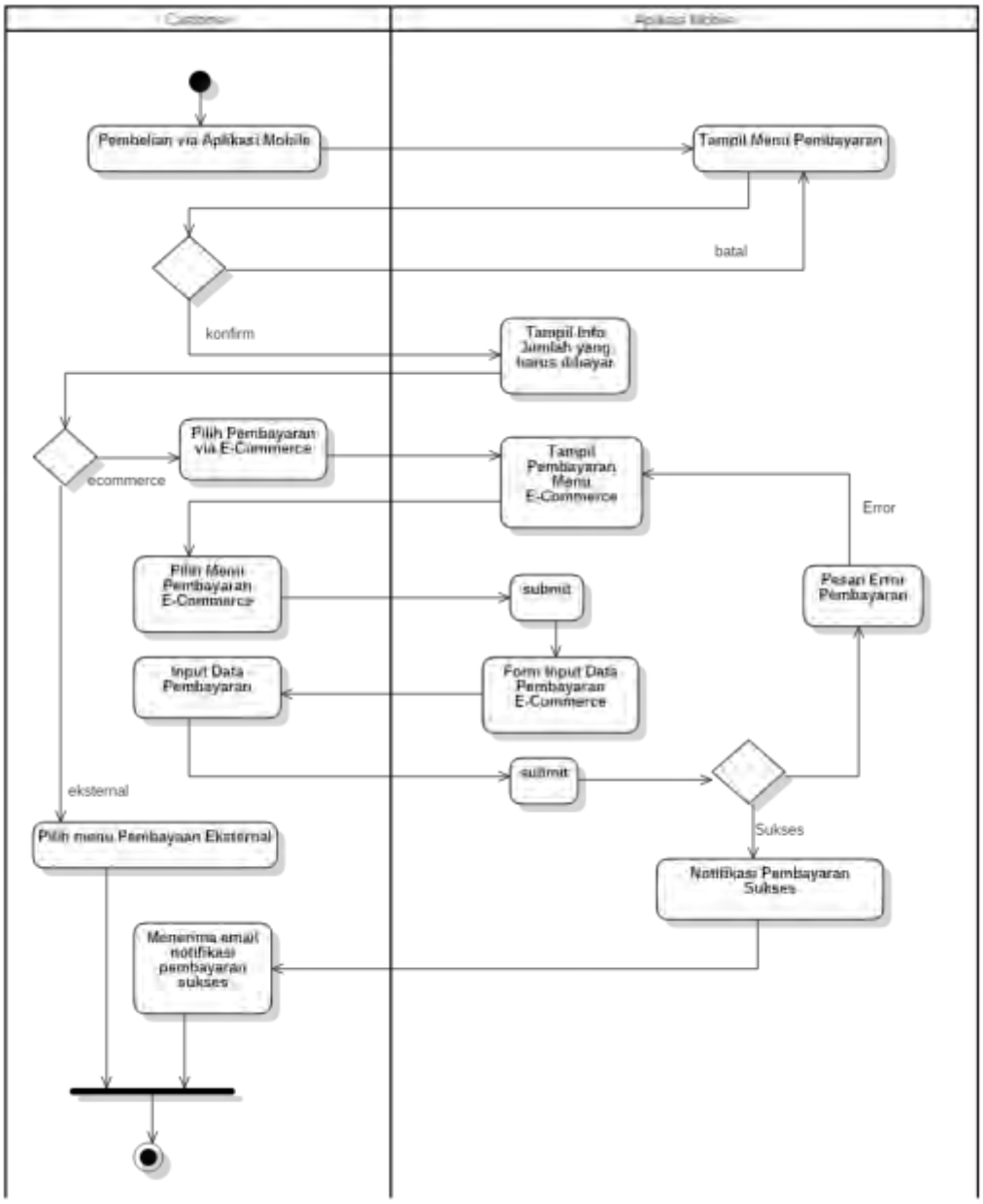
Aliran aktivitas dari pembayaran melalui *web* reservasi dapat dijelaskan sebagai berikut:

1) *Customer* melakukan pemesanan tiket melalui *web* reservasi
2) Sistem menampilkan menu pembayaran tiket

3) Jika *customer* membatalkan pembayaran, sistem akan mengembalikan tampilan ke menu pembayaran. Jika *customer* mengkonfirmasi pembelian tiket, sistem akan menampilkan nominal jumlah yang harus dibayar.
4) *Customer* dapat memilih menu pembayaran baik melalui *e-commerce* maupun pembayaran eksternal (seperti contohnya melalui mesin ATM dan *Virtual Account*).
5) Jika *customer* melakukan pembayaran secara eksternal, sistem akan mengirimkan email dan kode pembayaran.
6) Jika *customer* melakukan pembayaran secara e-commerce, sistem akan menampilkah halaman pembayaran e-commerce
7) *Customer* memilih channel pembayaran e-commerce yang diinginkan
8) Sistem melakukan *submit* channel pembayaran e-commerce.
9) Sistem akan menampilkan form input data pembayaran tiket e-commerce.
10) *Customer* melakukan input data pembayaran tiket
11) Sistem melakuan *submit* data pembayaran tiket
12) Jika hasil *submit* data pembayaran tiket gagal, sistem menampilkan pesan error pembayaran. Jika hasil *submit* sukses, sistem akan menampilkan notifikasi sukses pembayaran dan mengirimkan email notifikasi disertai *barcode* tiket.
13) *Customer* menerima email notifikasi pembayaran

c. Diagram *Activity* Pembayaran menggunakan Aplikasi *Mobile*

Pembayaran tiket yang dilakukan pada aplikasi *mobile* dilakukanan setelah *customer* melakukan pemesanan tiket melalui aplikasi *mobile* pada h. Diagram *Activity* dari proses pembayaran melalui aplikasi *mobile* dapat dilihat pada gambar 4.15



Gambar 4.15
Diagram *Activity* Pembayaran melalui Aplikasi Mobile

Aliran aktivitas dari pembayaran melalui aplikasi mobile dapat dijelaskan sebagai berikut:

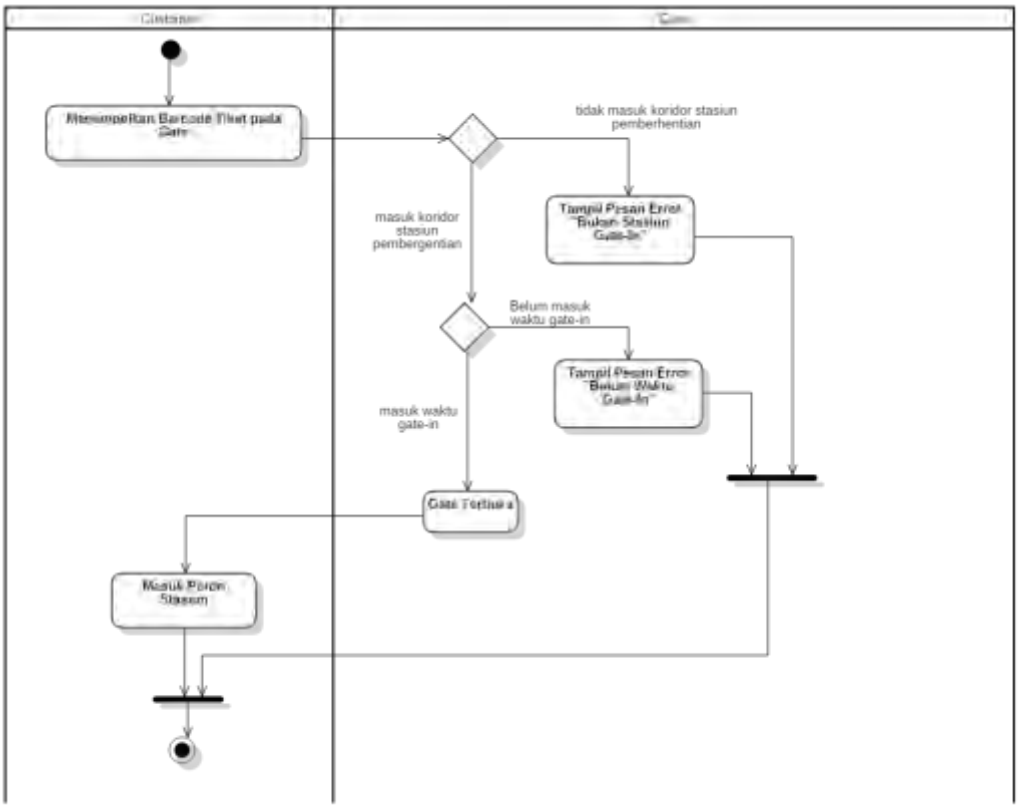
1) *Customer* melakukan pemesanan tiket melalui aplikasi *mobile*

2) Sistem menampilkan menu pembayaran tiket

3) Jika *customer* membatalkan pembayaran, sistem akan mengembalikan tampilan ke menu pembayaran. Jika *customer* mengkonfirmasi pembelian tiket, sistem akan menampilkan nominal jumlah yang harus dibayar.
4) *Customer* dapat memilih menu pembayaran baik melalui *e-commerce* maupun pembayaran eksternal (seperti contohnya melalui mesin ATM dan *Virtual Account*).
5) Jika *customer* melakukan pembayaran secara eksternal, sistem akan mengirimkan email dan kode pembayaran.
6) Jika *customer* melakukan pembayaran secara e-commerce, sistem akan menampilkah halaman pembayaran e-commerce
7) *Customer* memilih channel pembayaran e-commerce yang diinginkan
8) Sistem melakukan *submit* channel pembayaran e-commerce.
9) Sistem akan menampilkan form input data pembayaran tiket e-commerce.
10) *Customer* melakukan input data pembayaran tiket
11) Sistem melakuan *submit* data pembayaran tiket
12) Jika hasil *submit* data pembayaran tiket gagal, sistem menampilkan pesan error pembayaran. Jika hasil *submit* sukses, sistem akan menampilkan notifikasi sukses pembayaran dan mengirimkan email notifikasi disertai *barcode* tiket.
13) *Customer* menerima email notifikasi pembayaran

3. Diagram *Activity* Proses *Gate-in*

Proses *gate-in* dapat dilakukan setelah *customer* membeli tiket. Tiket yang telah tercetak atau dalam bentuk file *barcode* ditempelkan pada *gate* stasiun keberangkatan. Diagram *Activity* dari proses *gate-in* dapat dilihat pada gambar 4.16



