# IMPLEMENTASI ARSITEKTUR *ENTERPRISE* POLA *VENDING MACHINE* PADA TEKNOLOGI *MICROSERVICES*


Disusun Oleh:

N a m a : Muhammad Iqbal

NIM : 17523092

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**2021**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# IMPLEMENTASI ARSITEKTUR *ENTERPRISE* POLA *VENDING MACHINE* PADA TEKNOLOGI *MICROSERVICES*

## TUGAS AKHIR

Disusun Oleh:


N a m a : Muhammad Iqbal

NIM : 17523092


Yogyakarta, 25 Agustus 2021

Pembimbing 1,

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

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# IMPLEMENTASI ARSITEKTUR *ENTERPRISE* POLA *VENDING MACHINE* PADA TEKNOLOGI *MICROSERVICES*

## TUGAS AKHIR

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

Yogyakarta, 25 Agustus 2021

Tim Penguji

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

**Anggota 1**

Kurniawan Dwi Irianto, S.T., M.Sc.

**Anggota 2**

Moh. Idris, S.Kom., M.Kom.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

## HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Teknik Informatika

NIM : 17523092

Tugas akhir dengan judul:

# IMPLEMENTASI ARSITEKTUR *ENTERPRISE* POLA *VENDING MACHINE* PADA TEKNOLOGI *MICROSERVICES*

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

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

Yogyakarta, 25 Agustus 2021


( Muhammad Iqbal )

# HALAMAN PERSEMBAHAN

***Al-hamdu lillahi rabbil 'alamin,*** segala puji dan syukur atas nikmat umur, iman, dan rezeki yang berlimpah yang telah Allah SWT berikan. Alhamdulillah dengan izin Allah SWT penelitian ini dapat diselesaikan dengan baik. Semoga apa yang menjadi tujuan dari penelitian ini dapat tersampaikan serta memberikan manfaat tidak hanya bagi penulis tetapi juga untuk yang membaca.

Tugas akhir ini penulis persembahkan untuk:

**Kedua orang tua**

Terima kasih kepada kedua orang tua yang selalu memberikan semangat serta doa untuk menyelesaikan tugas akhir.

**Dosen pembimbing dan seluruh dosen Informatika**

Terima kasih kepada dosen pembimbing saya, bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., dan bapak Hanson Prihantoro Putro, S.T., M.T. serta seluruh dosen informatika yang telah memberikan ilmu, waktu, dan juga sabar dalam membimbing saya.

**Teman-teman**

Terima kasih kepada teman-teman yang juga memberikan semangat dan bantuan kepada saya.

**Seluruh pihak yang terlibat**

Terima kasih juga kepada seluruh pihak yang terlibat dalam pembuatan laporan tugas akhir saya atas dukungannya.

## HALAMAN MOTO

*"Jika kamu tidak berani mengambil resiko dalam hidupmu, kamu tidak akan pernah bisa menciptakan masa depan".*

"Tidak peduli seberapa sulit atau tidak mungkin untuk dicapai, kamu tidak boleh kehilangan pandangan terhadap tujuanmu!"

(Monkey D Luffy)

# KATA PENGANTAR

**Assalamualaikum Warahmatullahi Wabarakatuh Allhamdulillahirobbil'alamin,**

***Al-hamdu lillahi rabbil 'alamin***, segala puji dan syukur atas nikmat umur, iman, dan rezeki yang berlimpah yang telah Allah SWT berikan. Alhamdulillah dengan izin Allah SWT penelitian tugas akhir yang berjudul "Implementasi Arsitektur *Enterprise* Pola *Vending Machine* pada Teknologi *Microservices"* dapat diselesaikan dengan baik. Penulis menyadari bahwa dalam peneyelesaian tugas akhir banyak pihak yang telah membantu. Untuk itu, saya menyampaikan rasa terima kasih saya kepada yang terhormat:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D. Fathul Wahid, selaku Rektor Universitas Islam Indonesia.
2. Kedua orang tua yang selalu memberikan motivasi dan semangat serta doa untuk menyelesaikan tugas akhir.
3. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Hendrik, S.T., M.Eng., selaku Ketua Jurusan Informatika Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., dan bapak Hanson Prihantoro Putro, S.T., M.T. selaku dosen pembimbing tugas akhir saya yang telah memberikan masukkan, ilmu, dan juga sabar selama membimbing hingga terselesaikannya skripsi ini.
7. Teman-teman Remaja Cendana Asri yang selalu menghibur di setiap keadaan.
8. Teman-teman Sejahtera Grup yang selalu membantu dan memotivasi.
9. Terima kasih kepada teman-teman yang juga memberikan semangat dan bantuan kepada saya terutama kepada teman saya Muhammad Hanif Faturohman.
10. Seluruh pihak yang terlibat dan tidak bisa saya sebutkan satu per satu, terima kasih atas segala dukungannya.

Yogyakarta, 25 Agustus 2021

( Muhammad Iqbal )

# SARI

Salah satu tujuan dari penerapan arsitektur *enterprise* adalah menciptakan kesesuaian antara bisnis dan teknologi informasi bagi kebutuhan organisasi. Di sisi lain, *microservices* merupakan teknologi yang sering digunakan belakang ini berkat skalabilitasnya. Literatur yang membahas rancangan arsitektur *enterprise* yang diimplementasikan menggunakan teknologi *microservices* masih sangat terbatas. Begitu juga tingkat keberhasilannya yang masih rendah merupakan permasalahan yang dihadapi pada penelitian ini. Maka dari itu penelitian ini bertujuan membangun kerangka implementasi pola arsitektur *enterprise* menggunakan teknologi *microservices*. Dalam pengembangan penelitian ini menggunakan metode yang digunakan secara berulang dan menggunakan kerangka kerja TOGAF pada tiap ranah lapisan arsitektur *enterprise* yang disesuaikan ke dalam kasus yang diangkat yaitu *vending machine* (toko *online*), serta menghasilkan aplikasi *vending machine* (toko *online*) berbasis *microservices* yang mampu mengelola kebutuhan pada vending machine (toko *online*). Rancangan ini diharapkan dapat mempermudah pengembangan sistem *enterprise* serta meningkatkan peluang keberhasilannya.

Kata kunci: arsitektur *enteprise*, *microservices*, pola *vending machine*

# GLOSARIUM

| | |
|---|---|
| Vending machine | sebuah pola yang digunakan untuk pemecahan masalah dalam penyajian produk melalui internet. |
| Timeframe | kurun waktu tertentu yang digunakan untuk pengamatan harga. |
| Reuse | dapat digunakan kembali pada fungsi yang sama ataupun fungsi lainnya. |
| Microservices | memecah aplikasi menjadi layanan-layanan kecil yang bersifat independen, berkomunikasi melalui satu sama lain dengan baik. |
| Arsitektur *enterprise* | sebuah sarana yang digunakan dalam mewujudkan kesesuaian teknologi informasi dengan bisnis yang dijalankan sebuah organisasi. |

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Arsitektur *enterprise* adalah sebuah sarana yang digunakan dalam mewujudkan kesesuaian teknologi informasi dengan bisnis yang dijalankan sebuah organisasi. Arsitektur *enterprise* merepresentasikan tujuan *stakeholder* yang mencakup informasi, kegunaan, lokasi organisasi dan parameter kinerja. Arsitektur *enterprise* menggambarkan rencana untuk mengembangkan suatu sistem atau sekumpulan sistem teknologi informasi (Osvalds, 2001). Secara umum arsitektur *enterprise* sangat diperlukan, contohnya dalam menyelaraskan kondisi nyata perusahaan terhadap tujuan manajemen, mengurangi perubahan yang mungkin terjadi dalam pengembangan sistem, mengurangi *timeframe*, dan mengurangi kebutuhan sumber daya (Shah & El Kourdi, 2007). Karena rencana awal telah menetapkan kesesuaian antara teknologi informasi dan bisnis yang akan dijalankan.

Pembuatan arsitektur *enterprise* dimulai dengan memodelkan keseluruhan sudut pandang yang berkaitan (Osvalds, 2001), serta mengelompokkan sudut pandang berdasarkan ranah arsitektur menggunakan kerangka kerja yang digunakan (Perroud & Inversini, 2013). Kerangka kerja yang digunakan akan menjadi patokan dalam pembuatan arsitektur *enterprise*. Ada berbagai macam kerangka kerja yang dapat dimanfaatkan. Contohnya, TOGAF dan Zachman (Shah & El Kourdi, 2007).

Arsitektur *enterprise* memiliki beberapa ranah arsitektur yang tersusun ke dalam sejumlah lapisan. Antar kerangka kerja tidak memiliki kepastian akan jumlah dan ranah arsitektur penyusunnya. Laporan ini menggunakan ranah arsitektur yang ada pada TOGAF yaitu: arsitektur bisnis, arsitektur data, arsitektur aplikasi, dan arsitektur teknologi. Dikarenakan arsitektur yang lebih sederhana, TOGAF juga diharapkan dapat memperluas tingkat penerimaan terhadap kerangka implementasi yang ditawarkan dengan menggunakan beberapa pemetaan (Perroud & Inversini, 2013).

Perancangan arsitektur di tiap ranah dilakukan secara bertahap mulai dari arsitektur bisnis, arsitektur data, arsitektur aplikasi, hingga arsitektur teknologi. Perancangan arsitektur dilakukan secara bertahap untuk meyakinkan bahwa kebutuhan bisnis *enterprise* dibantu sepenuhnya oleh sistem teknologi informasi *enterprise* mulai dari ranah arsitektur data, aplikasi hingga teknologi. Hal ini dilakukan agar kebutuhan bisnis *enterprise* didukung sepenuhnya

oleh sistem yang akan dibangun. Tetapi perlu diperhatikan bahwa sebaiknya arsitektur *enterprise* dibebaskan dari pilihan teknologi yang akan digunakan. (C. Richardson, 2015).

Dalam arsitektur *enterprise* terdapat pola yang menjadi salah satu cara untuk menggambarkan solusi untuk masalah berulang *(reuse).* Banyak organisasi yang berusaha mengumpulkan pengalaman berharga, merumuskan dan mendokumentasikan solusi untuk masalah yang berulang (*reuse*). Pola dalam arsitektur *enterprise* dapat digunakan sebagai acuan dalam mendokumentasikan solusi bagi masalah yang berulang (*reuse*). Pola tersebut akan tersusun dalam beberapa lapis ranah. Salah satunya dari sudut pandang proses bisnis, tetapi menunjukkan juga dampak yang akan terjadi pada lapisan bawah (Perroud & Inversini, 2013). Contohnya, arsitektur *enterprise* pola *vending machine* yang ingin mengatasi masalah peningkatan penjualan yang sebelumnya menggunakan metode pemasaran klasik yang kurang efektif untuk menarik *customer* serta meningatkan penjualan. Untuk itu proses bisnis baru yang didukung oleh arsitektur *enterprise* diperlukan untuk menemukan solusi masalah peningkatan penjualan. Pola *vending machine* dapat digunakan untuk menyediakan produk maupun jasa melalui katalog untuk diperjualbelikan melalui internet yang dapat mempermudah *customer* dalam memilih produk tanpa harus datang ke toko serta hal ini dapat meningkatkan penjualan dikarenakan dapat dilakukan dimanapun serta pemasaran akan jauh lebih luas dikarenakan melalui internet (Perroud & Inversini, 2013).

Pengembangan sistem *enterprise* dapat dilakukan dengan menggunakan teknologi *microservices* salah satunya. Dalam pengembangan, perubahan bisa terjadi pada proses bisnis maupun beban komputasi. Teknologi *microservices* menerapkan konsep gunaulang (*reuse*) pada pengembangan sistem *enterprise* sehingga memudahkan sistem untuk dikembangkan lebih cepat dan sistem dapat menyesuaikan perubahan kebutuhan (T. Cerny, 2019).

Dalam hal teknologi, arsitektur *microservices* merupakan salah satu teknologi yang sering digunakan belakangan ini berkat skalabilitasnya. Karakteristik arsitektur *microservices* adalah memecah aplikasi menjadi layanan-layanan kecil yang bersifat independen, berkomunikasi melalui satu sama lain dengan baik, serta menggunakan mekanisme yang ringan untuk melayani tujuan bisnis. Komunikasi layanan satu sama lain bergantung pada kebutuhan aplikasi yang digunakan. Selain itu, berkat skalabilitasnya biaya dalam proses pengembangan teknologi *microservices* akan relatif lebih murah (Huston Tom, 2015).

Sebuah survei mengatakan bahwa dalam adopsi teknologi *microservices* sekitar 28% organisasi responden telah menggunakan teknologi *microservices* setidaknya kurun waktu 3 tahun dan sebanyak 62% responden telah menggunakan teknologi *microservices* selama satu

tahun lebih. Sebanyak 10% melaporkan bahwa adopsi teknologi *microservices* mereka telah sukses total, sedangkan mayoritas (54%) melaporkan bahwa setidaknya Sebagian besar berhasil. Sejumlah pengadopsi mengatakan bahwa kompleksitas yang menjadi tantangan terbesar bagi mereka (Loukides & Swoyer, 2020).

Teknologi *microservices* sangat dibutuhkan dalam pengembangan kasus *vending machine* (toko *online*) yang kompleks dan sulit untuk di *maintenance*. Teknologi *microservices* mampu beradaptasi dengan perubahan terutama pada kompleksitas dan *maintenance*. Seperti kendala yang ditemukan pada penelitian serupa yang mengatakan bahwa sebagian besar beralih dari *monolithic* ke teknologi *microservices* dikarenakan masalah utama yaitu: kompleksitas, dan juga sulitnya *maintenance monolithic*.

Maka dari itu penelitian ini bertujuan menggali lebih dalam mengenai deskripsi implementasi pola arsitektur *enterprise* pada teknologi *microservices*. Dengan menggunakan pola *vending machine* sebagai acuan, serta menggunakan ranah arsitektur yang ada pada TOGAF, Penelitian ini diharapkan dapat memudahkan dalam mengembangkan sistem *enterprise* dan meningkatkan peluang keberhasilan.

## 1.2 Rumusan Masalah

Bagaimana proses implementasi arsitektur *enterprise* pola *vending machine* menjadi sebuah aplikasi berbasis teknologi microservices?

## 1.3 Batasan Masalah

Hal-hal yang menjadi batasan impelementasi arsitektur *enterprise* pola *vending machine* antara lain:

a. Menggunakan TOGAF sebagai kerangka kerja penelitian.
b. Berfokus pada implementasi teknologi *microservices* dalam mengembangkan arsitektur *enterprise* pola *vending machine*.
c. Berfokus pada implementasi fungsionalitas, bukan pada antarmuka.

## 1.4 Tujuan Penelitian

Tujuan penelitian ini adalah memberikan deskripsi implementasi pola arsitektur *enterprise* pada teknologi *microservices* dengan menggunakan pola *vending machine* pada buku (Perroud & Inversini, 2013) sebagai acuan.

## 1.5 Manfaat Penelitian

### 1.5.1 Manfaat Bagi Pengembang

Dapat mengetahui proses implementasi pola arsitektur *enterprise* pada teknologi *microservices.*

### 1.5.2 Manfaat Bagi *Stakeholder* Pola *Vending Machine*

Dapat mengetahui proses bisnis yang digunakan dalam implementasi pola arsitektur *enterprise* pada teknologi *microservices.*

### 1.5.3 Manfaat Bagi Peneliti

Dapat mengetahui keuntungan dan kekurangan dalam proses implementasi pola arsitektur *enterprise* pada teknologi *microservices*.

## 1.6 Metodologi Penelitian

Langkah-langkah yang digunakan dalam proses penerapan implementasi pola arsitektur *enterprise* pada teknologi *microservices* adalah sebagai berikut:

a. Dekomposisi Arsitektur *Enterprise* Pola Vending Machine

Dalam mendekomposisikan arsitektur *enterprise* pola *vending machine* digunakan Domain-Driven Design (DDD) yang merupakan pendekatan dalam pengembangan aplikasi yang kompleks untuk menghubungkan antara konsep bisnis dan pengimplementasian teknikal. Domain-Driven Design (DDD) berfokus kepada desain aplikasi yang mempercepat pengembangan aplikasi.

b. Desain API Layanan

Pada desain API layanan terdapat beberapa usulan tahap yang harus dilakukan seperti pada buku (Richardson, 2018), yaitu: mengidentifikasi API layanan, Mengidentifikasi fungsi yang ada pada API layanan, Mengidentifikasi proses hubungan antar API layanan, Mengidentifikasi format komunikasi yang ada pada API layanan, Mengidentifikasi kebutuhan pola saga yang ada pada API layanan.

c. Desain Skema Basis Data

Desain skema basis data dilakukan untuk mendefinisikan tiap skema basis data.

d. Implementasi Basis Data

Basis data dirancang berdasarkan arsitektur *enterprise* pola *vending machine* yang selanjutnya basis data dikelompokkan ulang sesuai dengan kebutuhan pengimplementasian.

e. Implementasi API Layanan

Pengembangan API mengimplementasikan arsitektur *enterprise* toko *online* yang dikembangkan berdasar pola *vending machine* yang sudah disesuaikan dengan kebutuhan *microservices* pada saat pengembangan.

f. Pengujian API Layanan

Pengujian API layanan melalui dua tahap, yaitu: *unit testing*, dan juga *integration testing*.

g. Implementasi Antarmuka

Antarmuka dikembangkan untuk arsitektur *enterprise* toko *online* yang dikembangkan berdasar pola *vending machine* yang telah dimodifikasi. Pengembangan antarmuka menggunakan teknologi React.js.

h. Pengujian Antarmuka

Pengujian dilakukan untuk mengetahui apakah proses bisnis berjalan dengan baik. Pengujian proses bisnis dilakukan berdasarkan sudut pandang masing-masing pengguna. Pengujian dilakukan terhadap sekaligus semua fungsionalitas pada layanan API melalui antarmuka.

## 1.7 Sistematika Penulisan

Sistematika penulisan dibuat secara terstruktur agar pembahasan yang ada dapat dipahami secara runtut. Terdapat lima bab dalam penelitian ini. Berikut merupakan penjelasan isi lima bab tersebut.

BAB I PENDAHULUAN

Bab ini menjelaskan masalah utama dari topik yang diambil meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi keseluruhan kajian pustaka yang digunakan dalam penelitian.

BAB III METODOLOGI PENELITIAN

Bab ini membahas Langkah-langkah yang dilakukan dalam proses penerapan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas hasil dari keseluruhan langkah-langkah yang telah dilakukan dan keluaran dalam proses penerapan.

# BAB V KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dari keseluruhan proses penerapan yang telah dilakukan serta saran yang diberikan untuk penelitian serupa selanjutnya berdasarkan hasil penelitian saat ini.

# BAB II
# LANDASAN TEORI

## 2.1 Arsitektur *Enterprise*

### 2.1.1 Definisi

Pendekatan arsitektur *enterprise* pada perusahaan dapat membantu menyesuaikan sumber daya bisnis dan TI. Selain itu, arsitektur *enterprise* juga dapat menyesuaikan prinsip-prinsip dasar dan metodologi umum yang mengatur seluruh sistem informasi pada saat proses pengembangan. (Shah & El Kourdi, 2007).

Secara umum keuntungan penggunaan arsitektur *enterprise* adalah (Shah & El Kourdi, 2007):

a. Menyelaraskan kenyataan perusahaan terhadap maksud implementasi manajemen.
b. Memberikan konsistensi terhadap penerapan aturan dalam perusahaan.
c. Mengelola dan memfasilitasi perubahan yang terjadi pada setiap aspek perusahaan.
d. Mengurangi perubahan dalam pengembangan sistem, mengurangi perubahan timeframe, dan mengurangi kebutuhan sumber daya.

### 2.1.2 Kerangka Kerja Arsitektur *Enterprise*

Kerangka kerja arsitektur *enterprise* membantu menyediakan struktur pengorganisasian dan pemodelan informasi dalam arsitektur *enterprise*. Arsitektur *enterprise* dapat memfasilitasi arsitektur yang diperlukan beserta hubungannya satu sama lain dalam proses pengembangan. Saat ini telah banyak kerangka arsitektur *enterprise* yang digunakan dalam proses pengembangan sistem informasi, salah satunya TOGAF.

TOGAF melalui *Architecture Development Method* (ADM) memberikan metode dalam mengembangkan dan mengelola serta mengimplementasikan arsitektur *enterprise*. ADM merupakan sebuah metode generik yang berisi sekumpulan aktivitas untuk memodelkan pengembangan arsitektur *enterprise*. Metode ini juga bisa digunakan sebagai acuan atau alat yang digunakan untuk merencanakan, merancang, mengembangkan dan mengimplementasikan arsitektur sistem informasi untuk organisasi (Yunis & Surendro, 2009). Berikut ini merupakan ranah arsitektur yang ada pada TOGAF (Perroud & Inversini, 2013):

a. Arsitektur Bisnis

Arsitektur bisnis mendeskripsikan proses bisnis organisasi. Arsitektur ini merupakan penggerak bagi komponen lain dari arsitektur *enterprise*.

b. Arsitektur Data/Informasi

Arsitektur data/informasi merepresentasikan data/informasi sebagai aset untuk mendukung bisnis.

c. Arsitektur Aplikasi

Arsitektur aplikasi mendeskripsikan aplikasi-aplikasi utama yang akan digunakan dalam mengelola data dan diperlukan dalam mendukung bisnis.

d. Arsitektur Teknologi

Arsitektur teknologi mendeskripsikan *platform* teknologi sebagai penyedia lingkungan bagi aplikasi-aplikasi tersebut.

Arsitektur *enterprise* memiliki pola yang digunakan untuk memodelkan solusi bagi masalah yang serupa. pola tersebut tersusun dalam beberapa lapis arsitektur seperti yang ada pada TOGAF. Pola tersebut dapat digunakan sebagai contoh untuk mendokumentasi solusi bagi masalah yang serupa secara berurutam mulai dari lapisan ranah teratas hingga bawah.

## 2.2 Pola Arsitektur *Vending Machine*

### 2.2.1 Defensi

Pola arsitektur *vending machine* merupakan pola untuk membantu pemecahan masalah dalam penyajian produk atau layanan dari katalog untuk diperjualbelikan melalui *platform* internet (toko berbasis *web*), baik pada *Business-to Consumer* (B2C) atau *Business-to-Business* (B2B). Pola ini memungkinkan *customer* dapat menelusuri katalog *online*, memperoleh informasi harga produk, detail produk, dan memasukkan produk yang diinginkan ke dalam keranjang pembelian *virtual*. *Customer* difasilitasi dalam proses pembayaran hingga nantinya produk diproses oleh penjual (Perroud & Inversini, 2013).

Berdasarkan definisi tersebut, berikut beberapa kemampuan dari pola arsitektur *vending machine*:

a. Mengelola dan menyajikan katalog artikel dengan info produk kepada *customer*.
b. Menjual atau memberi segala jenis produk atau jasa yang dapat dijual dan dibeli (termasuk yang dapat diunduh seperti artikel, perangkat lunak, atau dokumen elektronik).

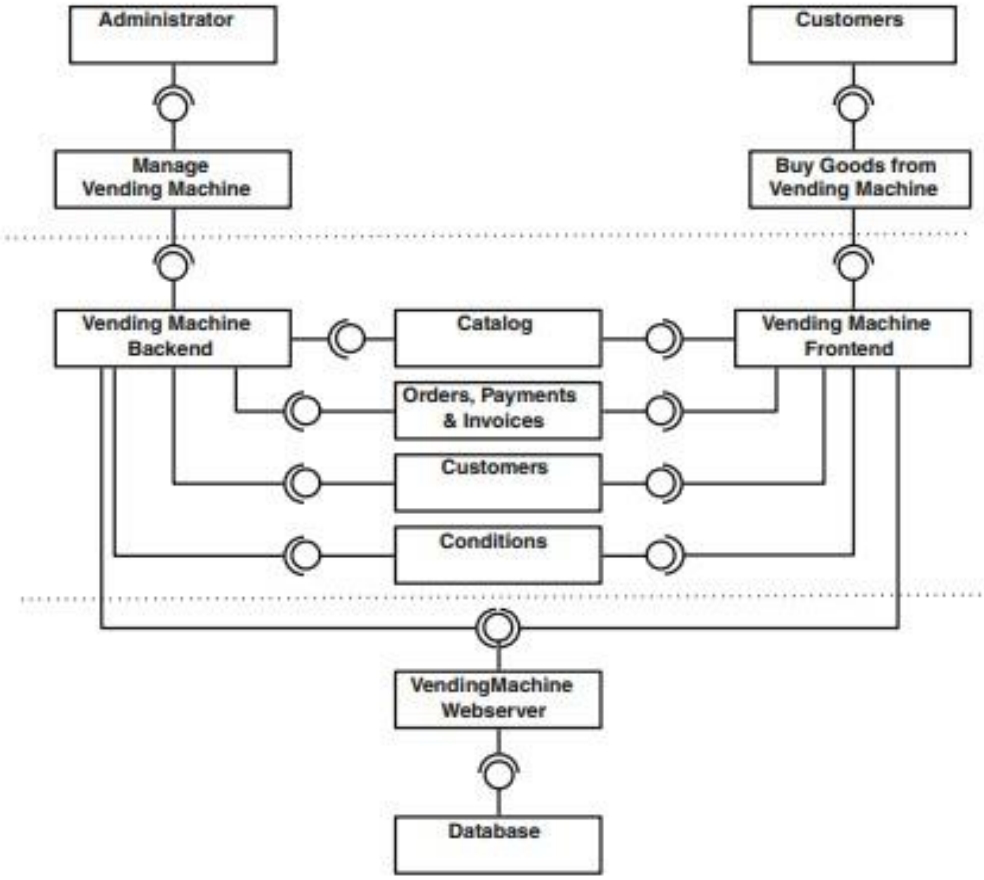
c. Memberi *customer* kemungkinan untuk meletakkan produk ke dalam keranjang *virtual*.
d. Memandu *customer* melalui proses pembayaran sesuai dengan pengiriman dan kondisi pembayaran yang ditentukan oleh kebutuhan bisnis.
e. Memberi *customer* kemampuan untuk mengelola profil dan pesanan mereka.
f. Membantu dalam manajemen *order* dan pembayaran.
g. Memberi pemilik *vending machine* sarana untuk menganalisis dan melacak aktivitas di *platform online*.

Pola ini dapat digunakan untuk setiap jenis bisnis dan untuk semua ukuran perusahaan yang menjual produk. Pola ini dapat menyediakan jalur bagi penjualan dengan berbagai jenis namun tidak wajib diterapkan di setiap perusahaan. Dua pelaku utama adalah

a. Perusahaan itu sendiri, yang diwakili oleh pengelola *vending machine* (dengan istilah *administrator* yang dimaksud adalah orang-orang yang bertugas mengelola *vending machine* dari sudut pandang bisnis) dan.
b. *Customer*, yakni orang-orang yang menggunakan *vending machine* untuk membeli produk dan mendapatkan keuntungan layanan *online*.

### 2.2.2 Pandangan *Holistic*

Pandangan ini memuat proses bisnis beserta aktor dan teknologi yang digunakan dalam membagun arsitektur *vending machine*.




Gambar 2.1 Pandangan holistic. Sumber: (Perroud & Inversini, 2013)

Gambar 2.1 menunjukkan kotak dari paling atas yaitu aktor *customer* yang menggunakan proses bisnis *buy goods from vending machine* dalam menemukan produk yang diinginkan dengan mengakses kotak aplikasi yang berada di tengah (*catalog, order, customer, dan conditions*) dan *administrator* yang menggunakan proses bisnis *manage vending machine* untuk mengatur keseluruhan aplikasi yang berada pada kotak di tengah (*catalog, order, customer, dan conditions*). Masing-masing bagian menyediakan antarmuka dan layanan khusus yang dapat diakses melalui *frontend* dan *backend*. Berikut Tabel 2.1 yang menjelaskan deskripsi pada tiap layanan yang ada pada pandangan holistic:

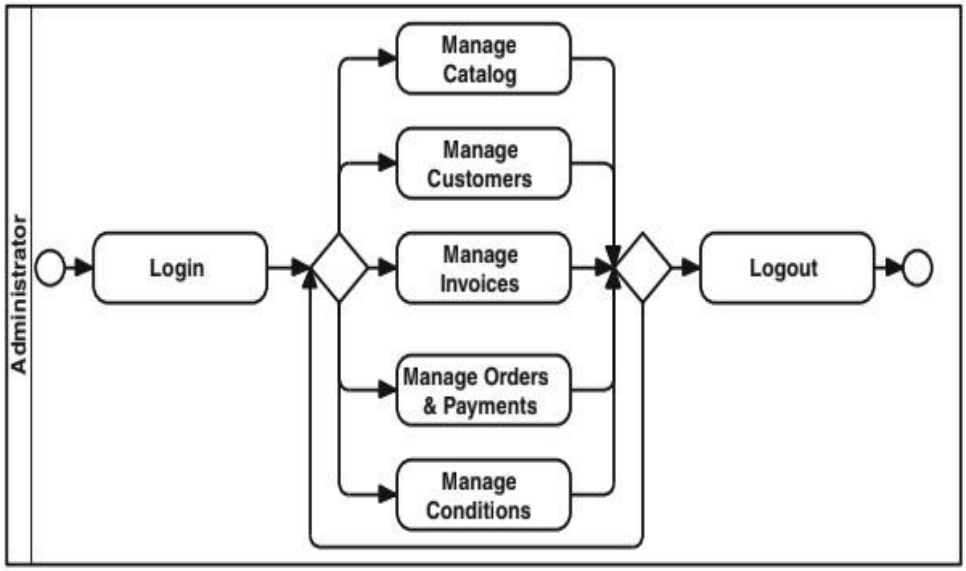
Tabel 2.2.1 Penjelasan layanan yang ada pada pandangan *holistic*

| **Kotak** | **Deskripsi** |
|---|---|
| Vending machine backend | Memungkinkan *administrator* untuk mengelola *catalog, invoices, payment, customer,* dan *condition.* |

| Kotak | Deskripsi |
|---|---|
| Vending machine frontend | Menyediakan fungsionalitas utama bagi *customer*. Ini memungkinkan *customer* untuk mencari dan menjelajahi katalog, untuk mengelola daftar keinginan, profil mereka, pesanan, dll. |
| Catalog | Kotak ini adalah bagian utama dari VendingMachine (B2) dan berisi semua fail artikel dalam struktur (hierarki). Semua artikel dijelaskan dengan kumpulan atribut (harga, warna, berat, gambar, dan sebagainya). |
| Orders, payments, and invoices | Berisi semua pesanan dan statusnya (tertunda, ditutup, ditahan, dll.), produk yang dipesan dan kondisi pengiriman serta pembayaran yang dipilih untuk urutan tertentu. Fungsi untuk mengelola *invoices* yang dikirim ke *customer* juga didukung. |
| Customer | Aplikasi untuk mengelola informasi *customer* dan membiarkan *customer* memodifikasi profil mereka. |
| Conditions | Berisi segala sesuatu yang berkaitan dengan syarat dan ketentuan komersial dari *vending machine*. pengiriman, pembayaran, aturan harga, *disclaimer*, dan sebagainya. |

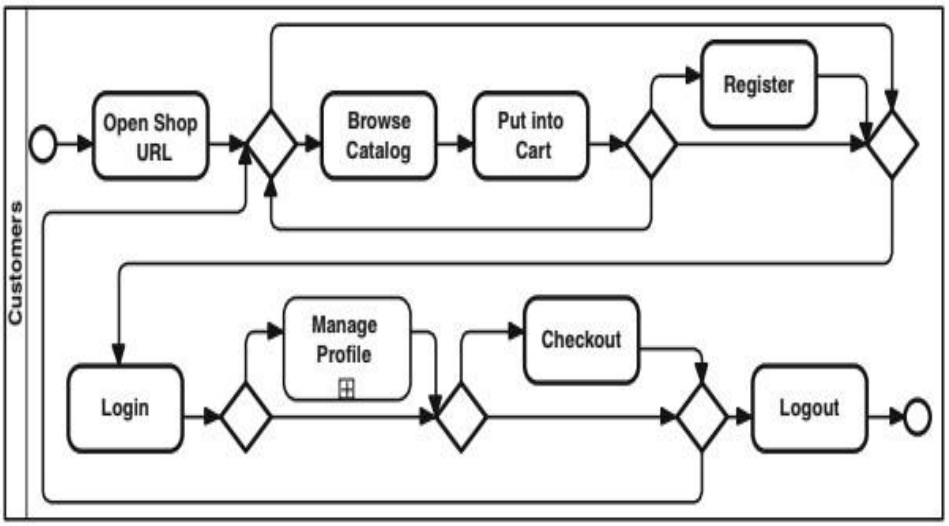
### 2.2.3 Proses Bisnis

Terdapat dua proses bisnis dalam arsitektur *enterprise* pola *vending machine* yaitu proses bisnis *administrator* dan *customer*.




Gambar 2.2 Proses bisnis *administrator*. Sumber: (Perroud & Inversini, 2013)

Proses bisnis yang pertama sebagaimana pada Gambar 2.2 merupakan proses dimana *administrator* mengelola *catalog*, seluruh akun *customer*, informasi pemesanan, serta *conditions*.

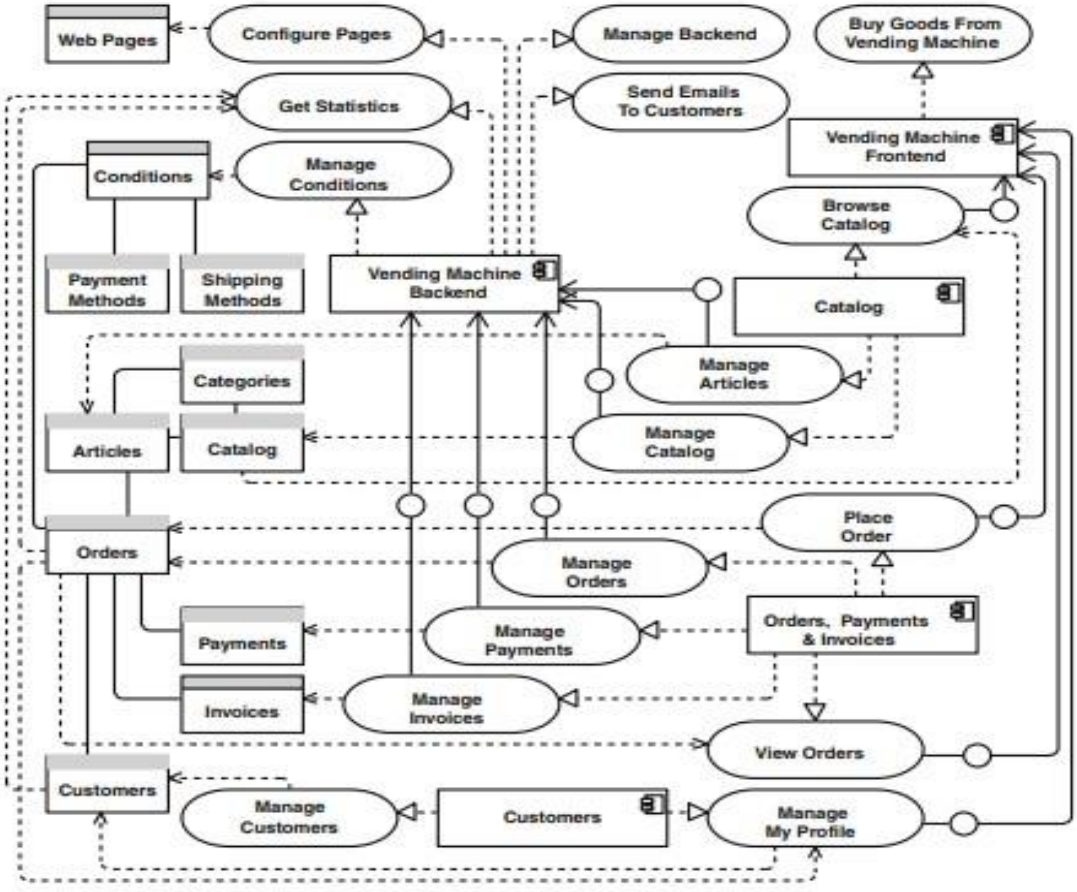



Gambar 2.3 Proses bisnis *customer*. Sumber: (Perroud & Inversini, 2013)

Proses bisnis kedua seperti yang ditunjukkan pada Gambar 2.3 terkait dengan bagaimana *customer* yang menggunakan layanan *vending machine*. Proses ini tergantung dengan fungsionalitas yang disediakan. Biasanya, *customer* akan melakukan penelusuran *catalog* dan memasukkan artikel ke keranjang. *Vending machine* mendukung fungsi pencarian, sehingga

*customer* dapat dengan cepat menemukan file artikel bergantung pada fitur seperti kategori, warna, harga, atau ukuran. Tergantung pada syarat dan ketentuan yang berlaku, *customer* dapat melakukan pembayaran tanpa harus mendaftar, atau *customer* dapat melakukan pembayaran sebagai tamu (tidak ada profil pribadi yang akan disimpan untuk kunjungan berikutnya ke vending machine). *Customer* terdaftar dapat mengakses pesanan dan profilnya setelah *login* (Kelola Profil tidak ditampilkan secara rinci di pola).

### 2.2.4 Struktur Data dan Aplikasi




Gambar 2.4 Struktur data dan aplikasi. Sumber: (Perroud & Inversini, 2013)

Gambar 2.4 menunjukkan objek data terpenting di sisi kiri dan koneksi antar komponen aplikasi di sisi kanan. Setiap aktivitas proses bisnis mempunyai layanan yang sesuai dengan lapisan aplikasi. *Customer* hanya dapat mengakses dua layanan yakni: *manage my profile* dan *buy goods from vending machine* yang memungkinkan operasi penulisan data objek, *place order*, dan *manage my profile* (alamat, kata sandi dan nama pengguna.). Pengelolaan dalam *vending machine* juga termasuk *configures pages* yang memungkinkan membuat dan

mengubah konten web, seperti halaman informasi, jam buka, serta informasi kontak. *Administrator* juga bisa mendapatkan statistik tentang kinerja *vending machine* (contoh: jumlah penjualan produk dalam suatu bulan, serta informasi keuangan yang akan diteruskan ke aplikasi finansial).

Dalam aplikasi, katalog sering kali disusun dalam kategori. Suatu produk mungkin berbeda kategori, seperti sebuah sepatu akan berada dalam kategori model sepatu yang berbeda. serta tedapat beberapa Ketentuan (*conditions*) yang dipilih oleh *customer* seperti apa metode pengiriman dan pembayaran, yang akan digunakan dalam proses pengiriman dan pembayaran.