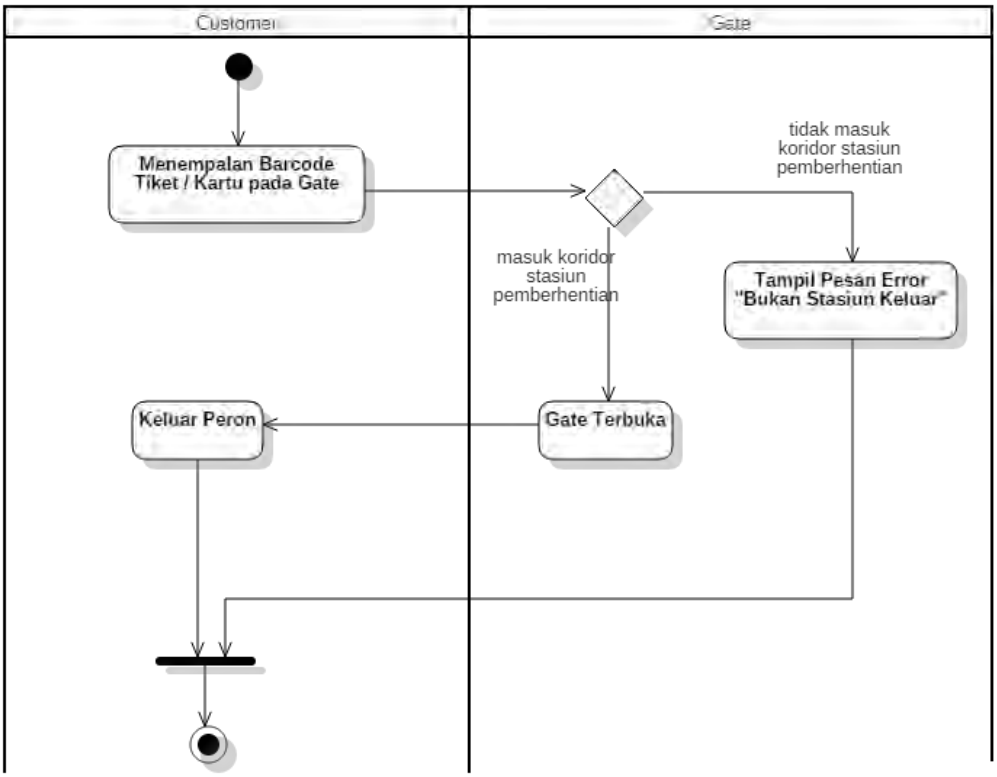
Gambar 4.16
Diagram *Activity* Proses Gate-in

Aliran aktivitas dari proses *gate-in* penumpang di stasiun keberangkatan dijelaskan sebagai berikut:

1) *Customer* menempelkan barcode tiket pada gate di stasiun keberangkatan kereta api
2) Jika status stasiun tidak masuk dalam koridor stasiun pemberhentian, maka sistem menampilkan pesan error “bukan stasiun keberangkatan”
3) Jika waktu proses *gate-in* belum masuk waktunya, sistem akan menampilkan pesan error “belum waktu *gate-in*”
4) Jika status stasiun masuk dalam koridor stasiun pemberhentian dan waktu *gate-in* sudah masuk waktunya, sistem akan membuka gate di stasiun keberangkatan.
5) *Customer* memasuki peron dan menunggu kedatangan kereta api.

4. Diagram *Activity* Proses *Gate-out*

Proses *gate-out* dilakukan setelah *customer* tiba di stasiun kedatangan. Tiket yang telah tercetak atau dalam bentuk file *barcode* ditempelkan pada *gate* stasiun kedatangan (stasiun tujuan). Diagram *Activity* dari proses *gate-out* dapat dilihat pada gambar 4.17



Gambar 4.17
Diagram *Activity* Proses Gate-out

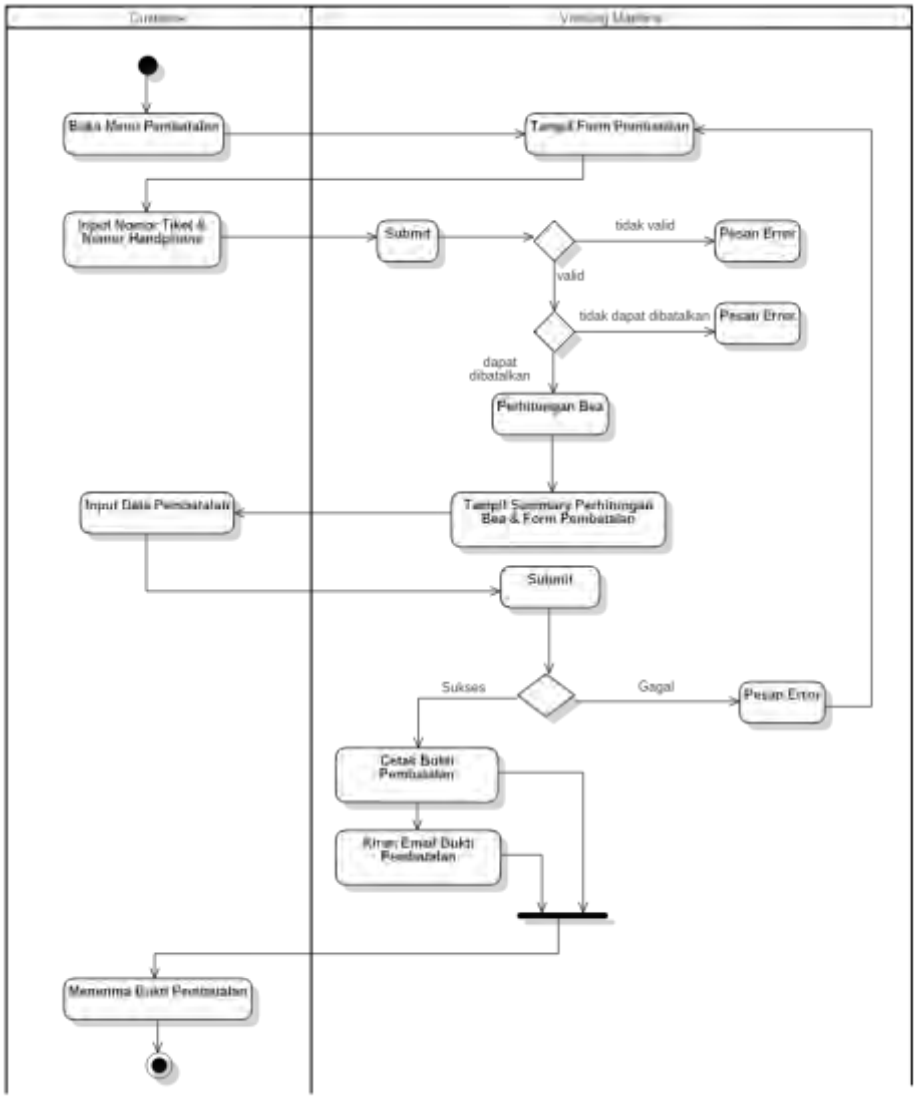
Aliran aktivitas dari proses *gate-out* penumpang di stasiun kedatangan dijelaskan sebagai berikut:

1) *Customer* menempelkan barcode tiket pada gate di stasiun kedatangan kereta api
2) Jika status stasiun tidak masuk dalam koridor stasiun pemberhentian, maka sistem menampilkan pesan error “bukan stasiun kedatangan”
3) Jika status stasiun masuk dalam koridor stasiun pemberhentian, sistem akan membuka gate di stasiun kedatangan.
4) *Customer* dapat keluar dari stasiun kedatangan

5. Diagram *Activity* Proses Pembatalan

   a. Diagram *Activity* Pembatalan melalui *Vending Machine*

      Proses pembatalan tiket hanya dapat dilakukan pada *vending machine* yang terdapat di stasiun-stasiun yang telah ditunjuk sebagai stasiun pembatalan. Proses pembatalan dapat dilakukan sebelum ssatuan waktu parameter yang telah ditentukan sebelum waktu keberangkatan kereta api. Diagram *Activity* dari proses pembatalan tiket melalui *vending machine* dapat dilihat pada gambar 4.18



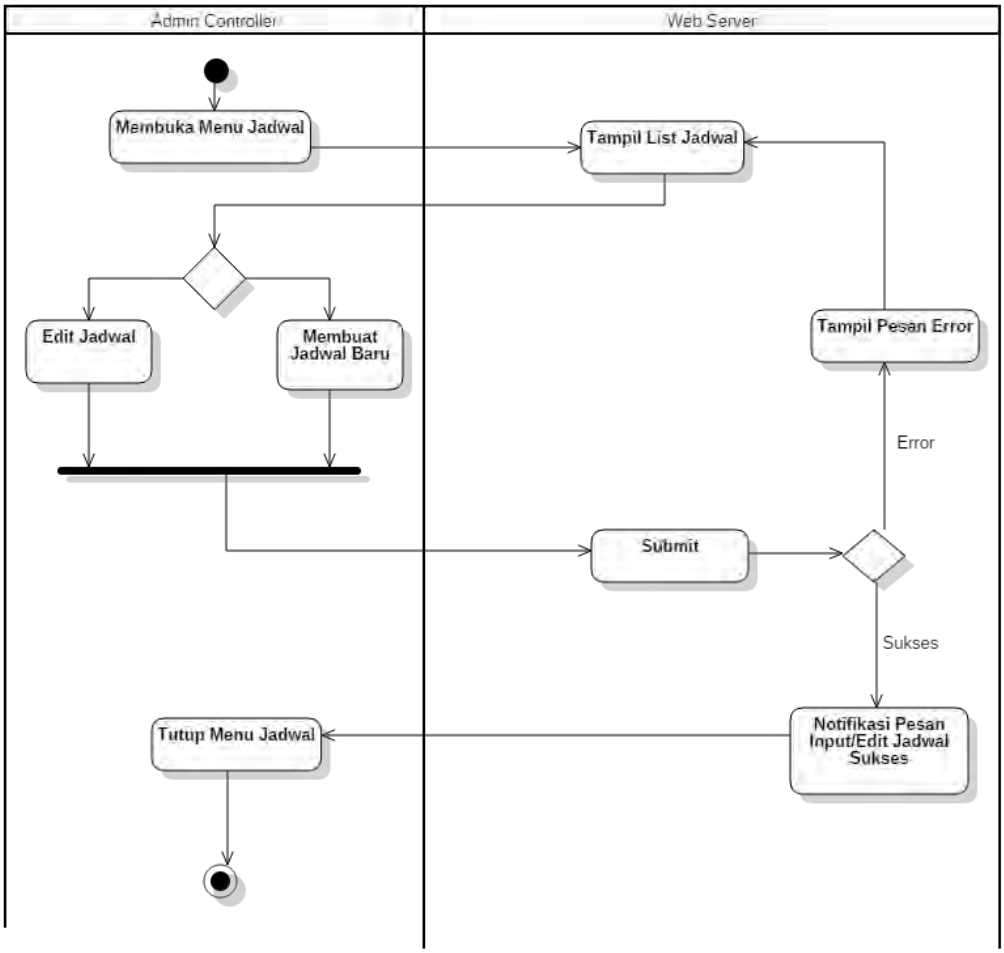
Gambar 4.18
Diagram *Activity* Proses Pembatalan melalui Vending Machine

Aliran aktivitas dari proses pembatalan tiket dijelaskan sebagai berikut:

1) *Customer* menempelkan barcode tiket pada gate di stasiun kedatangan kereta api
2) Jika status stasiun tidak masuk dalam koridor stasiun pemberhentian, maka sistem menampilkan pesan error “bukan stasiun kedatangan”
3) Jika status stasiun masuk dalam koridor stasiun pemberhentian, sistem akan membuka gate di stasiun kedatangan.
4) *Customer* keluar dari stasiun kedatangan.

6. Diagram *Activity* Penjadwalan

Penjadwalan dilakukan oleh seorang administrator melalui aplikasi *web* (*back-end* administrator). Penjadwalan diperlukan untuk memberikan data kereta api beserta perjalanan yang akan dijalankan termasuk di dalamnya proses pengalokasian tempat duduk yang akan dijual pada masing-masing gerbong dan pendefinisian subklas-subklas dari masing-masing tempat duduk. Diagram *Activity* dari proses penjadwalan tiket melalui *vending machine* dapat dilihat pada gambar 4.19



Gambar 4.19
Diagram *Activity* Proses Penjadwalan

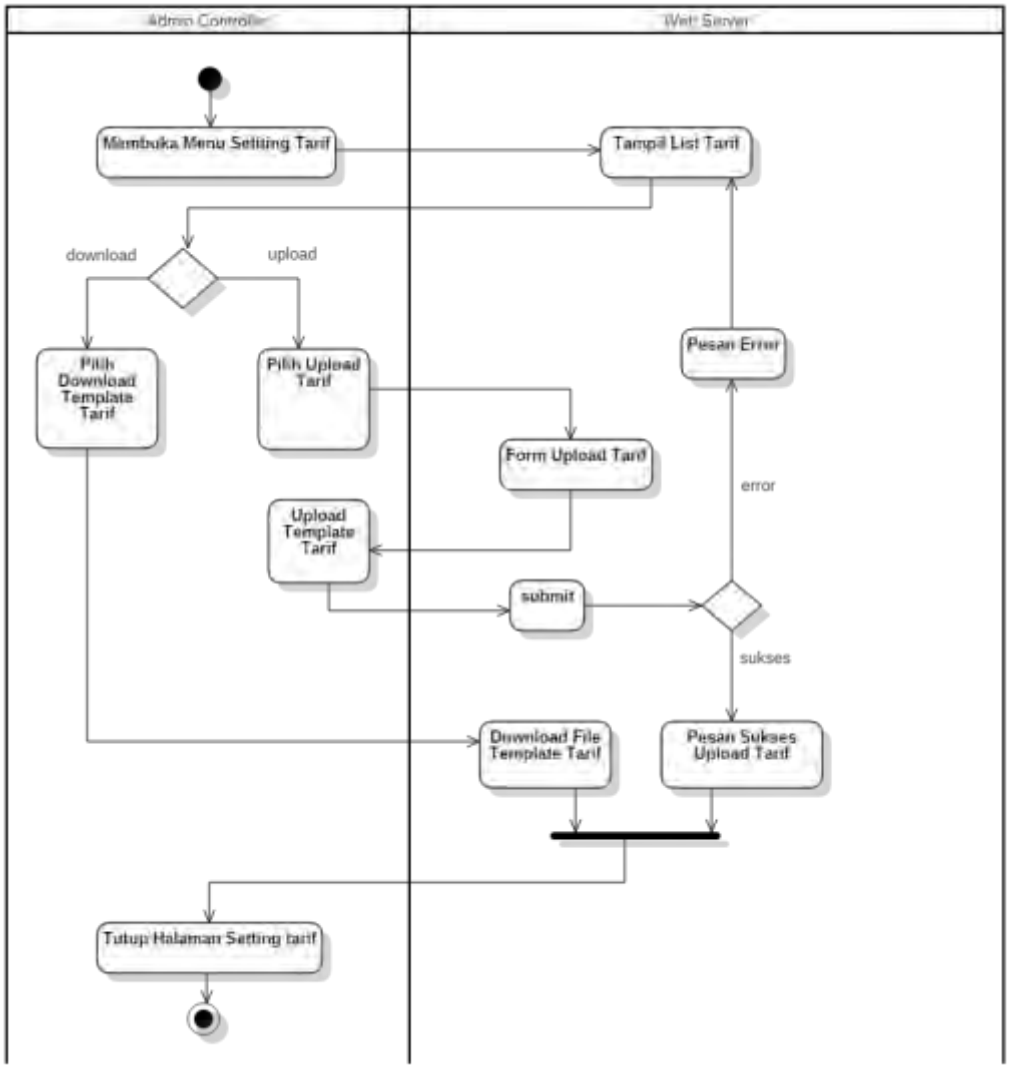
Aliran aktivitas dari proses penjadwalan oleh administrator *web* dijelaskan sebagai berikut:

1) Admin membuka menu jadwal
2) Sistem menampilkan *list* jadwal yang sudah dibuat
3) Jika admin memilih menu pembuatan jadwal baru, sistem menampilkan panduan form pembuatan jadwal baru.
4) Jika admin memilih menu perubahan jadwal, sistem akan menampilkan form edit jadwal
5) Sistem melakukan *submit* data jadwal

6) Jika submit data jadwal gagal, sistem menampilkan pesan error. Jika submit data jadwal sukses, sistem menampilkan notifikasi pesan input/edit jadwal sukses.

7. Diagram *Activity Setting* Tarif

Proses setting tarif dilakukan apabila jadwal sudah tersedia melalui proses pembuatan jadwal kereta api. Tarif yang diterapkan disesuaikan berdasarkan nomor kereta, tanggal keberangkatan, jenis penumpang, dan subklas gerbongnya. Diagram *Activity* dari proses setting tarif dijelaskan pada gambar 4.20



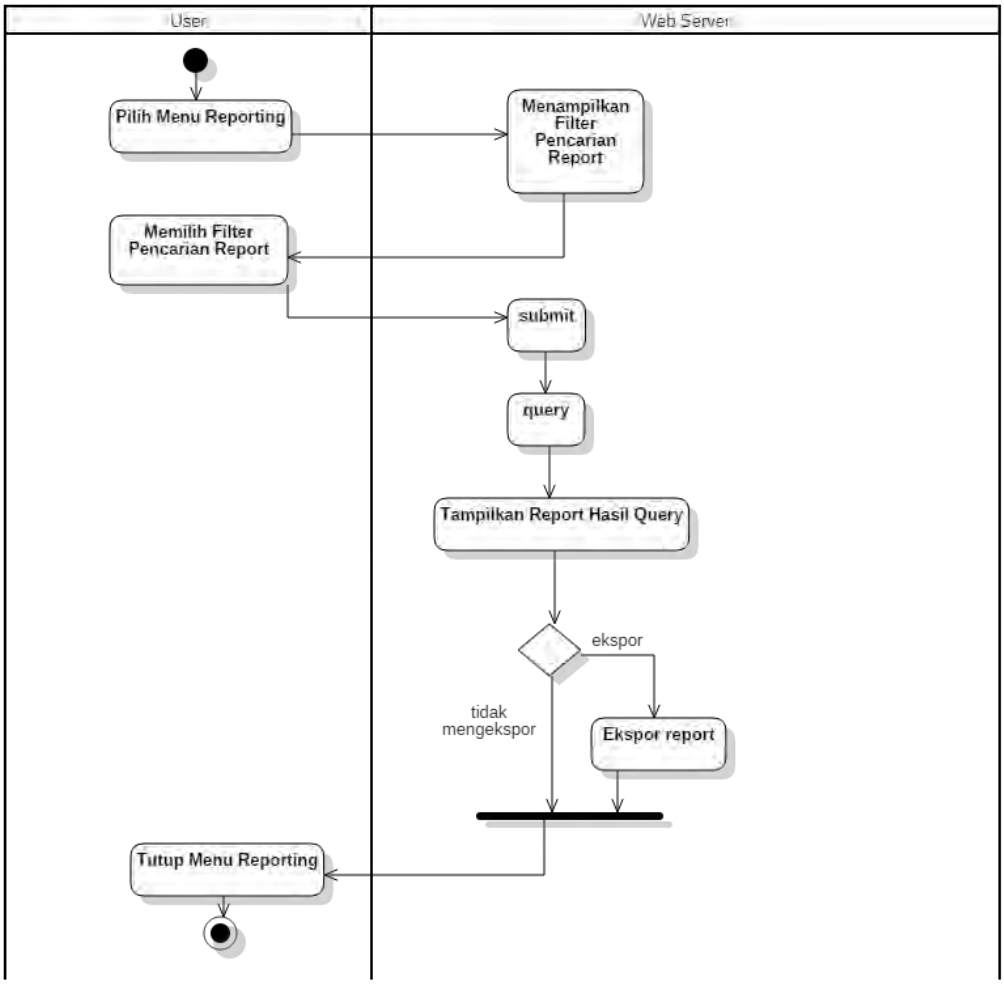
Gambar 4.20
Diagram *Activity* Proses Setting Tarif

Aliran aktivitas dari proses setting tarif oleh administrator *web* dijelaskan sebagai berikut:

1) Admin membuka menu setting tarif
2) Sistem menampilkan *list* tarif yang sudah dibuat
3) Jika admin memilih menu download template tarif, sistem akan mendownload file template tarif.
4) Jika admin memilih menu upload tarif, sistem akan menampilkan form upload file template tarif.
5) Admin melakukan *upload* file tarif
6) Jika upload file tarif gagal, sistem menampilkan pesan error. Jika upload file tarif sukses, sistem menampilkan notifikasi pesan upload tarif sukses.

8. Diagram *Activity Web Reporting*

User internal perusahaan melakukan proses untuk menghasilkan *report* yang dilakukan dari hasil *query* pengolahan data disesuaikan dengan kebutuhan terutama adanya *report* rekapitulasi dari hasil penjualan tiket (pemesanan dan pembayaran tiket). Diagram *Activity* dari proses *web reporting* dijelaskan pada gambar 4.21



Gambar 4.21
Diagram *Activity* Proses Web Reporting

Aliran aktivitas dari proses *reporting* tarif oleh administrator *web* dijelaskan sebagai berikut:

1) User memilih menu *reporting*
2) Sistem menampilkan filter pencarian *report*
3) User memilih filter pencarian *report*
4) Sistem melakukan submit dan *query* berdasarkan filter pencarian *report*
5) Data *report* hasil *query* ditampilkan
6) Jika user memilih menu *export*, sistem menghasilkan *file report* sesuai format yang dipilih.