### 2.2.5 Teknologi Microservices

Tujuan utama teknologi *microservices* adalah memecah dan merangkai aplikasi menjadi layanan-layanan kecil. Setiap layanan memiliki tanggung jawab masing-masing, tetapi tetap saling berkomunikasi. Sistem pada teknologi *microservices* juga didistribusikan secara independen yang hasilnya setiap layanan dapat beradaptasi dengan setiap perubahan kebutuhan (Lewis & Fowler, 2014).

Dalam setiap upaya penerapan teknologi tentunya tidak terlepaskan dari kelemahan dan kelebihan pada saat digunakan. beberapa kelemahan dalam penerapan teknologi *microservices* adalah kesulitan dalam membuat komunikasi antar layanan dan mengatasi kegagalan di sebagian layanan serta menerapkan permintaan yang menjangkau banyak layanan yang membutuhkan koordinasi yang baik antar layanan, Serta penggunaan memori yang jauh lebih tinggi dan biaya yang dikeluarkan menjadi lebih besar.

Sedangkan kelebihan teknologi *microservices* diungkapkan langsung oleh responden pada survei yang dilakukan (Loukides & Swoyer, 2020). Hasil survei mengatakan bahwa:

- 45% responden senang dengan fleksibilitas fitur.
- Di bawah 45% responden mengatakan bahwa teknologi *microservices* merespons dengan cepat terhadap perubahan teknologi dan kebutuhan bisnis.
- Di bawah 44% responden mengatakan bahwa manfaat dari skalabilitas keseluruhan yang lebih baik.
- 15% responden merujuk pada biaya pengembangan yang lebih rendah.

Teknologi *microservices* memiliki 2 struktur komposisi, yaitu koreografi dan orkestrasi. Pada orkestrasi, terdapat proses bisnis tunggal yang dieksekusi secara terpusat dengan mengkoordinasikan hubungan antar layanan. Pada koreografi merupakan keseluruhan layanan

yang berpartisipasi menentukan pertukaran pesan dan aturan interaksi dengan dua atau lebih keluaran serta menggunakan pendekatan desentralisasi dalam komposisi layanan (T. Cerny, 2019).

### 2.2.6 Penelitian Serupa

Kajian Pustaka ini mengkaji empat penelitian serupa yang menggunakan teknologi *microservices*. Berikut empat kajian Pustaka tersebut:

a. Penelitian (Putra & Saputra, 2019) membahas desain dan Implementasi teknologi *microservices* studi kasus pada layanan *taking order* ( aplikasi *e-commerce* PT. Xyz ). Penelitian ini dilakukan dikarenakan penggunaan arsitektur *monolithic* yang memperlambat perkembangan aplikasi. maka dari itu penelitian ini dilakukan dengan tujuan mengatasi masalah yang ada pada arsitektur *monolithic* dengan menggunakan arsitektur *microservices* dengan cara membagi layanan menjadi kecil.

b. Penelitian (Mufrizal & Indarti, 2019) membahas mengenai *refactoring* arsitektur teknologi *microservices* pada aplikasi absensi PT. Graha Usaha Teknik. Penelitian ini dilakukan karena berbagai masalah yang ditemukan pada saat penggunaan arsitektur *monolithic* seperti: meningkatnya jumlah pengguna akan mempengaruhi proses *maintenance*, kinerja dan sulitnya pembaruan aplikasi. Permasalahan selanjutnya ditemukan apabila terjadi perubahan modul yang membutuhkan proses *restart* aplikasi yang mana pada saat di *restart* aplikasi tersebut tidak dapat digunakan. maka dari itu penelitian ini bertujuan untuk melakukan *refactoring* arsitektur *microservices* yang dianggap dapat menyelesaikan masalah yang ada pada arsitektur *monolithic* dengan cara membagi layanan menjadi kecil yang tiap layanan memiliki basis data masing-masing.

c. Penelitian (Rafiqi et al., 2019) membahas mengenai implementasi arsitektur teknologi *microservices* pada aplikasi *online travel* tourinc untuk membuat sebuah aplikasi yang mampu menangani kompleksitas salah satunya pada akses REST API yang dibutuhkan untuk mengakses diantaranya akses ke PT. KAI, penyedia jasa penerbangan (Garuda Indonesia, Lion Air, dll), berbagai hotel. Serta menangani proses konfirmasi pembayaran. Untuk dapat membuat aplikasi tersebut serta mengatasi kebutuhan yang diperlukan, Digunakan arsitektur

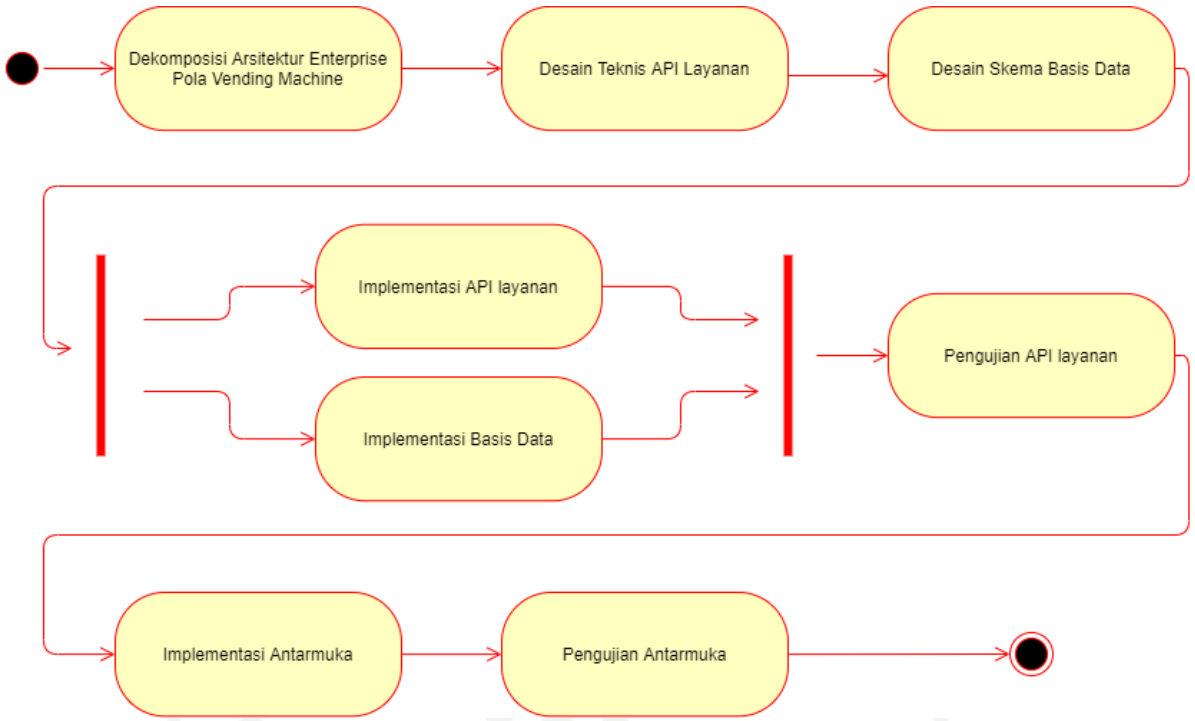
*microservices* yang tiap layanan aplikasi dibagi-bagi menjadi kecil yang memungkinkan meminimalisir kompleksitas.

d. Penelitian (Purnama, Heri, 2010) membahas aplikasi pengelolaan skripsi di STMIK AKAKOM menggunakan arsitektur teknologi *microservices* dengan node.js. penelitian ini bertujuan mengatasi kompleksitas dalam penyimpanan seluruh skripsi mahasiswa STMIK AKAKOM YOGYAKARTA dan *maintenance* yang dilakukan lebih mudah dengan cara tiap layanan memiliki basis datanya sendiri.

Dari seluruh penelitian serupa, belum ada pengimplementasian teknologi *microservices* yang didasari oleh arsitektur *enterprise*. Maka dari itu, penelitian ini bermaksud mengimplementasikan teknologi *microservices* menggunakan pola yang didasari oleh arsitektur *enterprise*.

# BAB III
# Metodologi Penelitian

## 3.1 Diagram Alir Metodologi Penelitian




Gambar 3.1 *Activity diagram* metodologi penelitian

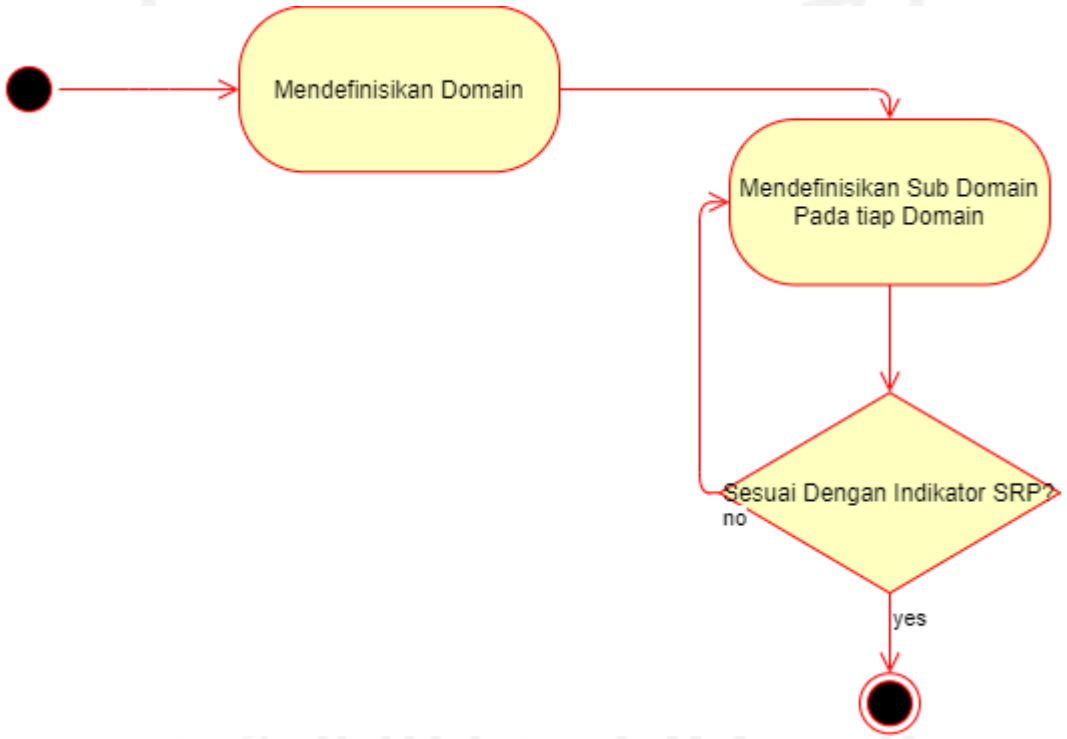
Gambar 3.1 merupakan gambaran umum langkah-langkah dalam metodologi penelitian. Dimulai dengan mendekomposisikan pola arsitektur *enterprise* dan mendesain teknis tiap API layanan menggunakan cara yang digunakan pada buku (Richardson, 2018). Kemudian menyusun desain skema basis data menggunakan cara yang disarankan langsung oleh MongoDB. Selanjutnya setelah semua desain telah disusun masuk kebagian pengimplementasian API layanan dan juga basis data yang dapat dilakukan secara bersamaan (paralel). Pada bagian pengimplementasian API layanan digunakan teknologi NodeJS dan ExpressJS. Kemudian pada basis data menggunakan teknologi MongoDB.

Setelah dilakukannya pengimplementasian pada API layanan dan juga basis data, dilakukan pengujian terhadap API layanan dan juga basis data. Pengujian ini dilakukan untuk menguji kualitas keduanya seperti yang dilakukan pada buku (Richardson, 2018). Setelah dilakukannya pengujian terhadap API layanan dan basis data. Kemudian dilakukan

pengimplementasian terhadap antarmuka dan pengujian kualitas antarmuka agar mudah diakses oleh pengguna (Richardson, 2018).

### 3.2 Dekomposisi Arsitektur *Enterprise* Pola Vending Machine

Dalam mendekomposisikan arsitektur *enterprise* pola *vending machine* digunakan Domain-Driven Design (DDD) yang merupakan pendekatan dalam pengembangan aplikasi yang kompleks untuk menghubungkan antara konsep bisnis dan pengimplementasian teknikal. Domain-Driven Design (DDD) berfokus kepada desain aplikasi yang mempercepat pengembangan aplikasi.




Gambar 3.2 *Activity diagram* metodologi dekomposisi

Pada gambar 3.2 menjelaskan dua aktivitas pendefinisian dan proses validasi. Aktivitas ini dilakukan untuk memisahkan fungsi dari bisnis yang ada pada pola arsitektur *enterprise* pola *vending machine*. Pada aktivitas pendefinisian sub domain dilakukan pemisahan fungsi bisnis dari tiap domainnya. Kemudian menetukan apakah dekomposisi yang dilakukan sudah tepat dengan menggunakan *single responsibility principle* (SRP).

*Single responsibility principle* (SRP) digunakan untuk menyaring tanggung jawab yang dimiliki pada tiap layanan. Dimana pada tiap layanan diharuskan memiliki satu tanggung jawab

tidak boleh lebih. Apabila satu layanan memiliki lebih dari 1 tanggung jawab maka, layanan tersebut harus dipecah Kembali karena dianggap terlalu besar (Richardson, 2018).

Dalam tahap penyaringan ini akan menghasilkan model sub domain dari arsitektur *enterprise* pola *vending machine* yang akan digunakan untuk penentu detail dari tiap API layanan pada tahap selanjutnya.

### 3.3 Desain API Layanan

Pada desain API layanan terdapat beberapa usulan tahap yang harus dilakukan seperti pada buku (Richardson, 2018). Berikut merupakan tahapan tersebut.

a. Mengidentifikasi API layanan

   Perlunya mengidentifikasi API layanan melalui sub domain untuk meminimalisir kendala yang terjadi contohnya berupa kendala biaya, atau waktu.

b. Mengidentifikasi fungsi yang ada pada API layanan

   Mengidentifikasi fungsi yang ada pada API layanan penting untuk memberikan penjelasan terkait fungsi apa saja yang dilakukan oleh tiap API layanan. Pada tahap ini memerlukan diagram proses bisnis dan aplikasi data sebagai acuan untuk mengidentifikasi fungsi.

c. Mengidentifikasi proses hubungan antar API layanan

   Mengidentifikasi proses hubungan antar API layanan dilakukan untuk menyatukan kembali fungsi dari proses bisnis yang terpecah pada saat pengidentifikasian sub domain.

d. Mengidentifikasi format komunikasi yang ada pada API layanan

   Menetukan format komunikasi diperlukan untuk membuat hubungan antar API layanan agar proses bisnis dapat berjalan.

e. Mengidentifikasi kebutuhan pola saga yang ada pada API layanan

   Dalam teknologi *microservices* terdapat pola saga yang membantu agar basis data tetap konsisten. Pada proses transaksi data antar API layanan terdapat banyak data yang terdistribusi. untuk itu cara menyelesaikan masalah tersebut dengan pola saga.

### 3.4 Desain Skema Basis Data

Desain skema basis data dilakukan untuk mendefinisikan tiap skema basis data. MongoDB memiliki struktur yang fleksibel sehingga tidak memerlukan pemodelan khusus atau mendefinisikan tipe data pada saat mendesain skema basis data. Namun, hal ini menjadi masalah pada saat pengolahan data. Dalam pengolahan data seluruh data harus melalui

strandardisasi. Untuk itu perlunya mongoose untuk mengatasi masalah tersebut. Mongoose digunakan untuk menstandardisasi tipe data yang akan masuk ke basis data dan desain skema basis data yang diterapkan juga menggunakan Mongoose.

### 3.5 Implementasi Basis Data

Basis data dirancang berdasarkan arsitektur *enterprise* pola *vending machine* yang selanjutnya basis data dikelompokkan ulang sesuai dengan kebutuhan pengimplementasian. Pengembangan basis data menggunakan MongoDB sebagai teknologi yang berbasis dokumen.

### 3.6 Implementasi API Layanan

Pengembangan API mengimplementasikan arsitektur *enterprise* toko *online* yang dikembangkan berdasar pola *vending machine* yang sudah disesuaikan dengan kebutuhan *microservices* pada saat pengembangan. Pengembangan API menerapkan pola-pola *microservices,* yaitu: pola saga untuk menjaga konsistensi data antar layanan, serta menggunakan pola koreografi yang memiliki kelonggaran antar layanan yang membuat perkembangan lebih cepat serta konsisten dan efisien.

### 3.7 Pengujian API Layanan

Pengujian API layanan melalui dua tahap, yaitu: *unit testing*, dan juga *integration testing*. *Unit testing* merupakan tahapan untuk menguji fungsionalitas dari tiap layanan. Sedangkan, *integration testing* merupakan pengujian terhadap proses komunikasi yang terjadi antar layanan. Meskipun tahapan dalam melakukan *unit testing*, dan juga *integration testing* sama perbedaan hanya pada jenis objek yang akan diuji.

### 3.8 Implementasi Antarmuka

Antarmuka dikembangkan untuk arsitektur *enterprise* toko *online* yang dikembangkan berdasar pola *vending machine* yang telah dimodifikasi. Pengembangan antarmuka menggunakan teknologi React.js.

### 3.9 Pengujian Antarmuka

Pengujian dilakukan untuk mengetahui apakah proses bisnis berjalan dengan baik. Pengujian proses bisnis dilakukan berdasarkan sudut pandang masing-masing pengguna. Pengujian dilakukan terhadap sekaligus semua fungsionalitas pada layanan API melalui antarmuka. Apabila keluaran dari tiap fungsionalitas belum menemui harapan, maka dilakukan perbaikan dan diuji ulang sehingga tiap keluaran fungsionalitas layanan API sesuai dengan harapan.