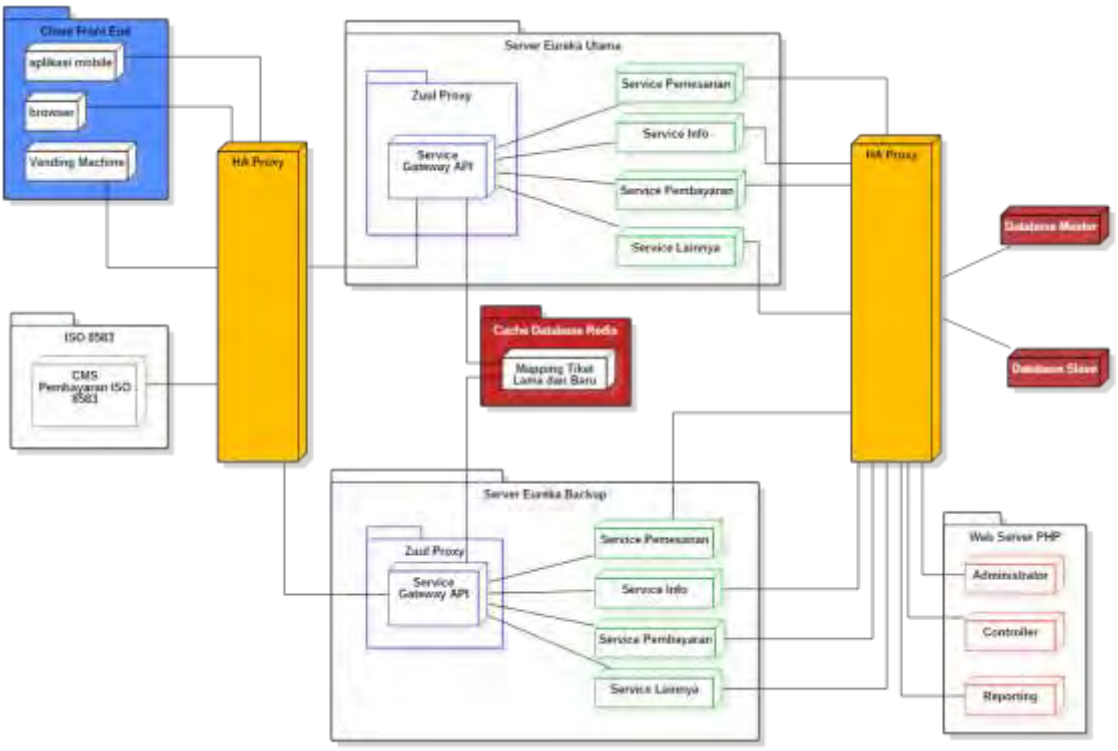
4.2.2 *Service* Bisnis *To-be*

Berdasarkan definisi *use case*, kebutuhan *service-service* dikekompokkan ke dalam kategori *service core*, *service non-core*, dan *service* eksternal. *Service* core merupakan *service* utama atau penting dari sebuah *service* bisnis, dimana bisnis mengembangkan atau mengoperasikan aktivitas utamanya. *Service* non-core merupakan kumpulan dari *service* penunjang *service* core. Dan kelompok *service* eksternal kaitanya dengan kebutuhan tambahan dan hubungan *service* dengan lingkungan eksternal.

Service core yang dikembangkan pada sistem *e-ticketing* kereta api *to-be* yaitu : *Service* Schedule, *Service* Transaction, *Service* Payment, dan *Service* Gate. *Service* non-core terdiri dari *Service* Info dan *Service* E-mail. Sedangkan *service* eksternal yaitu *service* payment-gateway.

4.2.3 Perancangan Arsitektur Aplikasi *To-be*

Perancangan arsitektur sistem *e-ticketing* kereta api *to-be* secara lojik dijelaskan dengan diagram *deployment* pada gambar 4.22 sebagai berikut



Gambar 4.22
Diagram *Deployment* Sistem *E-ticketing* Kereta Api

Aplikasi client terdiri dari aplikasi mobile, *web* browser dan vending machine. Client mengakses *service* melalui sebuah load balancer traffic jaringan yang dinamakan HA Proxy. Selain adanya load balancing jaringan, terdapat pula *service* gateway API yang dibungkus dalam komponen Zuul proxy sebagai load balancing sistem. Komponen microservice yang telah dibuat paketnua, dideploy dan diregistrasi ke dalam server eureka. Masing-masing *service* dideploy tersendiri dan secara otonom tidak mempengaruhi *service* lain dalam hal terjadi gangguan.

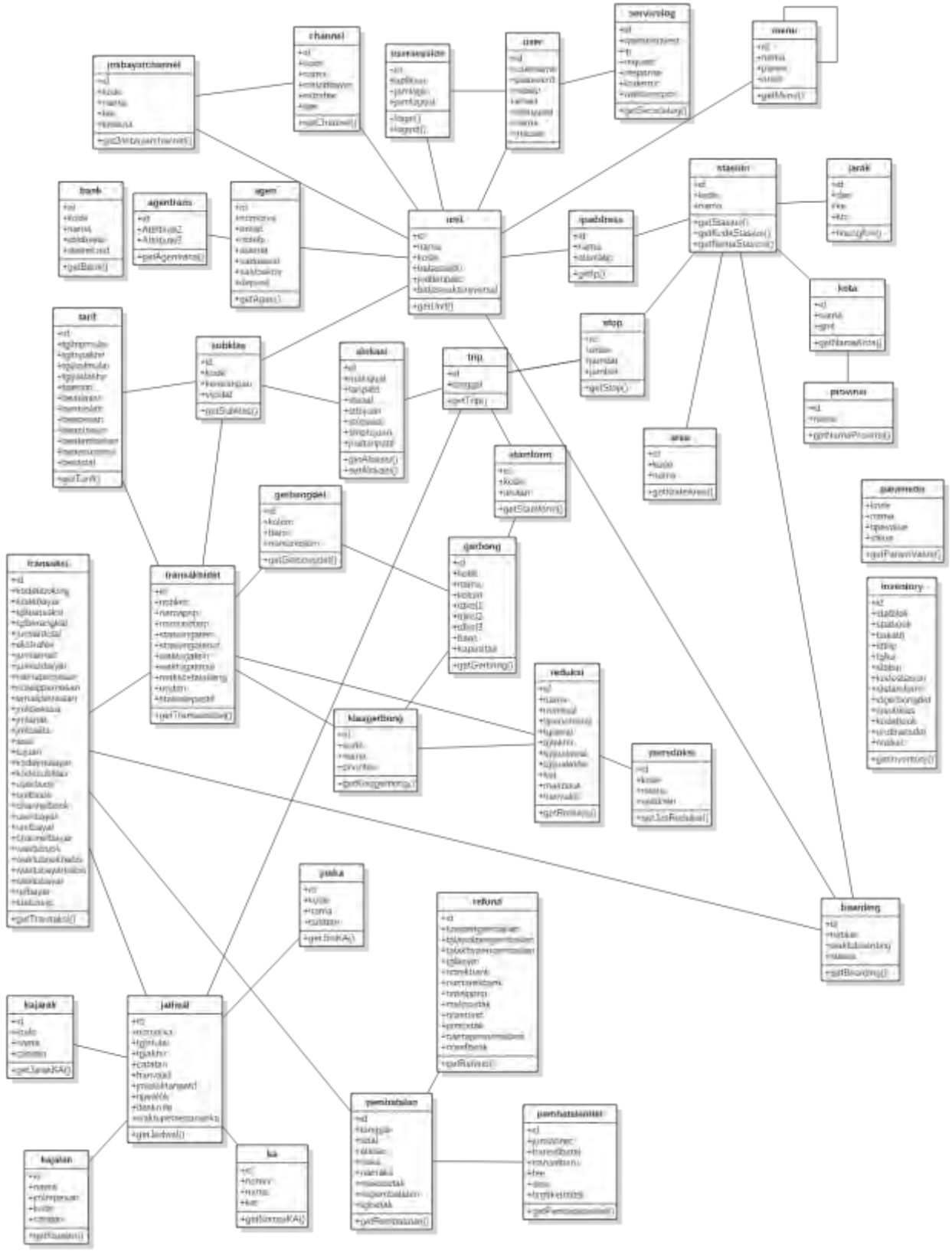
Server eureka yang dirancang dibuat dua server, dimana server pertama sebagai server utama dan server kedua sebagai server backup untuk keperluan *deployment service* ketika sistem sudah pada tahap *production*. Sebuah teknologi opensource redis juga digunakan untuk penempatan struktur data dalam memori dan juga sebagai caching data yang sering diakses sehingga respon pengolahan data yang dipanggil oleh suatu *service* menjadi lebih cepat.

Database server direplikasi secara *master-slave*, di antara pengaksesan *service* terhadap koneksi database terdapat load balancer HA Proxy. Replikasi database memungkinkan adanya salinan database secara real time satu arah dari database master ke slave dengan tujuan melakukan backup dari database master.

Aplikasi *web* administrator *back-end* digunakan sebagai penunjang data master jadwal kereta, setting tarif dan setting konfigurasi parameter yang digunakan dalam operasional penjualan tiket.

4.2.3.1 Diagram Class

Diagram Class menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem *e-ticketing* kereta api. Pada Gambar 4.24 Didefinisikan kelas-kelas dan relasinya untuk sistem *e-ticketing* kereta api.



Gambar 4.23
Diagram Class Sistem *E-ticketing* Kereta Api

Keterangan diagram *class* dapat dilihat pada table 4.7

Tabel 4.7 Deskripsi Diagram *Class*

No	Nama *Class*	Atribut	Tipe	Keterangan
1	jnsbayarchannel	id kode nama fee timeout	int string string real time	Merupakan kelas jenis pembayaran yang dilakukan *customer* dalam pemesanan tiket
2	channel	id kode nama methodbayar extrude tipe	int string string int string string	Merupakan pilihan bayar *customer*
3	usersession	id aplikasi jamlogin jamlogout	int int time time	Merupakan Waktu user dalam mengakses aplikasi
4	user	id username password no_telp email idrequest nama jnsuser	int string string int string int string int	Merupakan Data user atau pemesan ketika akan memsan tiket
5	*service* log	id wakturequest ip request response kodeerror wakturesponse	int date/time string string string string date/time	Merupakan Keterangan histori pengguna aplikasi
6	menu	id nama parent order	int string int int	Merupakan kelas menu sebagai pilihan pilihan yang diakses pengguna aplikasi
7	bank	id kode nama stsbayar	int string string int	Merupakan bank yang dipilih perusahaan sebagai mitra pembayaran user
8	agentrans	id atribute2 atribute3	int string string	Merupakan transaksi yang dilakukan agen dalam distribusi tiket
9	agen	id nomorva email notelp alamat saldoawal saldoakhir deposit	int string string string string float float float	Merupakan orang ketiga dalam penyaluran tiket
10	unit	id nama batassaldo jualtanpatd	int string float string	Merupakan organisasi