# BAB IV
# HASIL DAN PEMBAHASAN

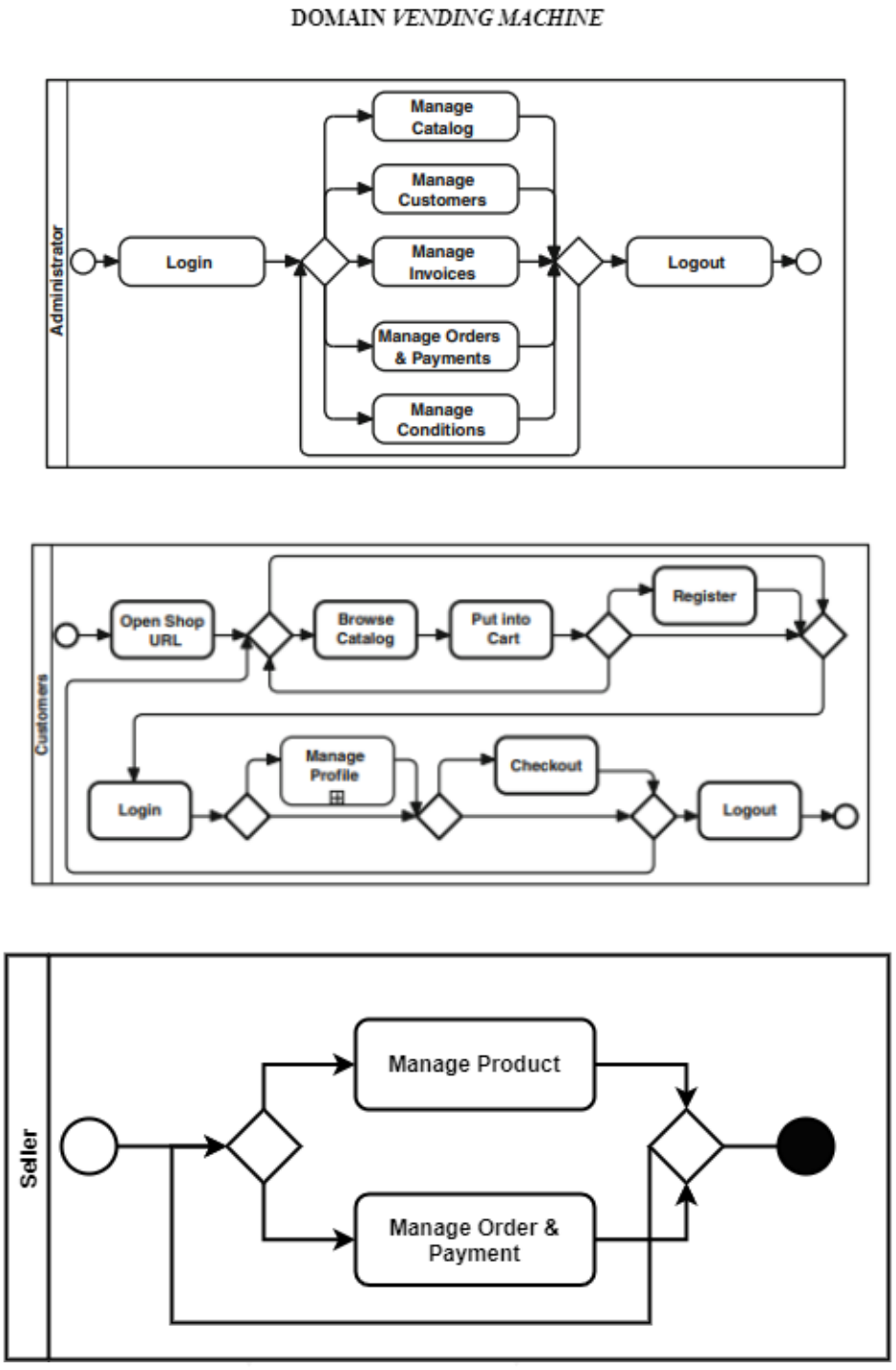
## 4.1 Dekomposisi Arsitektur *Enterprise* Pola *Vending Machine*

Tahap pertama dalam implementasi arsitektur *enterprise* pola *vending machine* pada teknologi *microsevices* adalah dekomposisi. Ada 3 tahap dalam melakukan dekomposisi, yaitu: identifikasi domain, identifikasi sub domain, dan pengujian hasil identifikasi sub domain.

### 4.1.1 Identifikasi Domain

Dalam tahapan pertama identifikasi domain dilakukan pendetailan tiap bidang keahlian dalam *enterprise*. Contohnya pada *enterprise* yang ada pada toko *online* terdapat beberapa domain seperti domain bidang pemasaran (*vending machine*), bidang finansial, dan juga bidang sumber daya manusia. Untuk mengidentifikasi tiap domain memerlukan seluruh aktivitas proses bisnis yang digunakan dalam *enterprise* tersebut.

Pada gambar 4.1 terdapat keseluruhan proses bisnis tiap aktor mulai dari *administrator, seller*, dan juga *customer*. Dalam keseluruhan proses bisnis ini terdapat satu domain pada kasus arsitektur *enterprise* pola *vending machine* yaitu: domain pemasaran (*vending machine*).

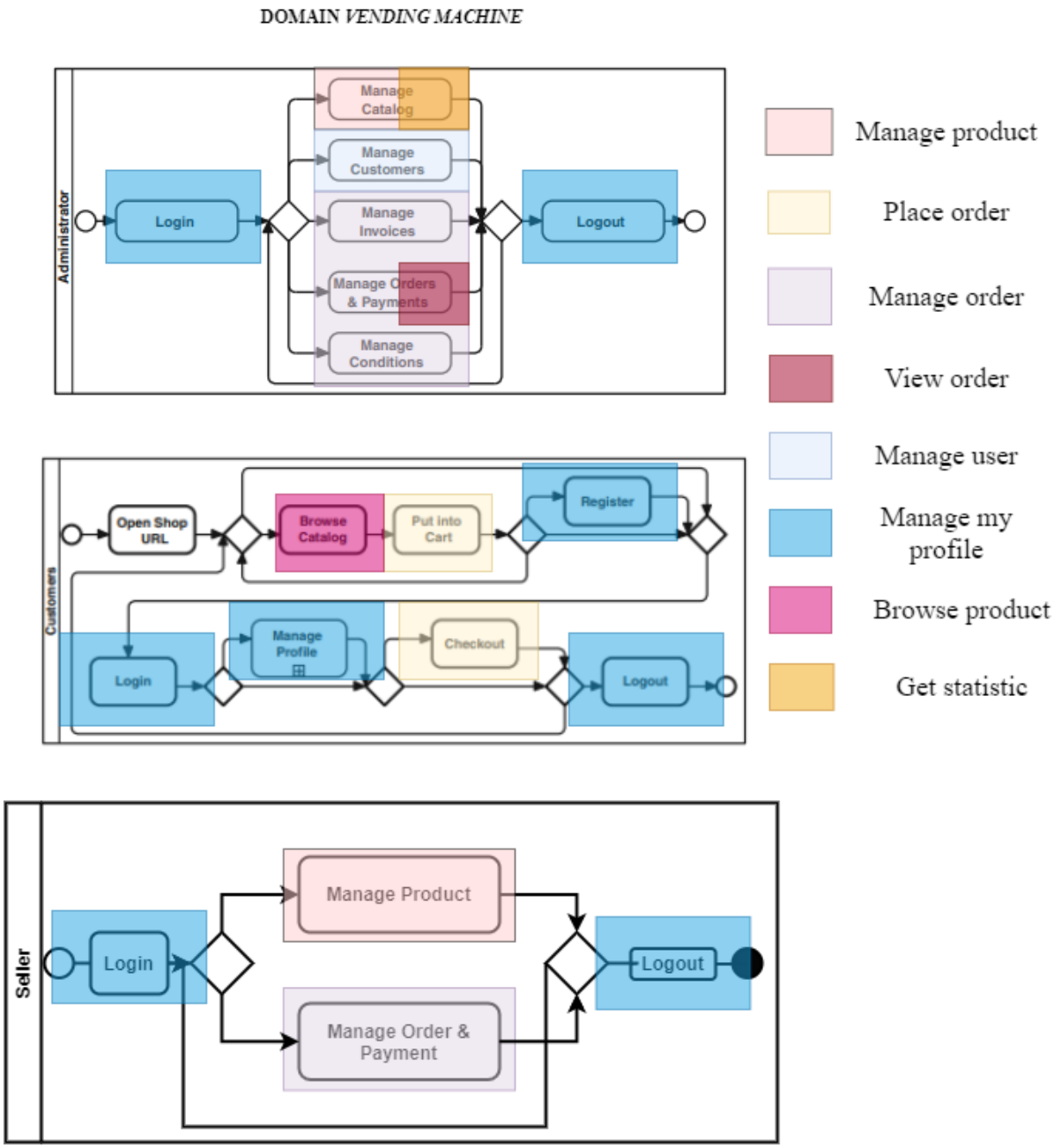



Gambar 4.1 Proses bisnis pola *vending machine* (Perroud & Inversini, 2013)

### 4.1.2 Identifikasi Sub Domain

Setelah domain diidentifikasi, dilakukan proses identifikasi bagian sub domain. Sub domain merupakan detail keahlian pada tiap domain. Dalam proses identifikasi sub domain memerlukan pemahaman terhadap proses bisnis. Identifikasi dilakukan berdasarkan struktur

organisasi, dan juga sub bidang keahliannya masing-masing (Richardson, 2018) Seperti pada gambar 4.2 yang terdapat 8 sub domain.




Gambar 4.2 Hasil dekomposisi sub domain dari domain *vending machine*

Gambar 4.2 memuat keseluruhan aktivitias pola *vending machine* dari berbagai sudut pandang dan juga tiap-tiap sub domain yang telah terbagi. Pada bagian aktivitas *manage product* memiliki sub domain tersendiri yaitu *manage product*. Kemudian aktivitas *put into cart*, dan *checkout* masuk kedalan sub domain *place order* yang digunakan untuk menyimpan

keseluruhan informasi pemesanan. *Manage invoice, manage order & payment* dan juga *manage condition* masuk kedalam sub domain *manage order* yang fungisnya mengelola keseluruhan produk dan juga *manage order* memiliki sub domain lain yaitu *view order* yang fungsinya melihat keseluruhan pemesanan.

Kemudian *manage user* merupakan sub domain dari *manage customer* untuk mengelola keseluruhan pengguna dari semua sudut pandang aktor yaitu, *seller, admin,* dan juga *customer.* Sub domain *manage my profile* memuat aktivitas mulai dari *login, logout, register,* dan juga *manage profile* sub domain ini memuat aktivitas untuk mengelola informasi akun maupun membuat akun. *Browse product* memiliki sub domain sendiri untuk mencari dan melihat keseluruhan produk melalui kategori produk. Sub domain terakhir terdapat dalam aktivitas *manage product* yang juga memuat sub domain *get statistic* untuk melihat ulasan produk maupun penjualan produk. Serta terjadi penambahan proses bisnis *seller* yang berfungsi mengelola sendiri produk dan pemesanan produk yang telah dibuat *seller*.

### 4.1.3 Pengujian Hasil Identifikasi Tiap Sub Domain

Pengujian dilakukan terhadap semua sub domain yang telah diidentifikasi. Pengujian pada tiap sub domain menggunakan *Single responsibility principle* (SRP). Berikut hasil dari pengujian tiap sub domain terdapat pada tabel 4.1.

Tabel 4.1 Verifikasi *Single responsibility principle* (SRP)

| No. | Sub Domain | Tanggung Jawab | Kesimpulan |
|---|---|---|---|
| 1. | *Manage product* | Manajemen keseluruhan produk | Valid |
| 2. | *Place order* | Menempatkan keseluruhan informasi pesanan | Valid |
| 3. | *Manage order* | Manajemen keseluruhan pesanan | Valid |
| 4. | *View order* | Melihat keseluruhan order | Valid |
| 5. | *Manage user* | Manajemen keseluruhan pengguna | Valid |
| 6. | *Manage my profile* | Manajemen profil pengguna | Valid |
| 7. | *Browse product* | Mencari produk sesuai kategori | Valid |
| 8. | *Get statistic* | Melihat keseluruhan penjualan dan ulasan produk | Valid |

## 4.2 Desain API Layanan

Setelah mendekomposisi dilakukan analisis kebutuhan terhadap kebutuhan dari tiap layanan API. Terdapat lima proses tahapan yang dilakukan, yaitu: Mengidentifikasi API layanan, Mengidentifikasi fungsi yang ada pada API layanan, Mengidentifikasi proses hubungan antar API layanan, Mengidentifikasi format komunikasi yang ada pada API layanan, Mengidentifikasi kebutuhan pola saga yang ada pada API layanan.

### 4.2.1 Mengidentifikasi API layanan

Pada aktivitas pertama yang dilakukan untuk mendesain API layanan adalah mengidentifikasi sub domain mana saja yang akan diimplementasikan menjadi API layanan. Berdasarkan hasil identifikasi 8 sub domain tersebut akan diimplementasikan. Berikut hasil identifikasi API layanan dapat dilihat pada tabel 4.2.

Tabel 4.2 Hasil identifikasi API layanan

| No. | Sub domain | Status |
|---|---|---|
| 1. | *Manage product* | Diimplementasikan sebagai API layanan |
| 2. | *Place order* | Diimplementasikan sebagai API layanan |
| 3. | *Manage order* | Diimplementasikan sebagai API layanan |
| 4. | *View order* | Diimplementasikan sebagai API layanan |
| 5. | *Manage user* | Diimplementasikan sebagai API layanan |
| 6. | *Manage my profile* | Diimplementasikan sebagai API layanan |
| 7. | *Browse product* | Diimplementasikan sebagai API layanan |
| 8. | *Get statistic* | Diimplementasikan sebagai API layanan |

### 4.2.2 Mengidentifikasi Fungsi yang ada pada API layanan

Aktivitas selanjutnya digunakan untuk menentukan fungsi dan kebutuhan dari tiap API layanan berikut fungsi tiap API layanan ditunjukkan pada tabel 4.3.

Tabel 4.3 Hasil identifikasi fungsi tiap API layanan.

| No. | Layanan | Fungsi | Keterangan |
|---|---|---|---|
| 1. | *Manage product* | *Create product* | Menambahkan produk |
| | | *View all product* | Melihat keseluruhan produk |
| | | *Delete product* | Menghapus produk |
| | | *Edit info product* | Mengubah informasi produk |
| 2. | *Place order* | *Create order* | Membuat pesanan |
| 3. | *Manage order* | *Update status payment* | Mengubah status pembayaran |
| | | *Update deliver order status* | Mengubah status pengiriman pesanan |
| | | *Create payment* | Membuat pembayaran |
| 4. | *View order* | *View all order* | Melihat keseluruhan pemesanan |
| | | *Delete order* | Menghapus pesanan |
| 5. | *Manage user* | *View all user* | Melihat keseluruhan pengguna |
| | | *Update user status* | Mengubah status pengguna |
| | | *Delete user* | Menghapus pengguna |
| 6. | *Manage my profile* | *Sign in* | Bergabung ke dalam aplikasi menggunakan akun yang telah dibuat |
| | | *Register* | Membuat akun |
| | | *View info user* | Melihat informasi pengguna |
| | | *Edit info user* | Mengubah informasi pengguna |
| 7. | *Browse product* | *Filter product* | Mencari product berdasarkan kategori |
| 8. | *Get statistic* | *Review product* | Ulasan produk |
| | | *View sales* | Melihat jumlah penjualan produk |

### 4.2.3 Mengidentifikasi proses hubungan antar API layanan

Aktivitas ketiga digunakan untuk menentukan hubungan API layanan antara satu sama lain. Proses ini dilakukan untuk menghubungkan kembali proses bisnis yang telah terpecah. Berikut identifikasi yang ditunjukkan pada tabel 4.4.

Tabel 4.4 Hasil identifikasi hubungan antar API layanan

| No. | Layanan | Fungsi | Hubungan fungsi |
|---|---|---|---|
| 1. | *Manage product* | *Create product* | *Get info seller* |
| | | *View all product* | *Get info seller* |
| | | *Delete product* | - |
| | | *Edit info product* | - |
| 2. | *Place order* | *Create order* | *Edit product* |
| | | | *Get info seller* |
| | | | *Get info user* |
| | | | *Get info product* |
| 3. | *Manage order* | *Update status payment* | - |
| | | *Update deliver order status* | - |
| | | *Create payment* | *Get info product* |
| 4. | *View order* | *View all order* | *Get info user* |
| | | | *Get info product* |
| | | *Delete order* | - |
| 5. | *Manage user* | *View all user* | - |
| | | *Update user status* | - |
| | | *Delete user* | *Edit order* |
| 6. | *Manage my profile* | *Sign in* | - |
| | | *Register* | - |
| | | *View info user* | - |
| | | *Edit info user* | - |
| 7. | *Browse product* | *Filter product* | - |
| 8. | *Get statistic* | *Review product* | *Get info user* |
| | | *View sales* | - |

Pada tabel 4.4 terdapat beberapa hubungan fungsi. Pada layanan *manage product* fungsi *create*, dan *view all product* memiliki hubungan fungsi *get info seller* dan juga untuk mendapatkan info terkait produk tersebut di kelola oleh *seller* siapa. Kemudian layanan *place order* yang memiliki fungsi *create order* memiliki hubungan fungsi, yaitu: *edit product, get*

*info seller*, *get info user*, dan *get info product*. Fungsi *edit product* untuk merubah jumlah produk, *get info seller* untuk mengirimkan informasi terkait *seller* siapa yg mengelola produk tersebut, *get info user* untuk mengirimkan informasi siapa pengguna yang membeli produk tersebut serta *get info product* untuk mengirimkan informasi detail dari produk yang dipesan.

Layanan *manage order* yang memiliki fungsi *create payment* memiliki hubungan fungsi dengan *get info product* untuk mengetahui jumlah harga produk yang dipesan. Layanan *view order* yang memiliki fungsi *view all order* memiliki hubungan dengan fungsi *get info user*, dan *get info product* untuk mengetahui siapa saja pegguna yang memesan dan juga produk apa saja yang di pesan. Sedangkan *delete order* yang ada pada layanan *view order* memiliki hubungan dengan fungsi *edit product* untuk merubah jumlah produk.

Kemudian layanan *manage user* yang memiliki fungsi *delete user* mempunyai hubungan dengan fungsi *edit order* untuk menghilang pesanan produk yang telah di pesan oleh pegguna yang dihapus. Terakhir *get statistic* yang mempunyai fungsi *review product* yang memiliki hubungan fungsi dengan *get info user* untuk mengetahui siapa pengguna yang memberikan ulasan terhadapat produk tersebut.

### 4.2.4 Mengidentifikasi Format Komunikasi yang ada pada API Layanan

Dalam aktivitas ini terdapat 2 hal yang harus diidentifikasi terhadap format komunikasi, yaitu: tipe komunikasi dan juga format pesan. Tipe komunikasi meggunakan *synchronous* dan format pesan meggunakan basis *text* dengan format JSON.

### 4.2.5 Mengidentifikasi kebutuhan pola saga yang ada pada API layanan

Pada implementasi arsitektur *enterprise* pola *vending machine* terdapat penggunaan pola saga yang ditujukan untuk memberikan solusi kepada sebuah layanan yang mengirimkan data sekaligus ke dua basis data. Pola saga digunakan untuk membuat data antar dua basis data tersebut tetap konsisten. Apabila terjadi masalah pada saat pengiriman pada salah satu basis data maka basis data satunya tidak akan menyimpan data untuk menjaga data tersebut tetap konsisten ataupun isi data yang dikirimkan dari layanan tetap sama di kedua basis data. Nantinya pola saga ini akan diterapkan pada basis data *order* dan *product*.