No	Nama *Class*	Atribut	Tipe	Keterangan
		bataswaktuinterval	date/time	
11	ipadress	id nama alamatip	int string string	Merupakan alamat IP yang digunakan oleh pengguna dalam mengakses aplikasi
12	stasiun	id kode nama	int string string	Merupakan stasuin yang dipilih pengguna aplikasi
13	jarak	id dari ke km	int string string int	Merupakan jarak perjalanan yang dipesan pengguna aplikasi
14	tarif	id tgltripmulai tgltripakhir tgljualmulai tgljualakhir tipepnp beadasar beapesanan beatuslah beastasuin beatambahan beaasuransu beatotal	int date date date date string string string string string string string string string	Merupakan harga yang harus dikeluarkan pemesan tiket
15	subclass	id kode keterangan vivstat	int string string string	Merupakan kelas yang dipilih pemesan tiket
16	alokasi	id maksjual tanpatd sttujuan stasal stopasal stoptujuan julatanpatd	int string string string string string string string	Merupakan transaksi pemesanan tiket oleh pemesan
17	trip	id tanggal	int date	Merupakan pembelina tiket oleh pemesan
18	stop	id order jamdat jamber	int string date date	Merupakan diberhentikan pemesanan tiket oleh system apabila waktu telah lewat
19	kota	id nama gmt	int string string	Merupakan Kota tujuan pemesan tiket
20	transaksi	id kodeboking kodebayar tgltransaksi tglbayar jumlahtotal ekstrafee jmlnet jmlbayar	int int int date date int string int float	Merupakan transaksi pemesanan, update status pembayaran, dan pembatalan

No	Nama *Class*	Atribut	Tipe	Keterangan
		namapemesanan notelppemesanan emailpemesanan jumlahdewasa jumlahanak jumlahbalita asal tujuan kodejnsbayar kodesubclas userbook unitbook channelbook userbayar unitbayar channelbayar waktubook waktubookhabis waktubayarhabis waktubayar refbayar statusviv	string string string int int int string string int int string string string float float string date/time date/time date/time string string	
21	transaksidet	id notiket namapnp nomoridpnp stasiungatein stasiungateout waktugatein waktugateout makscetakulang urutan statuslepasstd	int string string string string string date/time date/time int int string	Merupakan rekap dari transaksi pemesanan tiket
22	gerbong	ide kode nama kolom tdkol1 tdkol2 tdkol3 baris kapasitas	int string string string string string string string	Merupakan gerbong kereta yang dipilih oleh pemesan tiket
23	kelasgerbong	id kode nama prioritas	int string string string	Merupakan kelas gerbong sesuai dengan pilihan pemesan tiket
24	gerbongdet	id kolom baris nomorkolom	int string string int	Merupakan tempat duduk pemesan pada gerbong yang dipilih
25	stamform	id kode urutan	int string string	Urutan tempat duduk pemesan tiket
26	area	id kode nama	int string string	Merupakan area stasiun yang dipilh oleh pemesan tiket

No	Nama *Class*	Atribut	Tipe	Keterangan
27	kota	id nama gmt	int string string	Merupakan kota yang dipilih oleh pemesan tiket
28	provinsi	id nama	int string	Merupakan provinsi yang dipilih oleh pemesan tiket sesuai dengan jadwal kereta
29	parameter	kode nama tipevalue value	string string int int	Merupakan parameter atau acuan pemesanan tiket
30	reduksi	id nama nominal typenominal tglawal tglakhir tgljualawal tgljualkahir ket maxbook hari valid	int string string string date date date date string string string	Merupakan keterangan dari pemesanan tiket
31	jnsreduksi	id kode nama validhari	int string string date	Merupakan hari yang dipilih pemesan tiket
32	inventori	id statblok statbook bukatd idtrip tglka idstop kodestasiun idstamform idgerbongdet idsubklas kodebook uruttransdes notiket	int string string string int date int string int int int string string string	Merupakan kelengkapan pemesanan tiket oleh pemesan tiket
33	kajarak	id kode nama catatan	int string string string	Merupakan jarak tempuh kereta sesuai dengan pemesanan tiket
34	kajalan	id nama jmlpesan kode catatan	int string int string string	Merupakan keterangan perjalan kereta api pemberangkatan maupun kembli
35	jadwal	id nomorka tglmulai tglakhir catatan harivalid	int string date date string date	Merupakan jadwal kereta api

No	Nama *Class*	Atribut	Tipe	Keterangan
		jumlahaloktanptd tipealok deskrute waktupemesananka	int int string date	
36	ka	id nomor nama keterangan	int string string string	Merupakan kereta api yang dipilih oleh pemesan
37	jnska	id kode nama catatan	int string string string	Merupakan jenis kereta yang dipilih oleh pemesan tiket
38	pembatalan	id tanggal total alasan noka namaka makscetak nopembatalan tglcetak	int date int string string string string string date	Merupakan pembatalan pemesanan oleh pemesasan tiket
39	pembatalandet	id jumlahnet trransidbatal transidbaru fee sisa hrgtiketbatal	int int string string float string float	Merupakan keterangan pembatalan pemesanan tiket
40	refund	id tipepengembalian tglpengembalian tglakhirpengembalian tglbayar norekbank namarekbank notelppnp makscetak norefund jamcetak namapenerimabea norefbank	int string date date date int string string string string date/time string string	Merupakan pengembalian biaya kembali pada pemesan tiket apabila pemesanan tiket dibatalakan
41	boarding	id notiket waktuboarding status	id string date/time string	Merupakan proses penumpang masuk kadalam stasiun kereta api melalui peron yang dijaga petugas

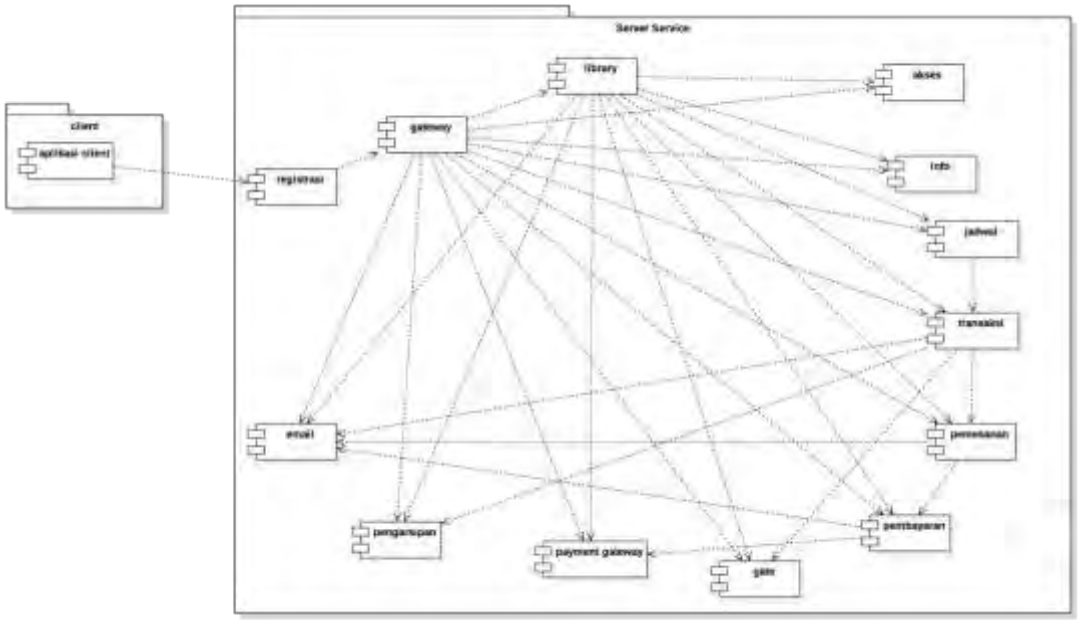
4.2.3.2 Perancangan *Service* Aplikasi *To-be*

Perancangan aplikasi *to-be* akan menjelaskan *service orchestration*, *service-service* yang mendukung fungsi bisnis *to-be* yang dikelompokkan ke dalam *service core*, *non-core* dan eksternal.

4.2.3.2.1 *Service Orchestration*

Serupa dengan sebuah alur kerja organisasi, *service orchestration* adalah koordinasi dan pengaturan berbagai *service* yang digambarkan sebagai satu *service* tunggal secara agregat. Pengembang memanfaatkan *service orchestration* untuk mendukung otomatisasi proses bisnis dengan menggabungkan *service* di berbagai aplikasi sehingga menciptakan aplikasi komposit berikutnya. Dengan kata lain, *service orchestration* adalah kombinasi dari interaksi *service* untuk menciptakan layanan bisnis tingkat yang lebih tinggi.

Service Orchestration dari Sistem *E-ticketing* Kereta Api dapat digambarkan pada Gambar 4.24:



Gambar 4.24
Diagram Component Service Orchestration

Service registry akan meregestrasi kumpulan *service-service* yang dijalankan pada aplikasi. Setiap *service* yang diakses oleh *client* dikelola melalui *service* gateway yang bertugas untuk *load balancer service* aplikasi. Semua library class database dari *service*

dikumpulkan ke dalam *service library* yang mendefinisikan semua struktur data dan tabel-tabel dari *database*, *service library* ini diakses oleh semua *service* yang memerlukan. Untuk keamanan *service* dikelola oleh sebuah *service* akses yang akan melakukan *filtering* akses *service* dari pengaksesan secara ilegal. Pengaksesan *service* dipastikan harus telah meregisterkan sebuah *IP Address* dan *Requester ID* sebagai identifikasi sumber pengaksesan yang sah. Sebuah alamat *web service* ditentukan dalam format REST (protokol HTTP atau HTTPS) yang dapat dibuat dalam sebuah nama *domain* maupun IP Address server *service*. Data yang dihasilkan oleh *service* berupa respon dalam format JSON (http://www.json.org).

Penggunaan parameter-parameter yang digunakan dalam *request service* bersifat *case-sensitive.* Parameter-parameter berikut ini wajib dikirimkan di setiap *web service* yang ada:

Tabel 4.8 Parameter *request service mandatory*

Field	Data Type	Mandatory	Description
Rqid	String	YES	Requester ID
App	String	YES	Module yang hendak diakses
Action	String	YES	Action yang akan dipanggil dari module terkait

Penjelasan dari *service-service* akan dijelaskan mengenai format *request*, parameter *input*, dan *response* dari setiap *service*.

4.2.3.2.2 *Service Core* Aplikasi *To-be*

Service Core sistem *e-ticketing* kereta api merupakan *service* utama yaitu: *Service* Schedule, *Service* Transaction, *Service* Payment, dan *Service* Gate.