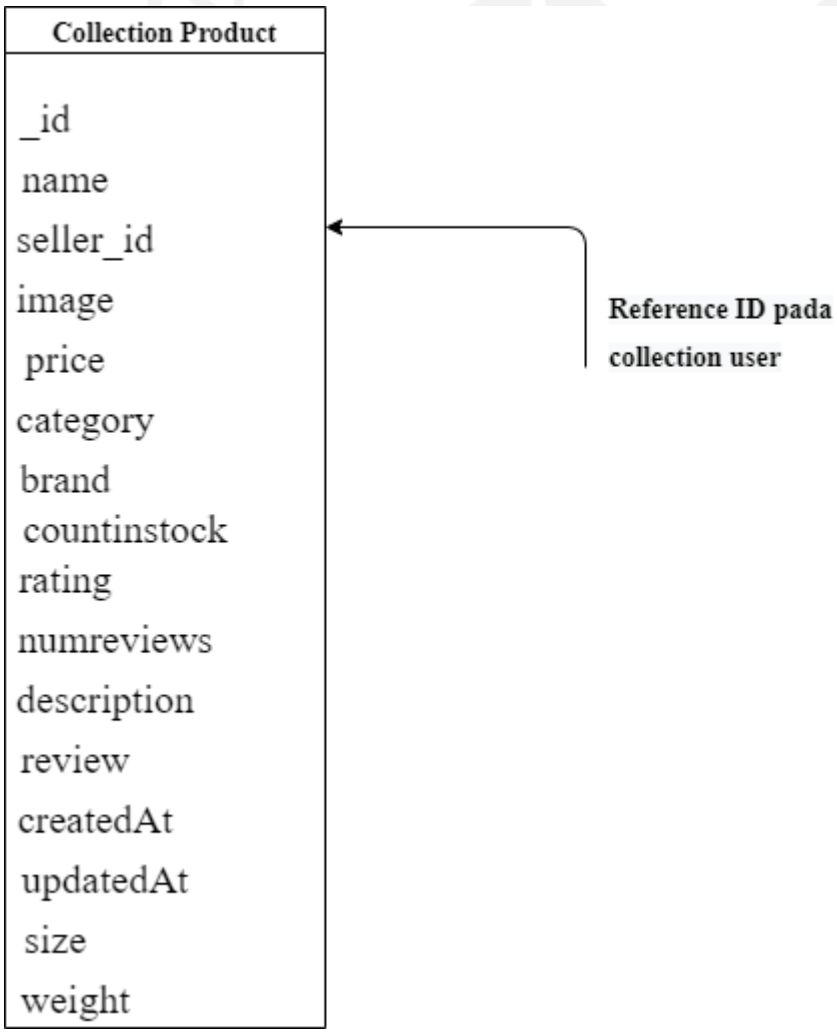
## 4.3 Desain Skema Basis Data

Langkah selanjutnya adalah mendesain dan menentukan kebutuhan pada tiap API layanan. Terdapat 3 basis data yang mencakup 8 layanan. Basis data *product* mencakup layanan

*manage product, browse product*, dan *get statistic*. Basis data *order* mencakup *layanan place order, manage order,* dan *view order*. Basis data *user* mencakup layanan *manage user*, dan *manage my profile*.

### 4.3.1 Skema Basis Data *Product*

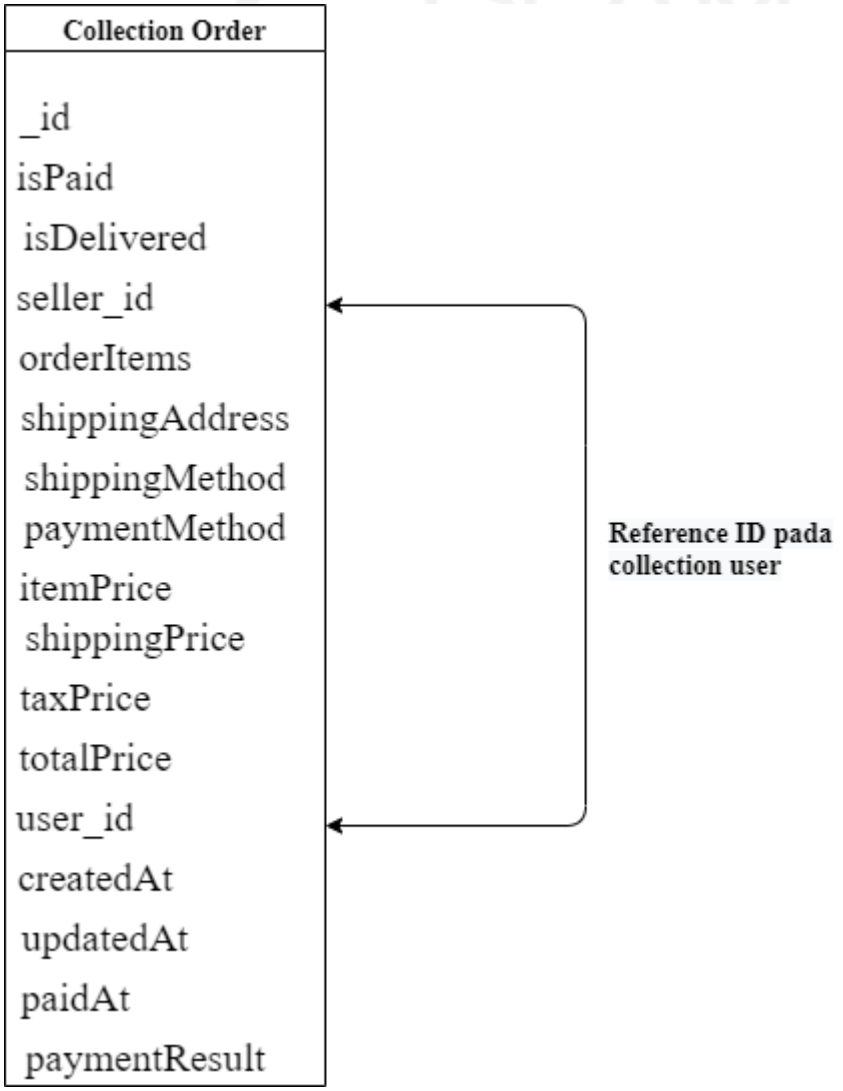
Dalam skema basis data yang dapat dilihat pada gambar 4.3 terdapat 16 atribut data yang ada pada *collection product*, yakni: _id, *name*, *seller*_id, *image*, *price*, *category*, *brand*, *countinstock*, *rating*, *numreviews*, *description*, *review*, *createdAt*, *updatedAt*, *size* dan juga *weight*. Atribur _id digunakan sebagai *primary key* dari *collection product* sedangkan *seller*_id merupakan *foreign key* dari basis data *user*.




Gambar 4.3 Diagram relasi antar *collection* pada basis data *product*

### 4.3.2 Skema Basis Data *Order*

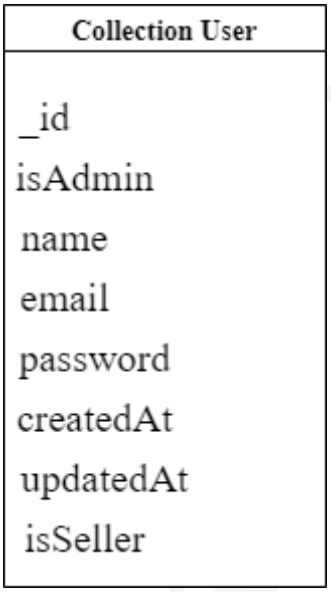
Skema basis data order yang dapat dilihat pada gambar 4.4 memiliki 17 atribut data yang dimiliki pada *collection order*, yakni: _id, *isPaid, isDelivered, seller*_id, *orderItems, shippingAddress, shippingMethod, paymentMethod, itemPrice, shippingPrice, taxPrice, totalPrice, user*_id, *createdAt, updatedAt, paidAt*, dan juga *paymentResult*. Atribut _id yang ada pada collection *order* merupakan *primary key*. Sedangkan *seller*_id dan juga *user*_id merupakan *foreign key* dari basis data *user*.




Gambar 4.4 Diagram relasi antar *collection* pada basis data *order*

### 4.3.3 Skema Basis Data *User*

Skema basis data *user* yang dapat dilihat pada gambar 4.5 *collection user* memiliki 8 atribut, yakni: _id, *isAdmin, name, email, password, createdAt, updatedAt*, dan juga *isSeller*. Atribut _id yang ada pada *collection user* digunakan sebagai *primary key*.



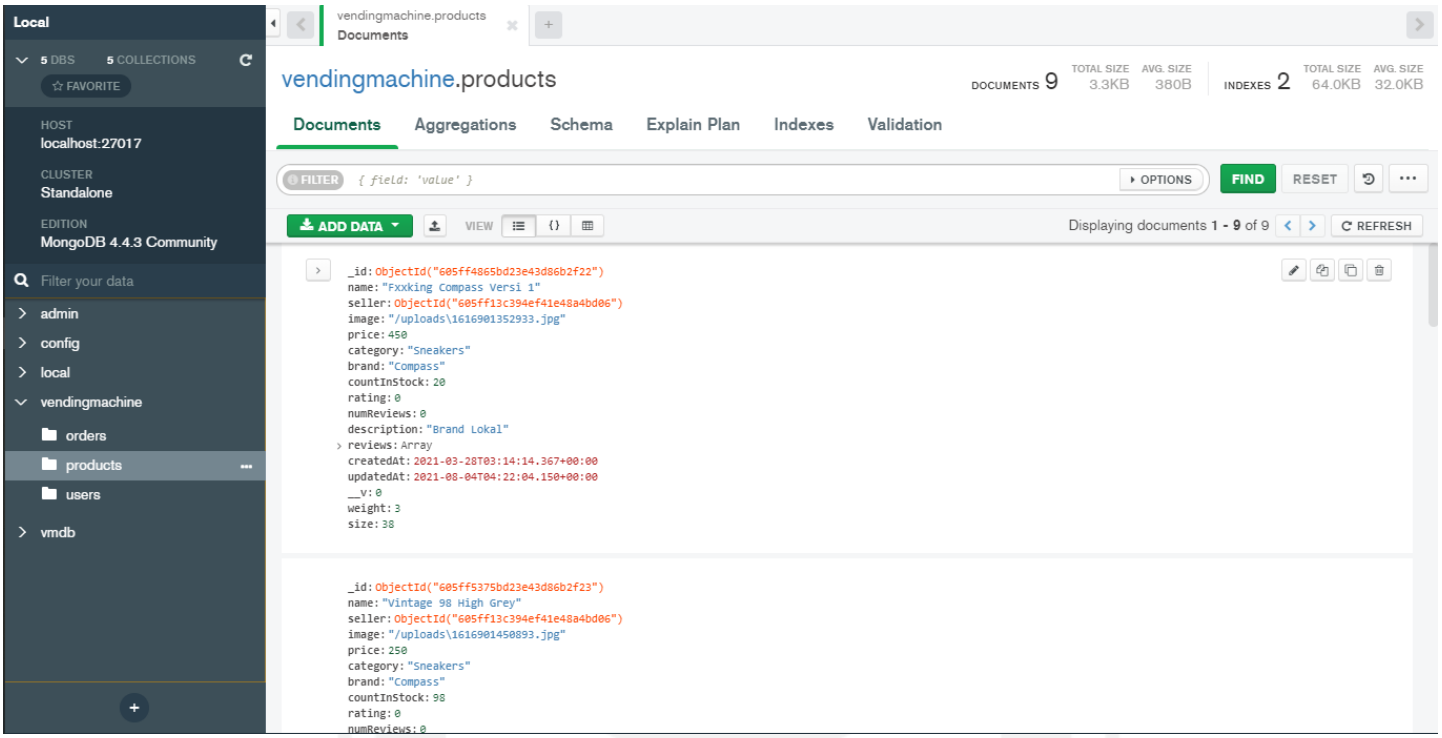

Gambar 4.5 Diagram relasi antar *collection* pada basis data *user*

## 4.4 Implementasi Basis Data

Langkah selanjutnya adalah mengimplementasikan basis data yang telah dianalisa kebutuhan dari masing-masing basis data. Pada tahap ini hanya perlu membuat basis data beserta tipe data yang diperlukan.

### 4.4.1 Implementasi Basis Data *Product*

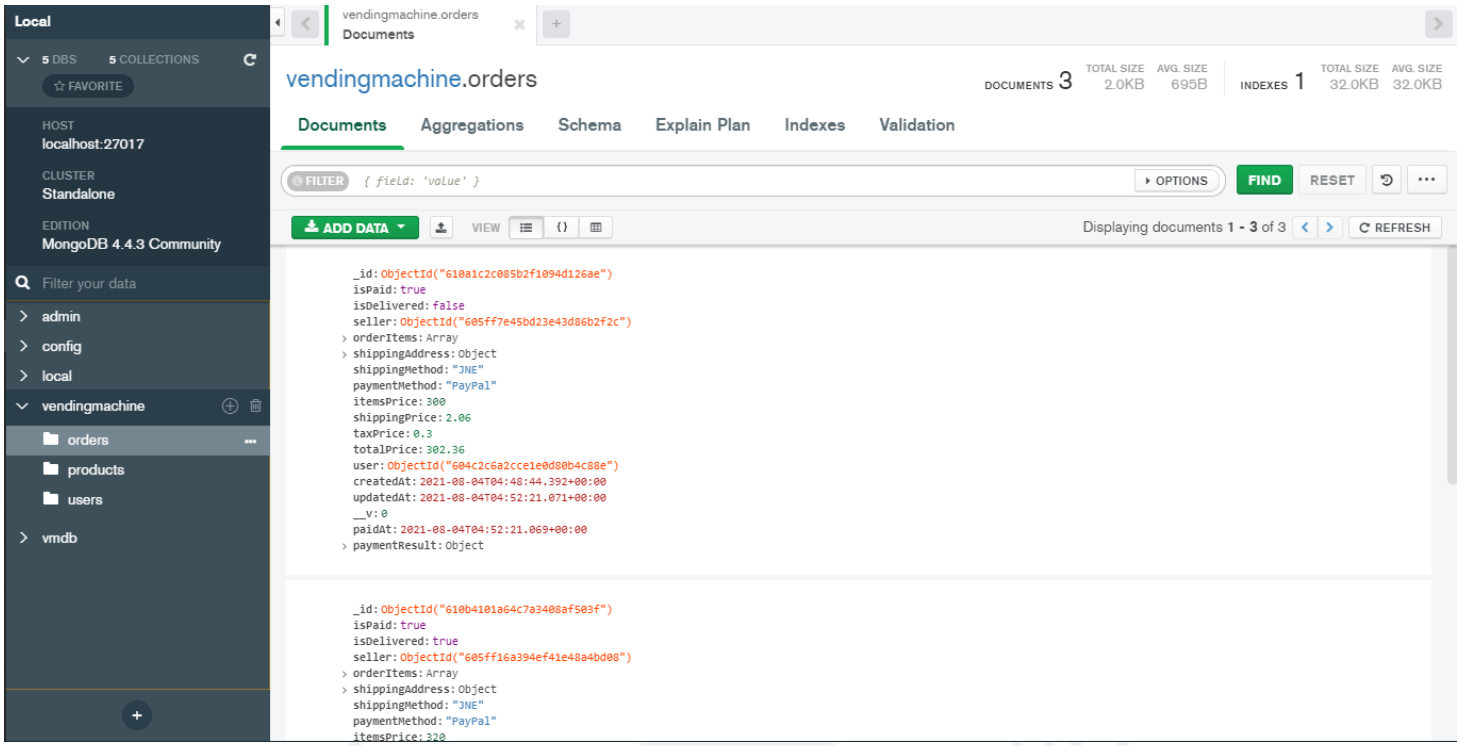
Hasil implementasi dari basis data *product* dapat terlihat pada gambar 4.6 dengan menggunakan mongoDB.




Gambar 4.6 Hasil implentasi basis data *product*

### 4.4.2 Implementasi Basis Data *Order*

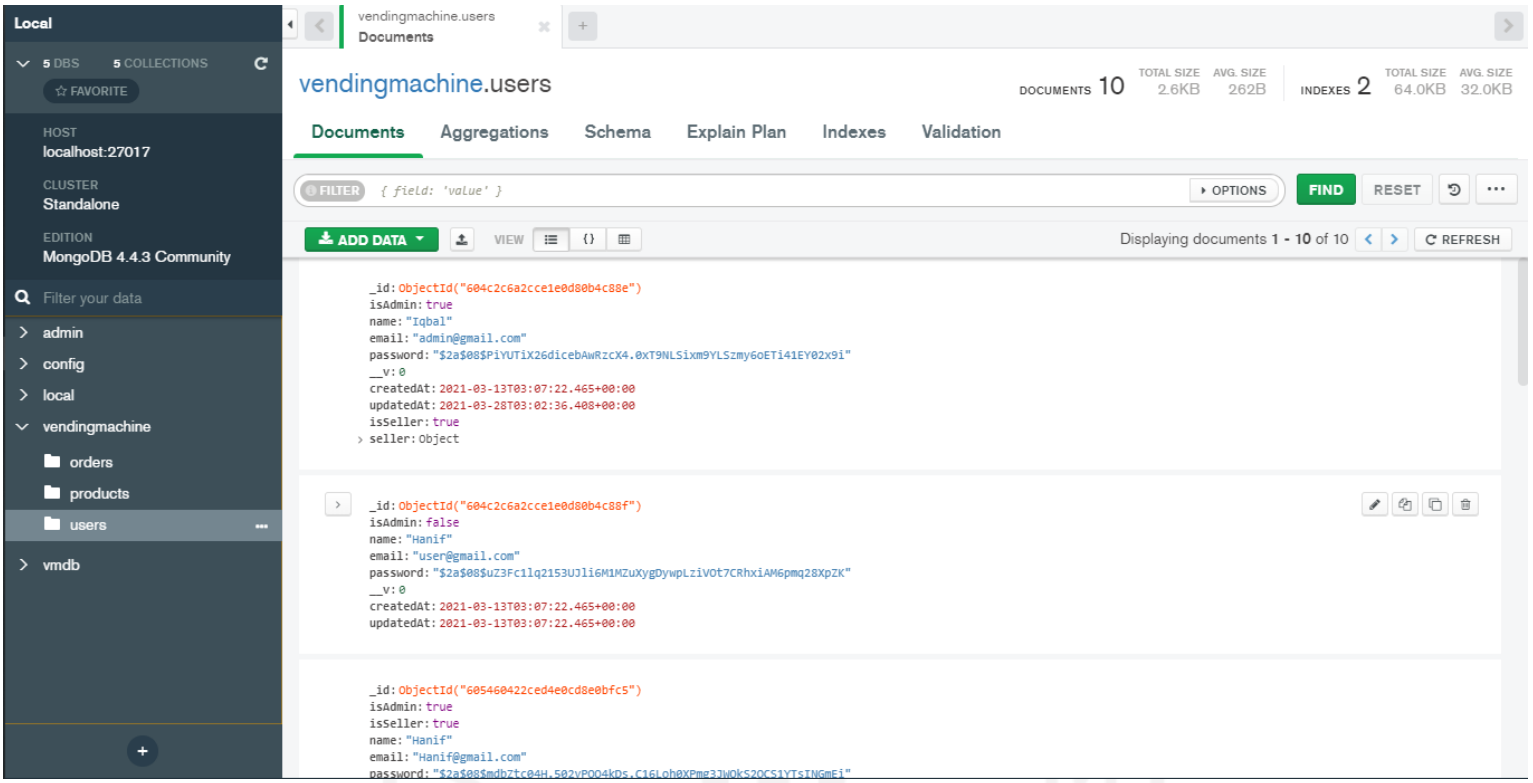
Hasil implementasi dari basis data *order* dapat terlihat pada gambar 4.7 dengan menggunakan mongoDB.