1. *Service* Schedule

   Service yang digunakan untuk mendapatkan jadwal keberangkatan

   Request

Method	URL	Sample
GET	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/{org}/{dest}/{dep_date}

Parameter In

Type	Params	Data Type	Mandatory	Description
HEAD	auth_key	String	Yes	Key Primary
GET	app	String	Yes	Information
GET	action	String	Yes	Nama *service*
GET	org	String	Yes	Kode Origination
GET	des	String	Yes	Kode Destination
GET	dep_date	Date	Yes	Tanggal Keberangkatan

Format Parameter In

Type	Params	Values
HEAD	auth_key	Unik ID
GET	app	“schedule”
GET	action	“getschedule”
GET	org	Jika ada valuenya maka submit valuenya. {org} = nama stasiun asal Jika tidak ada valuenya makan submit kosong. {org} = “”
GET	des	Jika ada valuenya maka submit valuenya. {des} = nama stasiun asal Jika tidak ada valuenya makan submit kosong. {des} = “”
GET	dep_date	Format tanggal = “DD/MM/YYYY”

Response

Status	Response

200	Response berupa list array objek yang berisi list jadwal keberangkatan berdasarkan data terkait yang didapat dari request.
	Hasil:
	
	{
	"response": {
	"availabilitydatalist": [{
	"id": "PRO003",
	"tripdate": "2017-01-01",
	"stasiunnameorg": "MANGGARAI",
	"stasiuncodeorg": "MRI",
	"stasiuncodedes": "BPR",
	"scheduleDatas": [{
	"id": "SCH-A001-001",
	"noka": "A001",
	"trainname": "AIRPORT RAILINK SERVICES",
	"stopdeparture": "0500",
	"stoparrival": "0540",
	"allocationDatas": [{

Status	Response

	"subclasscode": "A",
	"seatavailable": 100,
	"faretotamount": 100000
	}, {
	"subclasscode": "J",
	"seatavailable": 100,
	"faretotamount": 90000
	}]
	}]
	}, {
	"id": "PRO004",
	"tripdate": "2017-01-01",
	"stasiunnameorg": "MANGGARAI",
	"stasiuncodeorg": "MRI",
	"stasiuncodedes": "BSH",
	"scheduleDatas": [{
	"id": "SCH-A001-001",
	"noka": "A001",
	"trainname": "AIRPORT RAILINK SERVICES",
	"stopdeparture": "0500",
	"stoparrival": "0555",
	"allocationDatas": [{
	"subclasscode": "A",
	"seatavailable": 100,
	"faretotamount": 100000
	}, {
	"subclasscode": "J",
	"seatavailable": 100,
	"faretotamount": 90000
	}]
	}]
	}, {
	"id": "PRO001",
	"tripdate": "2017-01-01",
	"stasiunnameorg": "MANGGARAI",
	"stasiuncodeorg": "MRI",
	"stasiuncodedes": "SDB",
	"scheduleDatas": [{
	"id": "SCH-A001-001",
	"noka": "A001",
	"trainname": "AIRPORT RAILINK SERVICES",
	"stopdeparture": "0500",
	"stoparrival": "0510",
	"allocationDatas": [{
	"subclasscode": "A",
	"seatavailable": 100,
	"faretotamount": 100000
	}, {
	"subclasscode": "J",
	"seatavailable": 100,
	"faretotamount": 90000
	}]
	}]

Status	Response

	}, {
	"id": "PRO002",
	"tripdate": "2017-01-01",
	"stasiunnameorg": "MANGGARAI",
	"stasiuncodeorg": "MRI",
	"stasiuncodedes": "DU",
	"scheduleDatas": [{
	"id": "SCH-A001-001",
	"noka": "A001",
	"trainname": "AIRPORT RAILINK SERVICES",
	"stopdeparture": "0500",
	"stoparrival": "0525",
	"allocationDatas": [{
	"subclasscode": "A",
	"seatavailable": 100,
	"faretotamount": 100000
	}, {
	"subclasscode": "J",
	"seatavailable": 100,
	"faretotamount": 90000
	}]
	}]
	}]
	},
	"status": "00",
	"message": "SUCCESS"
	}

400	{"error":"Please specify database version."}
400	{"error":"Invalid database version."}
401	{"error":"Invalid API key."}
500	{"error":"Something went wrong. Please try again later."}

2. *Service* Transaction

Service yang digunakan untuk melakukan insert transaksi.

Request

Method	URL	Sample
POST	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/

Parameter In

Type	Params	Data Type	Mandatory	Description
HEAD	auth_key	String	Yes	Key Primary
POST	app	String	Yes	Information
POST	action	String	Yes	Nama *service*
POST	c_id	String	Yes	ID schedule yang dipilih
POST	nama	String	Yes	Nama penumpang
POST	alamat	String	Yes	Alamat penumpang
POST	notlp	String	Yes	No Telpon Penumpang
POST	c_inventory_id	String	Yes	List inventory Id

Format Parameter In

Type	Params	Values
HEAD	auth_key	Unik ID
POST	app	“transaction”
POST	action	“booking”
POST	c_id	Diisi oleh c_id yang sebelumnya didapat dari *service* “get_schedule”
POST	nama	Diisi nama calon penumpang dari form registrasi
POST	alamat	Diisi alamat calon penumpang dari form registrasi
POST	notlp	Diisi no telpon calon penumpang – hanya numerik
POST	c_inventory_id	Diisi oleh list c_inventory_id yang didapat dari *service* “set_schedule_transaction”

Response

Status	Response

200	**Response berupa Pesan sukses booking kursi.**
	{
	"err_code": "0",
	"c": "BOOK SEAT SUCCESS",
	"c_message": "BOOK SEAT SUCCESS",
	"c_message": "BOOK SEAT SUCCESS",
	"c_message": "BOOK SEAT SUCCESS",
	
	}

400	{"error":"Please specify database version."}
400	{"error":"Invalid database version."}
401	{"error":"Invalid API key."}
500	{"error":"Something went wrong. Please try again later."}

3. *Service* Payment

Service yang digunakan untuk melakukan pembayaran. Proses pembayaran melakukan *update* status transaksi menjadi "PAID"

Request

Method	URL	Sample
POST	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/

Parameter In

Type	Params	Data Type	Mandatory	Description
URL	rqid	String	Yes	Key Primary
URL	app	String	Yes	Information
URL	action	String	Yes	Nama *service*
POST	bookcode	String	Optional	Bookcode
POST	paycode	String	Optional	Paymentcode
POST	nett_amount	Number	Yes	Nilai nominal yang dibayar
POST	paytype_code	String	Yes	Tipe pembayaran(cash, non-cash)

Format Parameter In

Type	Params	Values
URL	rqid	Unik ID
URL	app	"payment"
URL	action	"arts_payment"
POST	bookcode	Diisi oleh bookcode yang sebelumnya didapat dari *service* "Insert_Transaction" – Harus diisi jika "paycode kosong" – Harus 7 Character
POST	paycode	Diisi oleh paycode yang sebelumnya didapat dari *service* "Insert_Transaction" – Harus diisi jika "bookcode kosong" – Harus 13 Character
POST	nett_amount	Total Nominal Yang Dibayar
POST	Paytype_code	Jenis pembayaran

Response

Status	Response

200	**Response berupa Pesan sukses pembayaran.**
	Hasil:-
	{
	"err_code": "0",
	"c_message": "SUCCESS PAYMENT"
	}

Status	Response

202	{
	"err_code":"01","error_message":"RQID is not valid."
	}

202	{
	"err_code":"02","error_message":"Paycode or Bookcode must be filled."
	}

202	{
	"err_code":"03","error_message":"Paytype Code must be filled."
	}

202	{
	"err_code":"04","error_message":"Transaction not found."
	}

4. *Service* Gate

Service yang digunakan untuk melakukan operasi *gate-in* dan *gate-out*.

Request

Method	URL	Sample
POST	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/

Parameter In

Type	Params	Data Type	Mandatory	Description
HEAD	auth_key	String	Yes	Key Primary
POST	app	String	Yes	Information
POST	action	String	Yes	Nama *service*
POST	bookcode	String	Yes	Kode Booking
POST	ticketinum	String	Yes	Nomor Tiket
POST	tripdate	Date	Yes	Tanggal
POST	stasiunid	String	Yes	Keberangkatan
POST	unitid	String	Yes	ID Stasiun
POST	unitcode	String	Yes	ID Unit
POST	time	Time	Yes	Kode Unit Waktu *Gate-in*

Format Parameter In

Type	Params	Values
HEAD	auth_key	Unik ID
POST	app	“gate”
POST	action	“gatein”
POST	bookcode	{bookcode} = “kode booking”
POST	ticketnum	{ticketnum} = “nomor tiket”
POST	tripdate	{tripdate} = “tanggal keberangkatan”
POST	stasiunid	{stasiunid} = “ide stasiun”
POST	unitid	{unitid} = “id unit”
POST	unitcode	{unitcode} = “kode unit”
POST	time	Format Waktu = “YYYY-MM-DD hh:mm:ss”

Response

Status	Response

200	**Response berupa Pesan sukses *gate-in*.**
	Hasil:-
	{
	"err_code": "0",
	"c_message": "SUCCESS *GATE-IN*"
	}

202	{
	"err_code":"01","error_message":"RQID is not valid."
	}

202	{
	"err_code":"03”,”error_message”:”Already *gate-in*.”
	}

202	{
	"err_code":"04”,”error_message”:”Transaction not found.”
	}

4.2.3.2.3 *Service* Non-Core Aplikasi *To-be*

Service Non-core sistem *e-ticketing* kereta api merupakan *service* pendukung yaitu: *Service* Info, dan *Service* Email.

1. *Service* Info

   Service yang digunakan untuk menampilkan informasi-informasi yang diperlukan oleh *service* lainnya.