Gambar 4.7 Hasil implentasi basis data *order*

### 4.4.3 Implementasi Basis Data User

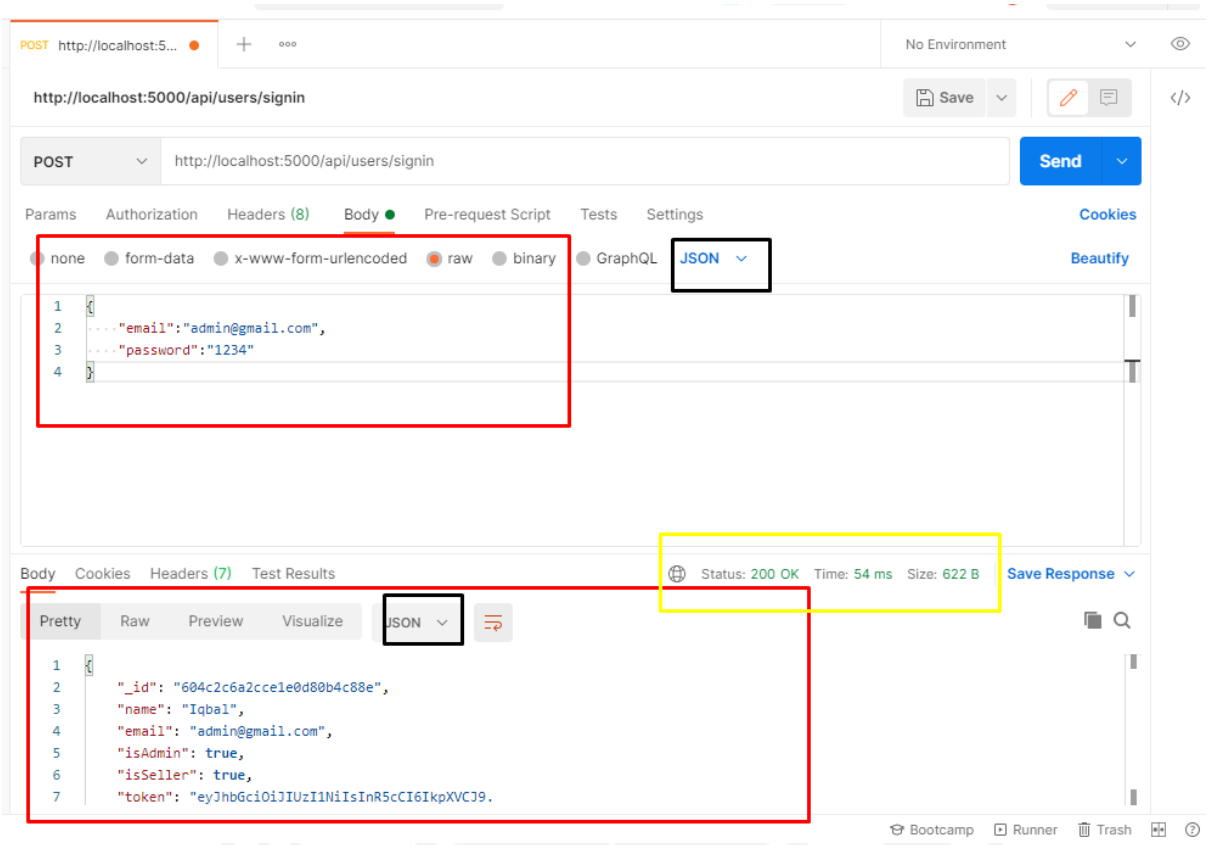
Hasil implementasi dari basis data *user* dapat terlihat pada gambar 4.7 dengan menggunakan mongoDB.




Gambar 4.8 Hasil implentasi basis data *user*

### 4.4.4 Implementasi format komunikasi yang ada pada API layanan

Gambar 4.9 merupakan proses komunikasi dan bentuk format komunikasi API layanan pada layanan *user sign in*. komunikasi *synchronous* dapat terlihat pada *request* dan *response* yang diberikan secara berurutan. Response dapat terlihat pada bagian kotak kuning yang diterima pada waktu 52*ms* pada saat pesan di *request*. Serta format penulisan pada *request* dan *response* ditunjukkan pada kotak merah, format penulisan yang berwarna merah merupakan *key* sedangkan warna biru merupakan *value*.




Gambar 4.9 Format komunikasi API layanan

### 4.4.5 Implementasi kebutuhan pola saga yang ada pada API layanan

Pada Gambar 4.10 merupakan kode yang digunakan untuk mengimplementasikan pola saga. Kode *try* merupakan kode utama dalam menjalankan pengiriman data, apabila terjadi kesalahan atau masalah pada kode *try* maka kode *catch* akan menjalankan tugasnya.

```
    try {
    const product = await Axios.post(
      `/api/products/${productId}`,
      {
        countInStock = -1,
      }
    );
} catch (error) {
  Axios.delete(`/api/orders/${data.order._id}`, {
    headers: { Authorization: `Bearer ${userInfo.token}` },
});

throw new Error("error")
}
```

Gambar 4.10 Contoh kode pola saga

## 4.5 Implementasi API Layanan

Setelah melakukan implementasi pada basis data kemudian dilakukan implementasi pada API layanan sesuai dengan Analisa kebutuhan dari tiap API layanan.

### 4.5.1 Hasil *Endpoint* pada Implementasi API Layanan

Tabel 4.5 menunjukkan hasil *endpoint* yang dapat diakses beserta *method* yang digunakan pada tiap layanan dan tiap fungsi layanan.

Tabel 4.5 Hasil *endpoint* pada implementasi API layanan

| No. | Layanan | Fungsi | *Method* | *Endpoint* |
|---|---|---|---|---|
| 1. | *Manage Product* | *Create product* | *POST* | http://localhost:3000/product/:id/edit |
| | | *View all product* | *GET* | http://localhost:3000/productlist |
| | | *Delete product* | *DELETE* | http://localhost:3000/productlist |
| | | *Edit info product* | *PUT* | http://localhost:3000/product/:id/edit |
| 2. | *Place order* | *Create order* | *POST* | http://localhost:3000/placeorder |
| 3. | *Manage order* | *Update status payment* | *PUT* | http://localhost:3000/order/:id |
| | | *Update deliver order status* | *PUT* | http://localhost:3000/order/:id |

| No. | Layanan | Fungsi | *Method* | *Endpoint* |
|---|---|---|---|---|
| | | *Create payment* | *POST* | http://localhost:3000/order/:id |
| 4. | *View order* | *View all order* | *GET* | http://localhost:3000/orderlist |
| | | *Delete order* | *DELETE* | http://localhost:3000/orderlist |
| 5. | *Manage user* | *View all user* | *GET* | http://localhost:3000/userlist |
| | | *Update user status* | *PUT* | http://localhost:3000/user/:id/edit |
| | | *Delete user* | *DELETE* | http://localhost:3000/userlist |
| 6. | *Manage my profile* | *Sign in* | *GET* | http://localhost:3000/signin |
| | | *Register* | *POST* | http://localhost:3000/register?redirect=/ |
| | | *View info user* | *GET* | http://localhost:3000/userlist |
| | | *Edit info user* | *PUT* | http://localhost:3000/user/:id/edit |
| 7. | *Browse product* | *Filter product* | *GET* | http://localhost:3000/search/category/Sneakers |
| 8. | *Get statistic* | *Review product* | *POST* | http://localhost:3000/product/:id |
| | | *View sales* | *GET* | http://localhost:3000/dashboard |

## 4.6 Pengujian API Layanan

Setelah melakukan implementasi pada API layanan kemudian API layanan diuji apakah *response* dari tiap *endpoint* sesuai dengan harapan yang dinginkan.

### 4.6.1 Pengujian API Layanan *Manage Product*

Tabel 4.6 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *manage product*.

Tabel 4.6 Pengujian API layanan *manage product*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *create product*** | | | | |
| 1. | *Unit* | Akses *endpoint create product* menggunakan parameter data yang sesuai | Data *product* pada basis data *product* bertambah | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 2. | *Unit* | Akses *endpoint create product* menggunakan parameter data yang **tidak** sesuai | *Error* dan data pada basis data *product* tidak bertambah | Valid |
| 3. | *Integration* | Akses *endpoint create product* menggunakan parameter data yang sesuai, serta menggunakan akun *seller* | Data *product* beserta informasi *seller* pada basis data *product* bertambah | Valid |
| 4. | *Integration* | Akses *endpoint create product* menggunakan parameter data yang sesuai, serta menggunakan akun *admin* | *Error* dan Data *product* beserta informasi *seller* pada basis data *product* tidak bertambah | |
| **Fungsi *view all product*** | | | | |
| 5. | *Unit* | Akses *endpoint view all product* | Seluruh data *product* ditampilkan termasuk informasi *seller* | Valid |
| **Fungsi *delete product*** | | | | |
| 6. | *Unit* | Akses *endpoint delete product* menggunakan parameter data yang sesuai | Data *product* pada basis data *product* terhapus | Valid |
| 7. | *Unit* | Akses *endpoint delete product* menggunakan parameter data yang **tidak** sesuai | *Error* dan data *product* pada basis data *product* **tidak** terhapus | Valid |
| **Fungsi *Edit info product*** | | | | |
| 8. | *Unit* | Akses *endpoint edit info product* menggunakan parameter data yang sesuai | Data pada basis data *product* berubah sesuai | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| | | | dengan parameter yang telah dimasukkan | |
| 9. | *Unit* | Akses *endpoint edit info product* menggunakan parameter data yang **tidak** sesuai | *Error* dan data pada basis data *product* **tidak** berubah | Valid |

### 4.6.2 Pengujian API Layanan *Place Order*

Tabel 4.7 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *place order*.

Tabel 4.7 Pengujian API layanan *place order*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *create order*** | | | | |
| 1. | *Unit* | Akses *endpoint create order* menggunakan parameter data yang sesuai | Data infromasi *order* pada basis data *order* bertambah | Valid |
| 2. | *Unit* | Akses *endpoint create order* menggunakan parameter data yang **tidak** sesuai | Data informasi *order* pada basis data *order* **tidak** bertambah | Valid |
| 3. | *Integration* | Akses *endpoint create order* menggunakan parameter data yang sesuai | Data informasi jumlah produk yang dimiliki setelah pemesanan berkurang | Valid |
| 4. | *Integration* | Akses *endpoint create order* menggunakan parameter data yang sesuai | Data *order* beserta informasi *seller* pada basis data *order* bertambah | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 5. | *Integration* | Akses *endpoint create order* menggunakan parameter data yang sesuai | Data *order* beserta informasi *user* pemesan pada basis data *order* bertambah | Valid |
| 6. | *Integration* | Akses *endpoint create order* menggunakan parameter data yang sesuai | Data *order* beserta informasi *product* pada basis data *order* bertambah | Valid |

### 4.6.3 Pengujian API Layanan *Manage Order*

Tabel 4.8 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *manage order*.

Tabel 4.8 Pengujian API layanan *manage order*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *update status payment*** | | | | |
| 1. | *Unit* | Akses *endpoint update status payment* menggunakan parameter data yang sesuai | Data status *payment* pada basis data *order* berubah menjadi sukses dibayarkan | Valid |
| 2. | *Unit* | Akses *endpoint update status payment* apabila belum dibayarkan | Data status *payment* pada basis data *order* **tidak** berubah menjadi sukses dibayarkan | Valid |
| **Fungsi *update deliver order status*** | | | | |
| 3. | *Unit* | Akses *endpoint update deliver order status* apabila telah sukses dibayarkan | Data *status deliver order* pada basis data *order* berubah menjadi sukses dikirimkan | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 4. | *Unit* | Akses *endpoint update deliver order status* apabila **belum** sukses dibayarkan | Data *status deliver order* pada basis data *order* belum bisa dirimkan | Valid |
| **Fungsi *create payment*** | | | | |
| 5. | *Unit* | Akses *endpoint create payment* terbentuk apabila telah melakukan place order | Data *payment* pada basis data *order* ditambahkan | Valid |
| 6. | *integration* | Akses *endpoint create payment* terbentuk apabila telah melakukan *place order* | Data *payment* pada basis data *order* ditambahkan beserta informasi produk | Valid |

### 4.6.4 Pengujian API Layanan *View Order*

Tabel 4.9 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *view order*.

Tabel 4.9 Pengujian API layanan *view order*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *view all order*** | | | | |
| 1. | *Unit* | Akses *endpoint view all order* | Seluruh data *order* ditampilkan | Valid |
| 2. | *Integration* | Akses *endpoint view all order* | Seluruh data *order* ditampilkan beserta informasi *user* yang memesan | Valid |
| 3. | *Integration* | Akses *endpoint view all order* | Seluruh data *order* ditampilkan beserta informasi produk yang dipesan | Valid |
| **Fungsi *delete order*** | | | | |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 4. | *Unit* | Akses *endpoint delete order* menggunakan parameter data yang sesuai | Data *order* pada basis order *user* terhapus | Valid |
| | *Unit* | Akses *endpoint delete order* menggunakan parameter data yang **tidak** sesuai | Data *order* pada basis data *order* **tidak** terhapus | Valid |

### 4.6.5 Pengujian API Layanan *Manage User*

Tabel 4.10 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *manage user*.

Tabel 4.10 Pengujian API layanan *manage user*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *view all user*** | | | | |
| 1. | *Unit* | Akses *endpoint view all user* | Seluruh data *user* ditampilkan | Valid |
| **Fungsi *update user status*** | | | | |
| 4. | *Unit* | Akses *endpoint update user status* menggunakan parameter data yang sesuai | Data *user status* pada basis data *user* berubah | Valid |
| 5. | *Unit* | Akses *endpoint update user status* menggunakan parameter data yang **tidak** sesuai | Data *user* status pada basis data *user* **tidak** berubah | Valid |
| **Fungsi *delete user*** | | | | |
| 6. | *Unit* | Akses *endpoint delete user* menggunakan parameter data yang sesuai | Data *user* pada basis data *user* terhapus | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 7. | *Unit* | Akses *endpoint delete product* menggunakan parameter data yang **tidak** sesuai | Data *product* pada basis data *product* **tidak** terhapus | Valid |
| 8. | *Integration* | Akses *endpoint delete product* menggunakan parameter data yang sesuai | Data *product* pada basis data *product* berubah apabila *seller* dihapus | Valid |

### 4.6.6 Pengujian API Layanan *Manage My Profile*

Tabel 4.11 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *manage my profile*.

Tabel 4.11 Pengujian API layanan *manage my profile*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *sign in*** | | | | |
| 1. | *Unit* | Akses *endpoint sign in* apabila telah melakukan *register* | Membaca data *user* dan membuat akun tersebut *login* | Valid |
| 2. | *Unit* | Akses *endpoint sign in* apabila **belum** melakukan *register* | Membaca data *user* dan mempersilahkan *user* untuk melakukan *register* | Valid |
| **Fungsi *register*** | | | | |
| 3. | *Unit* | Akses *endpoint register* menggunakan parameter data yang sesuai | Data *user* pada basis data *user* bertambah | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 4. | *Unit* | Akses *endpoint register* menggunakan parameter data yang **tidak** sesuai | Data *user* pada basis data *user* **tidak** bertambah | Valid |
| **Fungsi *view info user*** | | | | |
| 5. | *Unit* | Akses *endpoint view info user* | data *user* ditampilkan | Valid |
| **Fungsi *edit info user*** | | | | |
| 6. | *Unit* | Akses *endpoint edit info user* menggunakan parameter data yang sesuai | Data *user* info pada basis data *user* berubah | Valid |
| 7. | *Unit* | Akses *endpoint edit info user* menggunakan parameter data yang **tidak** sesuai | Data *user info* pada basis data *user* **tidak** berubah | Valid |

### 4.6.7 Pengujian API Layanan *Browse Product*

Tabel 4.12 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *browse product*.

Tabel 4.12 Pengujian API layanan *browse product*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *browse product*** | | | | |
| 1. | *Unit* | Akses *endpoint browse product* | Membaca data *product* yang dicari dan menampilkan produk yang diinginkan | Valid |

### 4.6.8 Pengujian API Layanan *Get Statistic*

Tabel 4.13 menunjukkan hasil pengujian dari tiap fungsi yang ada pada layanan *get statistic*.

Tabel 4.13 Pengujian API layanan *get statistic*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *review product*** | | | | |
| 1. | *Unit* | Akses *endpoint review product* menggunakan parameter data yang sesuai | Data *review product* pada basis data *product* bertambah | Valid |
| 2. | *Unit* | Akses *endpoint review product* menggunakan parameter data yang **tidak** sesuai | Data *review product* pada basis data *product* **tidak** bertambah | Valid |
| 3. | *Integration* | Akses *endpoint review product* | Menampilkan keseluruhan data *review* beserta info *user* yang melakukan *review* | Valid |
| **Fungsi *view sales*** | | | | |
| 4. | *Unit* | Akses *endpoint view sales* | Menampilkan keseluruhan data penjualan beserta info keuntungan penjualan | Valid |

## 4.7 Implementasi Antarmuka

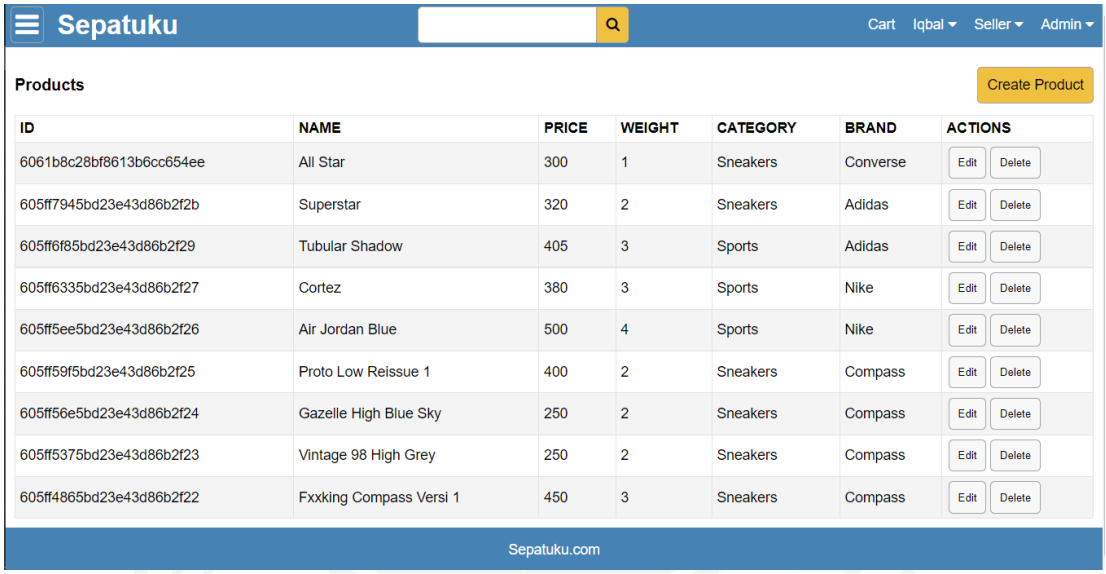
Setelah API layanan selesai di implementasikan dan juga diuji masing-masing fungsinya, maka langkah selanjutnya adalah melakukan implementasi terhadap antarmuka untuk kebutuhan tiap API layanan.

### 4.7.1 Implementasi Antarmuka *Manage Product*

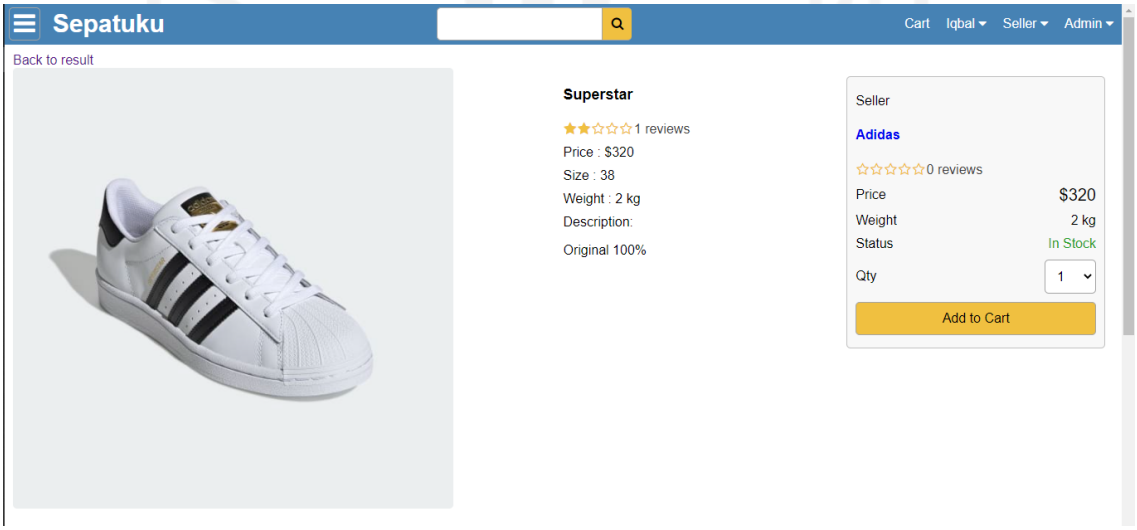
Pada bagian antarmuka *manage product* terdapat terdapat empat fungsi, yaitu: *create product, view all product, delete product, edit info product.*

## A. Fungsi *Create Product*

Pada fungsi *create product* antarmuka dibuat untuk menambahkan produk dengan cara mengisi informasi produk yang dibutuhkan seperti pada Gambar 4.11. kemudian antarmuka akan menampilkan keseluruhan informasi yang telah ditambahkan pada Gambar 4.12.



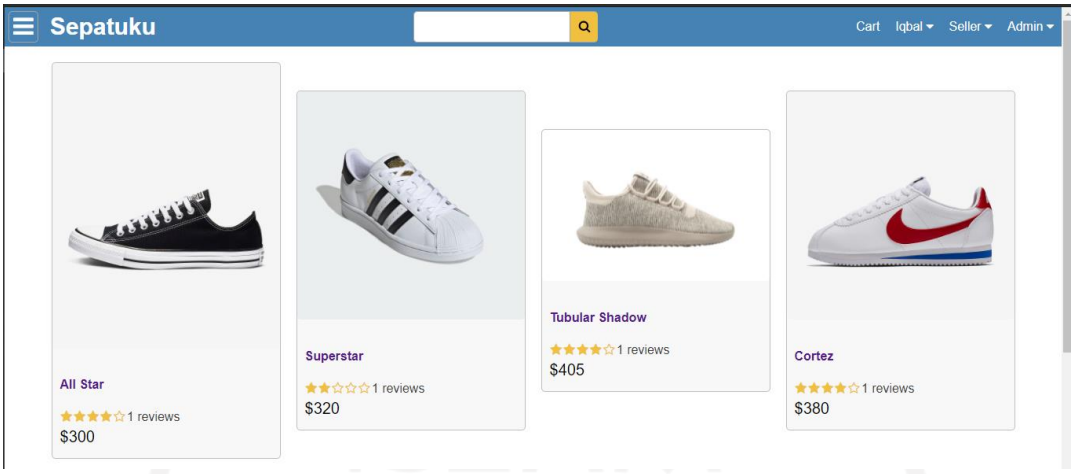

Gambar 4.11 Halaman antarmuka *manage product*




Gambar 4.12 Halaman antarmuka produk yang memunculkan informasi *seller*

## B. Fungsi *View All Product*

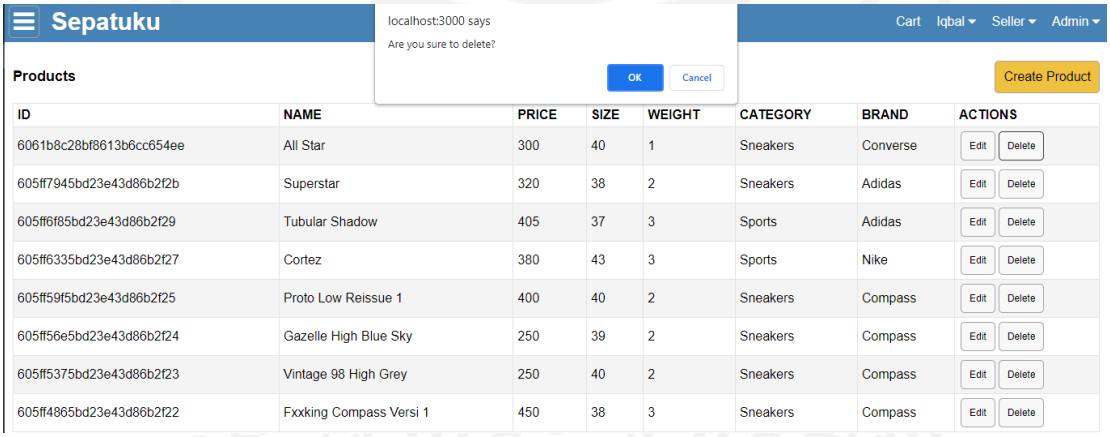
Pada bagian ini halaman antarmuka dibuat untuk menampilkan informasi keseluruhan produk mulai dari sudut pandang pengguna pada Gambar 4.13 dan juga sudut pandang *seller*.




Gambar 4.13 Halaman antarmuka *view all product*

## C. Fungsi *Delete Product*

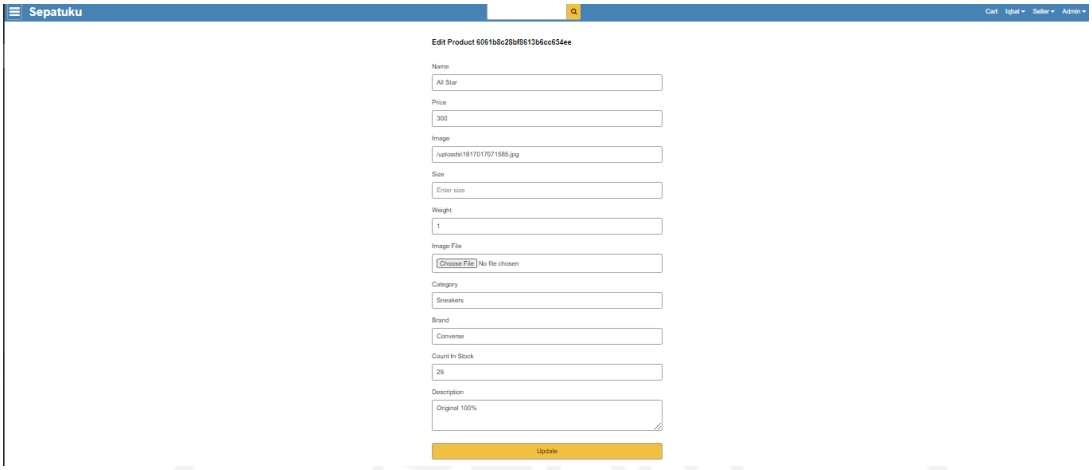
Pada bagian fungsi ini antarmuka digunakan oleh *seller* untuk melakukan penghapusan pada produk, serta sebelum produk benar-benar di hapus datanya, pada Gambar 4.14 *seller* akan menerima peringatan apakah produk benar-benar akan dihapus.




Gambar 4.14 Halaman antarmuka *delete product*

## D. Fungsi *Edit Info Product*

Fungsi antarmuka ini digunakan seller mengubah informasi dari sebuah produk seperti pada Gambar 4.15.



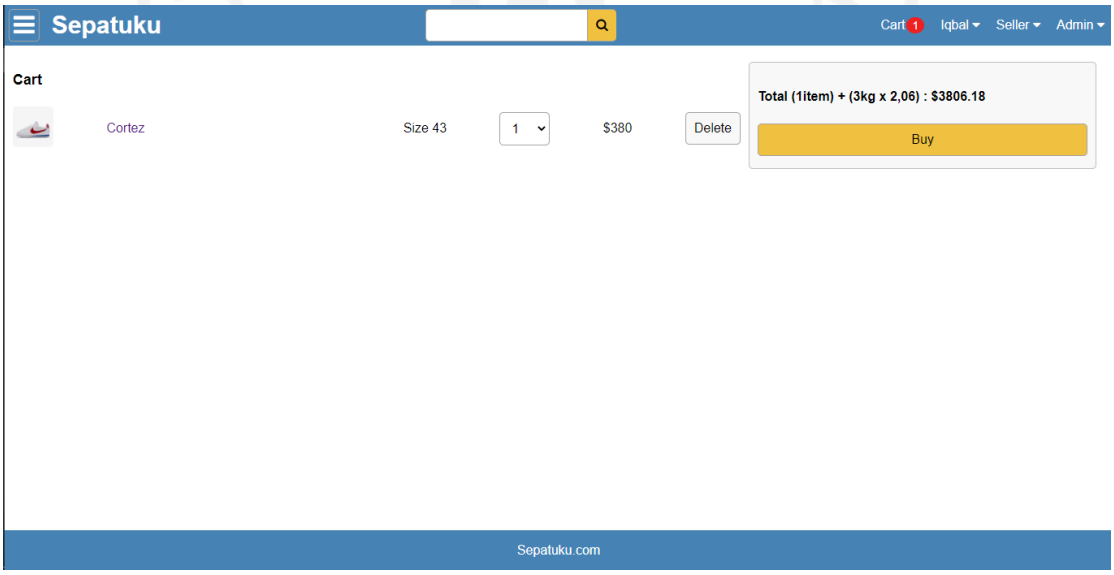

Gambar 4.15 Halaman antarmuka *edit info product*

### 4.7.2 Implementasi Antarmuka *Place Order*

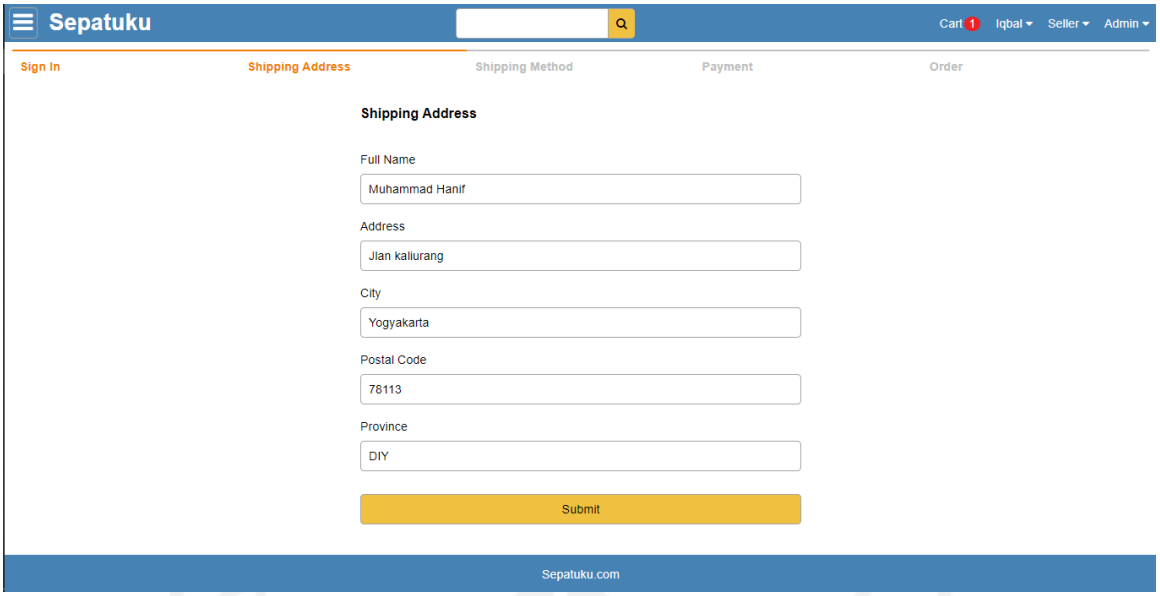
Pada bagian antarmuka *place order* terdapat terdapat satu fungsi, yaitu: *create order*.

#### A. Fungsi *Create Order*

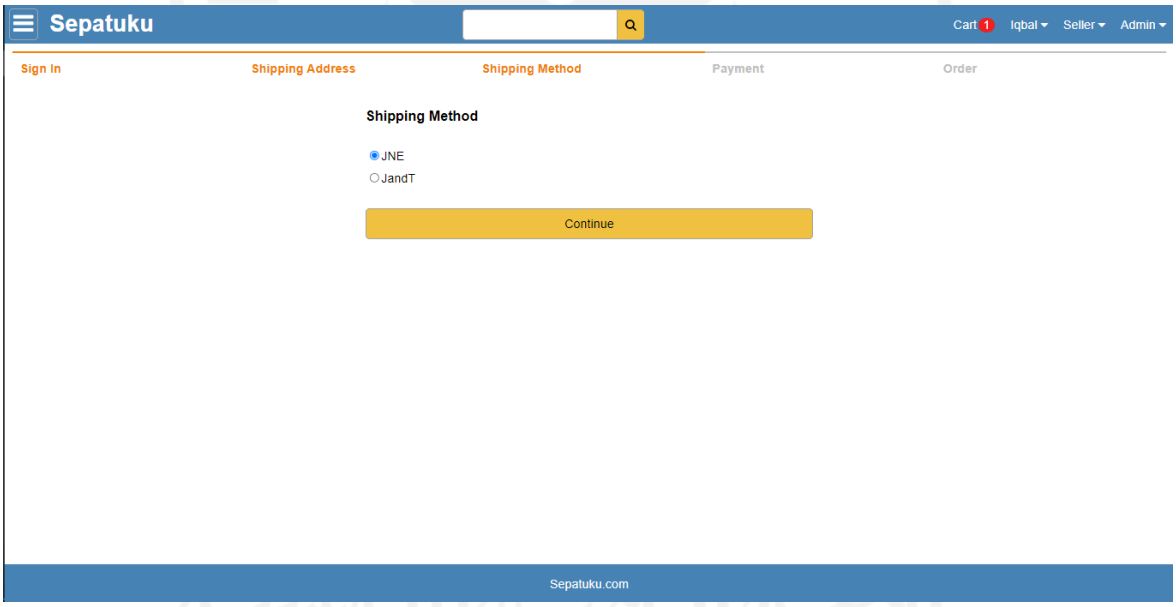
Antarmuka create order ditujukan untuk membuat pesanan dengan memasukkan informasi yang dibutuhkan penjual, yaitu: produk dan jumlah produk yang dipesan pada Gambar 4.16, alamat pengiriman pada Gambar 4.17, metode pengiriman pada Gambar 4.18, metode pembayaran Gambar 4.19, dan menampilkan keseluruhan informasi yang telah dibuat sebelumnya pada Gambar 4.20.



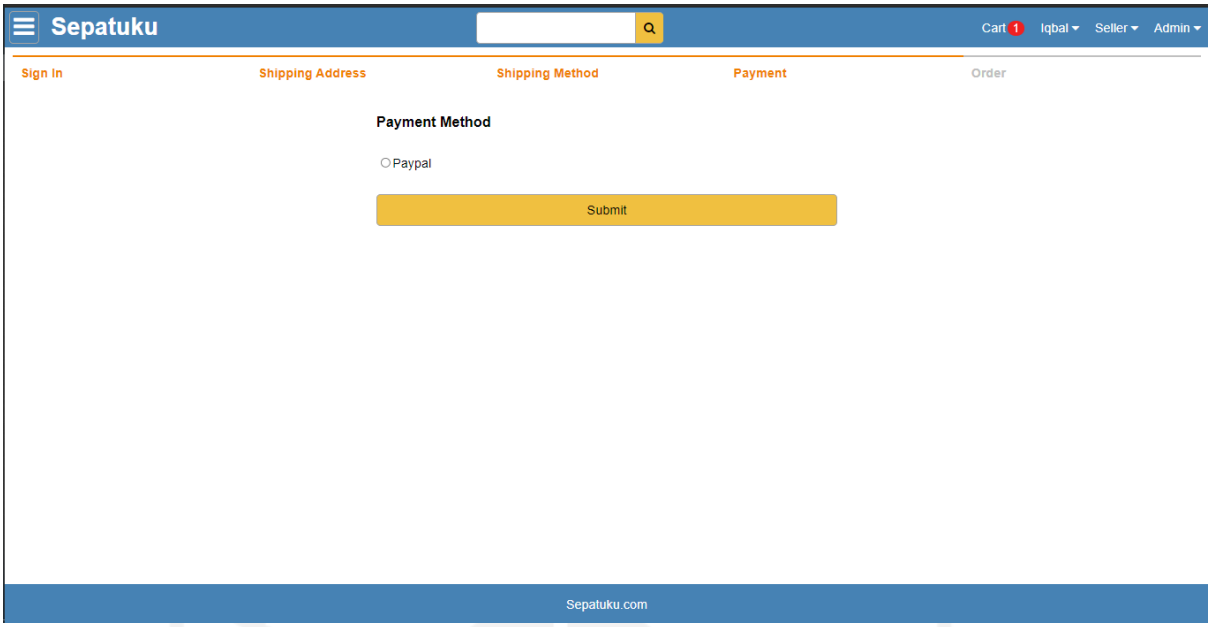

Gambar 4.16 Halaman atarmuka *add to cart*



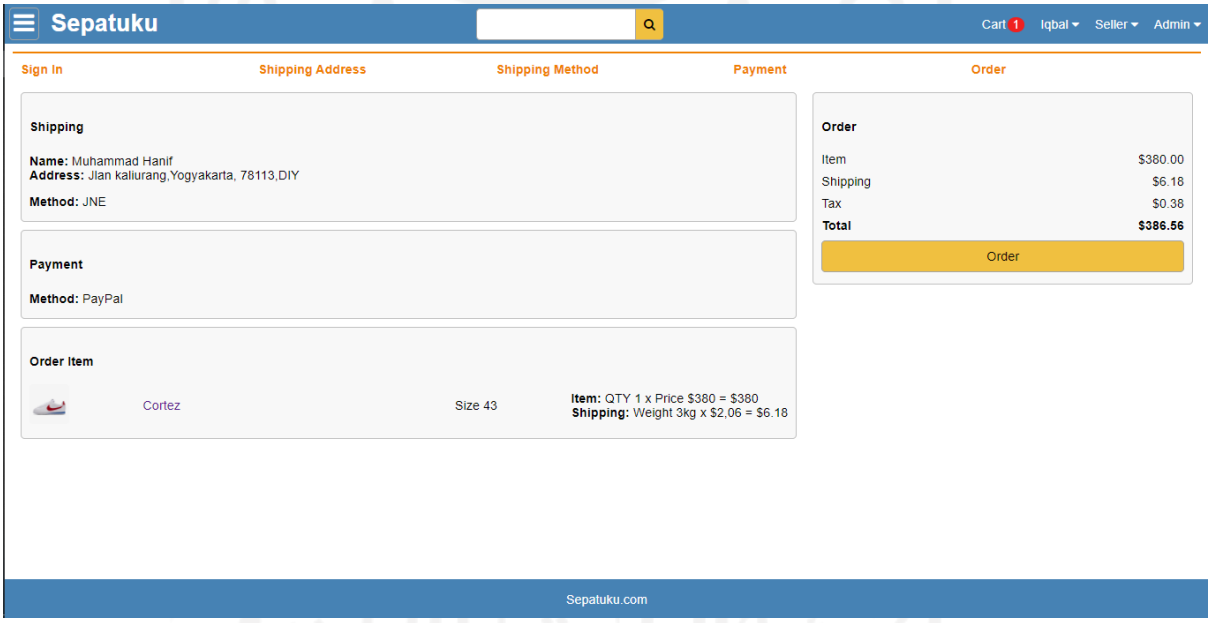

Gambar 4.17 Halaman antarmuka *shipping address*




Gambar 4.18 Halaman antarmuka *shipping method*




Gambar 4.19 Halaman antarmuka *payment method*



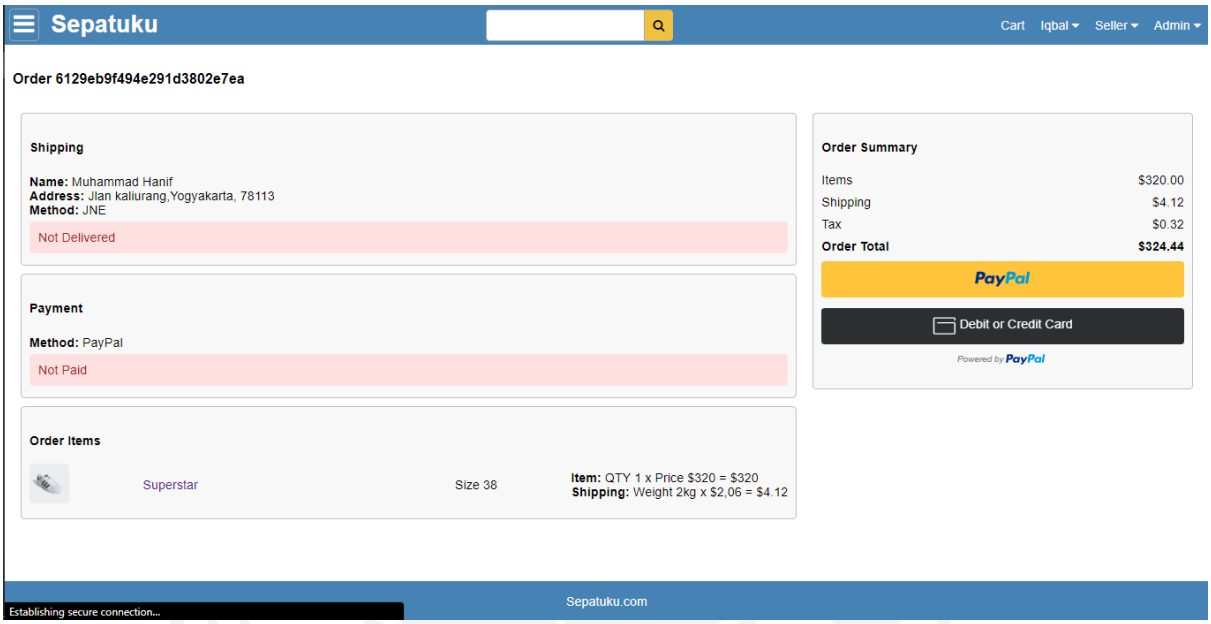

Gambar 4.20 Halaman antarmuka *place order*

### 4.7.3 Implementasi Antarmuka *Manage Order*

Pada bagian antarmuka *manage order* terdapat terdapat tiga fungsi, yaitu: *update status payment, update deliver order status, create payment.*