Request

Method	URL	Sample
GET	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/

Parameter In

Type	Params	Data Type	Mandatory	Description
HEAD GET GET	auth_key app action	String String String	Yes Yes Yes	Key Primary Information Nama *service*

Format Parameter In

Type	Params	Values
HEAD GET GET	auth_key app action	Unik ID “info” “getstasiun”

Response

Status	Response

200	Response berupa list array objek yang berisi list stasiun berdasarkan data terkait yang didapat dari request.
	Hasil:
	
	{
	"response": {
	"stasiunlist": [
	{
	"id": 1,
	"profitcenter": "0021032005",
	"name": "MANGGARAI",
	"code": "MRI",
	"pembatalan": "1",
	"pengembalianbea": "1",
	"status": "1",
	"domain": "1",
	"modifiedby": "50",
	"modifiedon": "2018-03-21 15:09:16",
	"areaid": "3",
	"cityid": "1",
	"createdby": "1",
	"createdon": "2016-10-06 15:19:17",
	"showpos": "1",
	"showibook": "1",
	"showmobile": "1",
	"startdate": "2016-11-16",
	"enddate": "9999-12-31",
	"goshowMin": 10,

Status	Response
	"goshowMax": 1440 }, { "id": 2, "profitcenter": "SDB0000001", "name": "SUDIRMAN BARU (BNI CITY)", "code": "SDB", "pembatalan": "1", "pengembalianbea": "1", "status": "1", "domain": "1", "modifiedby": "50", "modifiedon": "2018-03-21 15:09:18", "areaid": "3", "cityid": "1", "createdby": "1", "createdon": "2016-10-06 15:19:17", "showpos": "1", "showibook": "1", "showmobile": "1", "startdate": "2016-11-16", "enddate": "9999-12-31", "goshowMin": 10, "goshowMax": 1440 }, { "id": 3, "profitcenter": "0021034004", "name": "DURI", "code": "DU", "pembatalan": "0", "pengembalianbea": "0", "status": "1", "domain": "1", "modifiedby": "29", "modifiedon": "2018-03-21 15:09:22", "areaid": "3", "cityid": "1", "createdby": "1", "createdon": "2016-10-06 15:17:18", "showpos": "1", "showibook": "1", "showmobile": "1", "startdate": "2016-11-16", "enddate": "9999-12-31", "goshowMin": 10, "goshowMax": 1440 }, { "id": 4, "profitcenter": "BPR0000001", "name": "BATU CEPER", "code": "BPR", "pembatalan": "1", "pengembalianbea": "1",

Status	Response
	"status": "1", "domain": "1", "modifiedby": "3", "modifiedon": "2018-03-21 15:09:24", "areaid": "3", "cityid": "1", "createdby": "1", "createdon": "2016-10-06 15:19:17", "showpos": "1", "showibook": "1", "showmobile": "1", "startdate": "2016-11-16", "enddate": "9999-12-31", "goshowMin": 10, "goshowMax": 1440 }, { "id": 5, "profitcenter": "BST0000001", "name": "SHIA", "code": "BST", "pembatalan": "1", "pengembalianbea": "1", "status": "1", "domain": "1", "modifiedby": "2", "modifiedon": "2018-03-21 15:09:26", "areaid": "3", "cityid": "1", "createdby": "2", "createdon": "2017-11-30 16:35:39", "showpos": "1", "showibook": "1", "showmobile": "1", "startdate": "2017-04-17", "enddate": "2027-12-31", "goshowMin": 10, "goshowMax": 1440 }, { "id": 11, "profitcenter": "BKS0000001", "name": "BEKASI", "code": "BKS", "pembatalan": "1", "pengembalianbea": "1", "status": "1", "domain": "1", "modifiedby": "9", "modifiedon": "2018-03-21 15:09:35", "areaid": "3", "cityid": "1", "createdby": "9", "createdon": "2018-03-01 18:04:58", "showpos": "1",

Status	Response
	"showibook": "1", "showmobile": "1", "startdate": "2018-03-01", "enddate": "2028-03-31", "goshowMin": 10, "goshowMax": 1440 }] }, "status": "00", "message": "SUCCESS" }
400	{"error":"Please specify database version."}
400	{"error":"Invalid database version."}
401	{"error":"Invalid API key."}
500	{"error":"Something went wrong. Please try again later."}

2. *Service* Email

Service yang digunakan untuk mengirimkan email bukti pemesanan dan pembayaran

Request

Method	URL	Sample
POST	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/

Parameter In

Type	Params	Data Type	Mandatory	Description
HEAD	auth_key	String	Yes	Key Primary
POST	app	String	Yes	Information
POST	action	String	Yes	Nama *service*
POST	emailAddress	String	Yes	Alamat email
POST	message	String	Yes	Message email
POST	subject	String	Yes	Subject email

Format Parameter In

Type	Params	Values
HEAD	auth_key	Unik ID
POST	app	“email”
POST	action	“send”
POST	emalAddress	{emailAddress} = “alamat email”
POST	message	{message} = “message email”
POST	subject	{subject} = “subject email”

Response

Status	Response

200	**Response berupa Pesan sukses emal terkirm.**
	Hasil:-
	{
	"err_code": "0",
	"c_message": "SUCCESS EMAIL SENT"
	}

202	{
	"err_code":"01","error_message":"RQID is not valid."
	}

202	{
	"err_code":"03”,”error_message”:”Email Failed to send.”
	}

4.2.3.2.4 *Service* Eksternal Aplikasi *To-be*

Service Eksternal sistem *e-ticketing* kereta api merupakan *service* tambahan dan yang berkaitan dengan lingkungan eksternal yaitu *Service* Payment-gateway. *Service* Payment-Gateway merupakan *service* yang digunakan untuk mengakses API payment gateway pihak ketiga. Definisi *request, parameter, request dan response* dari *service* payment-gateway dijelaskan sebagai berikut:

Request

Method	URL	Sample
POST	http://domain.url.*service*?	http://domain-url-*service*/{app}/{action}/{rqid}/

Parameter In

Type	Params	Data Type	Mandatory	Description
HEAD	auth_key	String	Yes	Key Primary
POST	app	String	Yes	Information
POST	action	String	Yes	Nama *service*
POST	paycode	String	Yes	Kode Pembayaran
POST	bookcode	String	Yes	Kode Booking
POST	paytypecode	String	Yes	Tipe Pembayaran

Format Parameter In

Type	Params	Values
HEAD	auth_key	Unik ID
POST	app	“payment-gateway”
POST	action	“pay”
POST	paycode	{paycode} = “kode pembayaran”
POST	bookcode	{bookcode} = “kode booking”
POST	paytypecode	{paytypecode} = “kode tipe pembayaran”

Response

Status	Response

200	**Response berupa Pesan sukses pembayaran.**
	Hasil:-
	{
	"err_code": "0",
	"c_message": "SUCCESS PAYMENT"
	}

202	{
	"err_code":"01","error_message":"RQID is not valid."
	}

202	{
	"err_code":"03”,”error_message”:”Paytype Code must be filled.”
	}

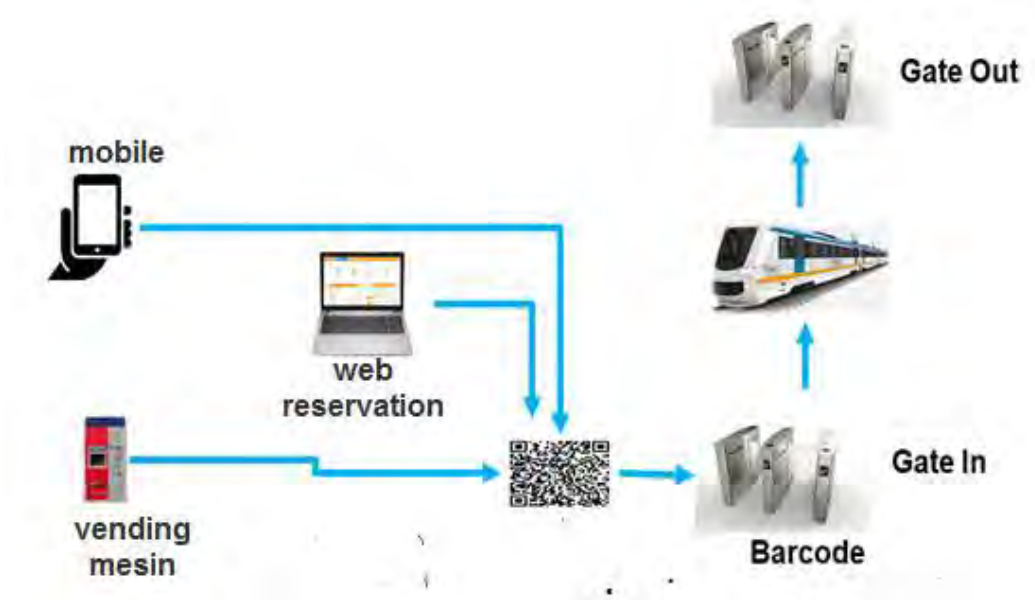
202	{
	"err_code":"04”,”error_message”:”Transaction not found.”
	}

4.2.4 Perancangan Arsitektur Infrastruktur *To-be*

Perancangan arsitektur infrastruktur *to-be* menjelaskan tentang topologi jaringan dan aspek kemanan yang akan diterapkan dalam sistem *e-ticketing* kereta api.

4.2.4.1 Topologi Jaringan

Blok diagram aliran secara global operasional darii Sistem *E-ticketing* Kereta Api *to-be* dapat diseperti pada gambar 4.25.



Gambar 4.25
Diagram alur global operasional sistem *e-ticketing* kereta api *to-be*

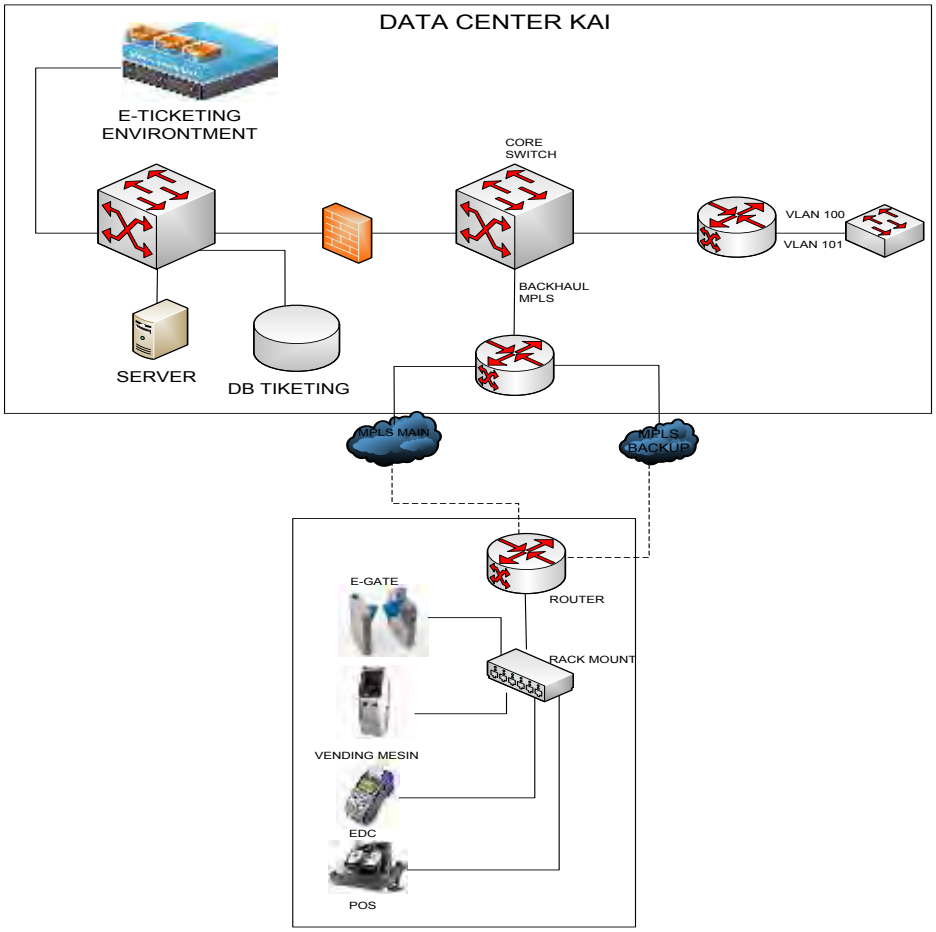
Dari gambar 4.25 dapat dilihat ada beberapa tahapan dalam mendapatkan tiket kereta api ada beberapa cara yaitu:

1. Pemesanan melalui aplikasi mobile

   Customer dapat melakukan pemesanan melalui handphone dengan aplikasi yang berbasis android atau IOS dan langsung mendapatkan nomor tiket sesuai dengan tgl pemesanan setelah melakukan pembayaran sehingga bisa memudahkan *customer* tanpa harus datang langsung ke stasiun. Pada saat *customer* sudah ke stasuin *customer* cukup menempelkan *barcode* pada mesin gate yang ada di stasiun pada saat pemberangkatan kereta maupun keluar dari stasiun kereta api
2. Pemesanan atau pembelian tiket langsung kereta api dilakukan dengan cara mendatangi langsung ke stasiun dan dilakukan melalui vending machine
3. Pemesanan melalui *web* reservasi, *customer* bisa langsung memesan melalui alamat *web* reservasi kereta api dan langsung membayar melalui e-commerce maupun

transfer sehingga ketika datang ke stasiun cukup menunjukan barcode tiket yang sudah dibeli pada mesin gate kereta api.

Topologi jaringan infrastruktur jaringan sistem *e-ticketing* dapat dilihat pada Gambar 4.26



Gambar 4.26
Topologi Jaringan Infrastruktur

Topologi jaringan infrastruktur digambarkan dengan memakai topologi jaringan komputer tree yang merupakan gabungan dari beberapa topologi star yang dihubungan dengan topologi bus, setiap topologi star akan terhubung ke topologi star lainnya menggunakan topologi bus, biasanya dalam topologi ini terdapat beberapa tingkatan jaringan, dan jaringan yang berada pada tingkat yang lebih tinggi dapat mengontrol jaringan yang berada pada tingkat yang lebih rendah.

4.2.4.2 Kemanan Jaringan

Dalam jaringan sistem *e-ticketing* kereta api yang melibatkan berbagai komponen *service* dan pihak-pihak yang berkepentingan, akan banyak terjadi hal yang dapat mengganggu kestabilan koneksi jaringan komputer tersebut, baik yang berkaitan dengan *hardware* (pengamanan fisik, sumber daya listrik) maupun yang berkaitan dengan *software* (sistem, konfigurasi, sistem akses, dan lain-lain). Gangguan pada sistem dapat terjadi karena faktor ketidaksengajaan yang dilakukan oleh pengelola (*human error*) maupun yang disebabkan oleh pihak ketiga. Gangguan dapat berupa perusakan, penyusupan, pencurian hak akses, penyalahgunaan data maupun sistem, sampai tindakan kriminal melalui sistem jaringan *e-ticketing* kereta api.

Aspek *security networking* yang dilakukan untuk pengemanan sistem *e-ticketing* KA Bandara, yaitu sebagai berikut:

1. Mengelompokkan terminal yang difungsikan sebagai pengendali jaringan atau titik pusat akses (*server*) pada suatu jaringan, yang selanjutnya diberikan pengamanan secara khusus.
2. Menyediakan pengamanan fisik berupa sebuah ruangan *Network Operating Center* (NOC) yang khusus untuk pengamanan perangkat yang disebut pada poin nomor 1. Ruangan tersebut membatasi personil yang diperbolehkan masuk.
3. Memisahkan sumber daya listrik untuk NOC dari pemakaian yang lain. Perlu juga difungsikan *Uninteruptable Power Supply* (UPS), *stabilizer*, dan genset untuk menjaga kestabilan *supply* listrik yang diperlukan perangkat pada NOC.
4. Merapikan *wiring* ruangan dan memberikan label serta pengklasifikasian kabel.
5. Memberikan *Network Security* berupa sistem *firewall* pada perangkat yang difungsikan di jaringan. Firewall adalah salah satu aplikasi pada sistem operasi yang dibutuhkan oleh jaringan komputer untuk melindungi intergritas data/sistem jaringan dari serangan-serangan pihak yang tidak bertanggung jawab. Caranya dengan melakukan *filter* terhadap paket-paket yang melewatinya.
6. Merencanakan *maintenance* dan menyiapkan *back-up* (pencadangan) sistem.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis perancangan arsitektur bisnis dan teknologi informasi *as-is*, perancangan arsitektur *enterprise* bisnis dan teknologi informasi *to-be* untuk sistem *e-ticketing* kereta api sesuai dengan kerangka SOEA, dapat disimpulkan sebagai berikut:

1. Perancangan arsitektur bisnis sistem *e-ticketing* dimulai dengan melakukan analisis terhadap proses bisnis *as-is* dengan mendefiniskan fungsi-fungsi bisnis pada sistem *e-ticketing* yang sedang berjalan. Definisi fungsi bisnis dianalisis *gap* yang menjadi kendala dalam pengembangan untuk menentukan perubahan fungsi bisnis dalam perancangan arsitektur bisnis *to-be*. Hasil *gap analysis* dari perancangan arsitektur bisnis tersebut adalah:
   a. Sistem pembayaran tiket elektronik kereta api secara non-tunai.
   b. Pelayanan penjualan tiket elektronik kereta api tidak menggunakan loket sebagai *channel* penjualannya.
   c. Perbaikan pada proses validasi penumpang kereta api pada proses *gate-in* dan *gate-out*.
2. Perancangan arsitektur teknologi informasi sistem *e-ticketing* kereta api *to-be* berbasis *service* menerapkan perancangan arsitektur aplikasi berbasis *microservice* dimana fungsi-fungsi bisnis dari perancangan aristektur bisnis, masing-masing dijadikan sebagai *service-service* kecil yang independen, resilien, dan dapat berintegrasi dengan *platform* yang berbeda dengan standar protokol API. Arsitektur sistem *e-ticketing* kereta api didukung dengan topologi infrastruktur jaringan yang memiliki kapabilitas tinggi untuk memenuhi kebutuhan sistem *e-ticketing* yang dapat beroperasi selama 24 jam non-stop dan *downtime server* yang minimal.

5.2 Saran

Beberapa hal yang perlu dipertimbangkan dalam perancangan arsitektur *enterprise e-ticketing* kereta api *to-be* berbasis *service* menggunakan kerangka SOEA:

1. Perancangan arsitektur yang memerlukan banyak layanan *microservice* akan meningkatkan kompleksitas sejalan dengan adanya interaksi antara komponen bisnis *service* yang semakin bertambah.
2. Penggunaan sumber daya minimal, respon *service* yang cepat untuk proses *deployment*, dan fleksibilitas terhadap adanya perubahan permintaan dapat dipertimbangkan dengan menggunakan teknologi kontainer seperti *Docker*.
3. Pada studi penelitian perancangan arsitektur *e-ticketing* kereta api berikutnya, fungsi bisnis dapat dimodifikasi dengan adanya proses manajemen keanggotaan, penjualan tiket *voucher*, dan penjualan tiket fleksibel jam keberangkatan, *Business-To-Business* (B2B) dan *Business-To-Consumer* (B2C).

DAFTAR PUSTAKA

Assauri, Sofyan., *Manajemen Pemasaran Dasar, Konsep, dan Strategi*. PT. Jakarta: Grafindopersada.2011

David, Fred. R. and Forest R., *Strategic Management Concept and Cases 14th Edition*. Pearson Education. New Jersey. USA. 2012.

Erl, Thomas, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR. New Jersey. USA. 2005.

Engels, G., Assman, M., *Service-Oriented Enterprise Architectures: Evolution of Concepts and Methods*. Department of Computer Science University of Paderborn. München, Germany. 2008.

Haki, M. Kazem, Forte, M. Wentland. *Service Oriented Enterprise Architecture Framework*, University of Lausanne, Switzerland. 2010

Heydt-Benjamin, T.S., Chae, H., Defend, B., Fu, K,. *Privacy for Public Transportation*, University of Massachusetts, Amherst, MA. USA. 2006

Karami, Mitra. *Factor Influencing Adopting Online Ticketing*. Lu lea University of Technology. Swedia. 2006

Kunal Mittal. *Service Oriented Unified Process (SOUP)*. [Online]. http://www.kunalmittal.com/html/soup.html. 2007 [Dikunjungi Nopember 2017]

Miri, Ima, *Microservices vs. SOA, https://dzone.com/articles/microservices-vs-soa-2. 2017.* [Dikunjungi Desember 2017]

Lucky. *XML Web Services* Aplikasi *Desktop*, *Internet*, dan *Handphone*. JASAKOM. Jakarta. 2008

Ng-Kruelle, Grace and Paul Antohony Swatman. *E-ticketing Strategy and Implementation in an Open Access System: The case of Deutsche Bahn*. reasearchgate.net. 2006 [Dikunjungi Nopermber 2017]

O.F., Iwuagwu. *Electronic Ticketing in Public Transportation Systems: the need for Standardization in Master Thesis University of Technology Eindhoven*. Eindhoven. 2009

Osvald, Gundars. *Definition of Enterprise Architecture-centric Models for the Systems Engineer*. TASC, Inc. 2001

Pearce, J. and Robinson, R. *Strategic Management.* McGraw-Hill/Irwin. New York. USA. 2007

Porter, Michael, E., *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, New York. 1998

Satzinger, J. W., Jackson, R. B., Burd, S. D. *System Analysis and Design in A Changing World. USA:* Cengage Learning. 2012

Ward, J., & Peppard, J., *Strategic Planning for Information Systems*. John Wiley & Sons Ltd. London. 2002

Wisdaningrum, Oktavima. *Analisis Rantai Nilai (Value Chain) Dalam Lingkungan Internal Perusahaan*. Jurnal Vol.1, No. 1. Banyuwangi, Fakultas Ekonomi Universitas 17 Agustus 1945, 2013.

Yvon, Parizeau. *Enterprise Architecture for Complex Government and The Challenge of Government On-Line in Canada, Riset Master.* Dalhousie University. Canada. 2002

_____.,*Contactless Ticketing for Mass Transit, STMicroelectronics,* Rousset, Perancis. 2003

_____., *SOA Alliance Overview*, http://soa.omg.org/Uploaded%20Docs/SOA-Info-Day/Kuhbock-SOA_Alliance.pdf. 2006 [Dikunjungi 17 Nopember 2017]

_____.,*Architecture Practice*, *http://www.riverton.com.* Riverton Corporation. 2008

_____.,*Microservices*, *https://martinfowler.com/articles/microservices.html*. 2017