A. **Fungsi *Create Payment***

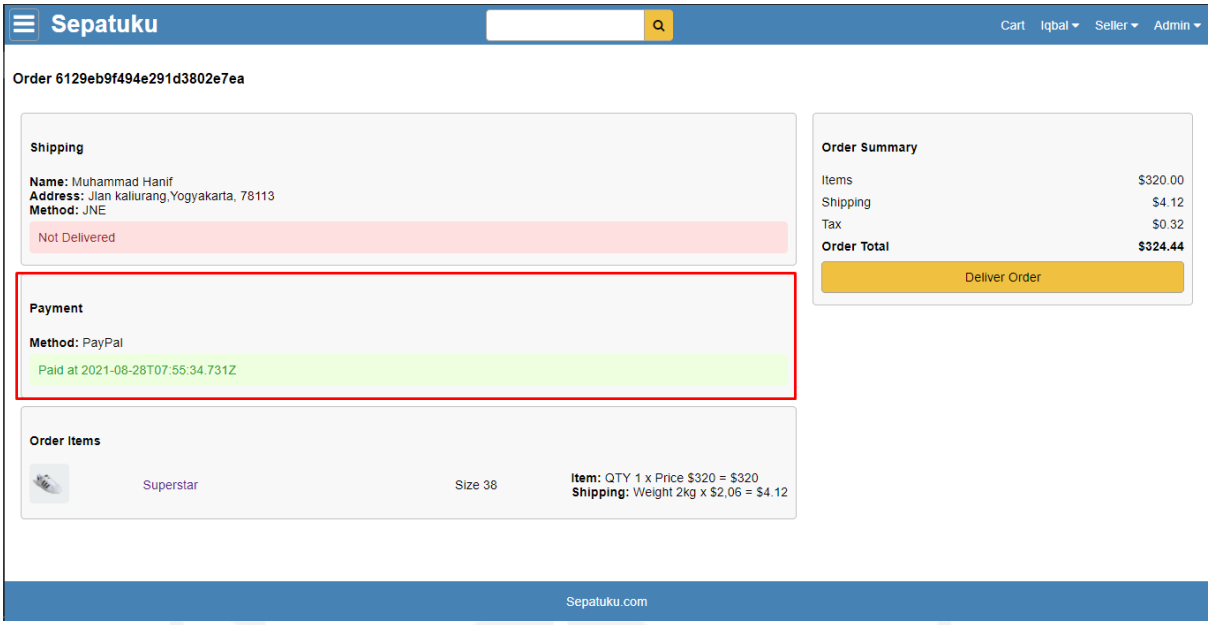
Pada halaman ini seperti Gambar 4.21, digunakan pengguna untuk melihat seluruh total harga yang harus dibayarkan.




Gambar 4.21 Halaman antarmuka *create payment*

B. **Fungsi *Update Status Payment***

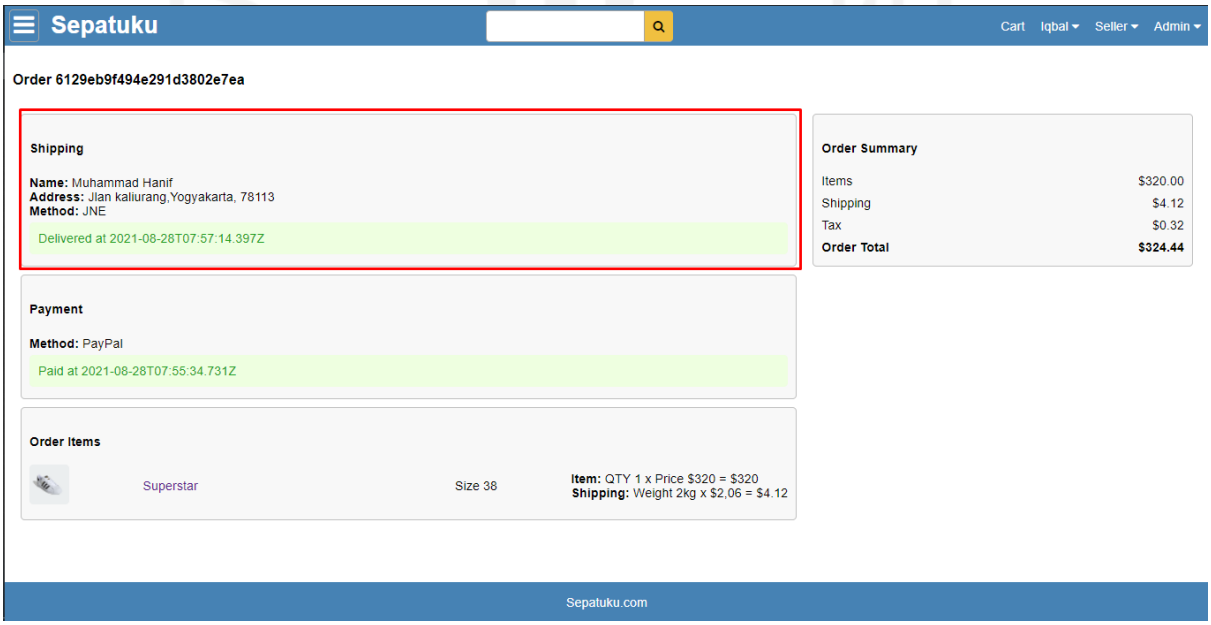
Pada halaman ini apabila pengguna telah sukses melakukan pembayaran maka status pembayaran akan berubah menjadi warna hijau yang artinya telah sukses dibayarkan seperti pada Gambar 4.22.




Gambar 4.22 Halaman antarmuka *update status payment*

## C. **Fungsi *Update Status deliver order***

Halaman ini Gambar 4.23 digunakan *seller* untuk mengubah status pengiriman menjadi telah dikirim apabila pengguna telah melakukan pembayaran.



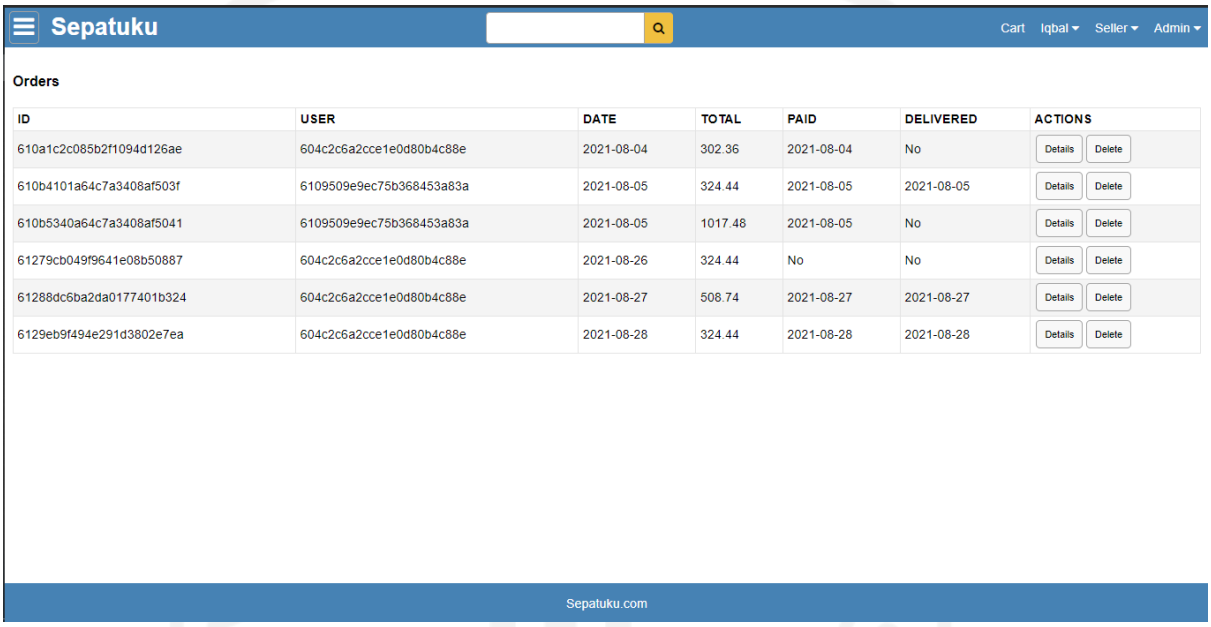

Gambar 4.23 Halaman antarmuka *update status deliver order*

### 4.7.4 Implementasi Antarmuka *View Order*

Pada bagian antarmuka *view order* terdapat terdapat dua fungsi, yaitu: *view all order, delete order.*

A. **Fungsi *View All Order***

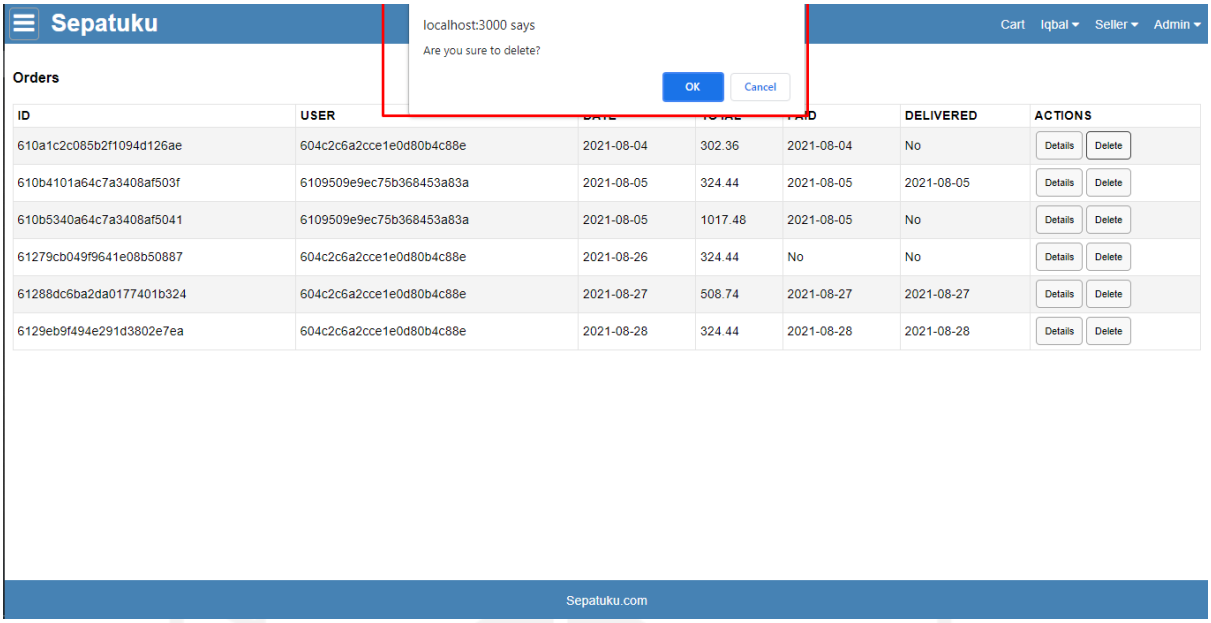
Halaman ini Gambar 4.24 digunakan *seller* untuk melihat keseluruhan pemesanan yang telah masuk beserta informasi pembayaran dan juga informasi produk.




Gambar 4.24 Halaman antarmuka *view all order*

B. **Fungsi *Delete Order***

Pada halaman ini antarmuka digunakan oleh *seller* untuk menghapus pesanan yang masuk, serta sebelum pesanan benar-benar di hapus datanya, pada Gambar 4.25 *seller* akan menerima peringatan apakah pesanan benar-benar akan dihapus.



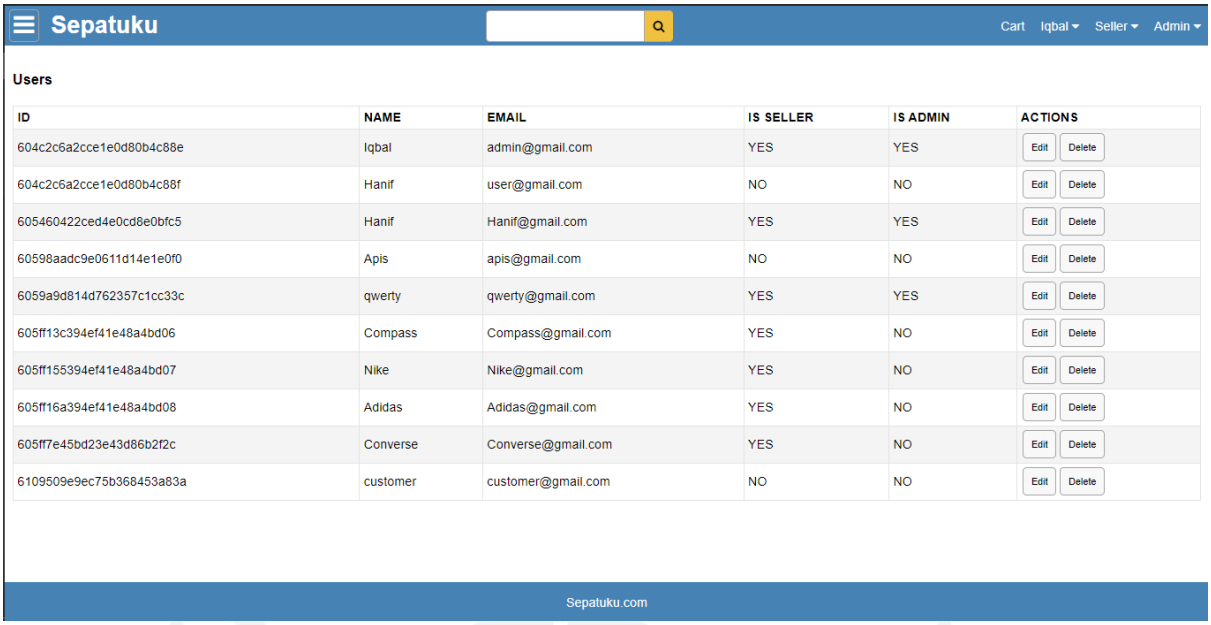

Gambar 4.25 Halaman antarmuka *delete order*

### 4.7.5 Implementasi Antarmuka *Manage User*

Pada bagian antarmuka *manage user* terdapat terdapat tiga fungsi, yaitu: *view all user, update user status, delete user.*

A. **Fungsi *View All User***

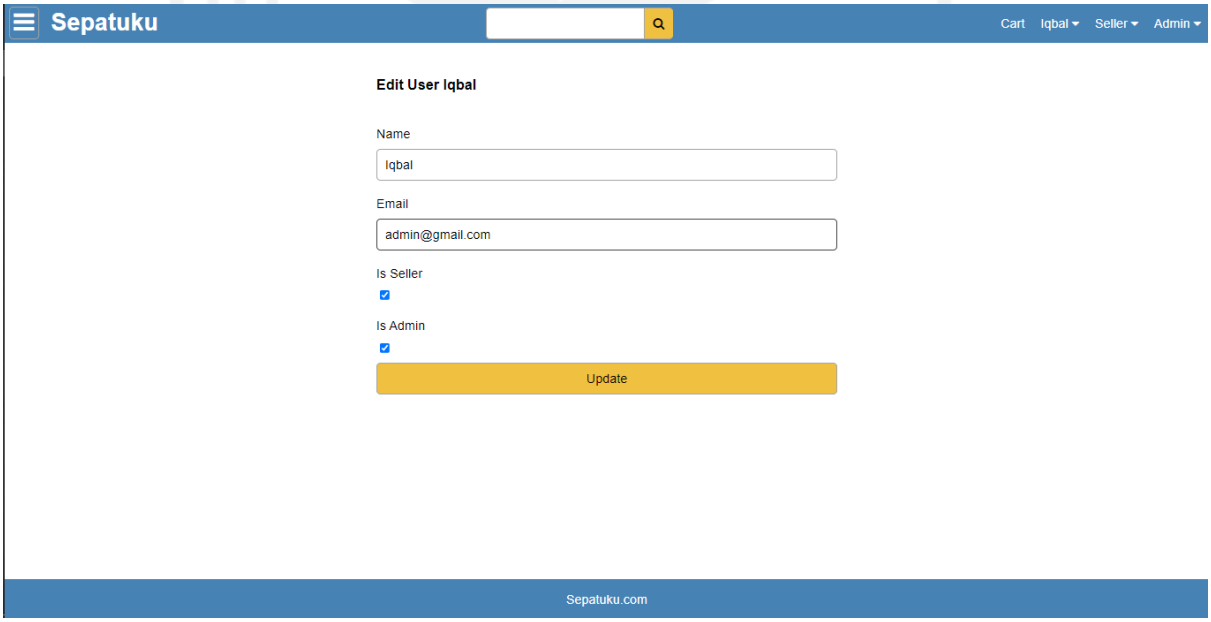
Pada halaman Gambar 4.26 antarmuka ini digunakan oleh *seller* untuk melihat keseluruhan pengguna serta melihat keseluruhan informasi pengguna.




Gambar 4.26 Halaman antarmuka *view all user*

## B. **Fungsi *Update User Status***

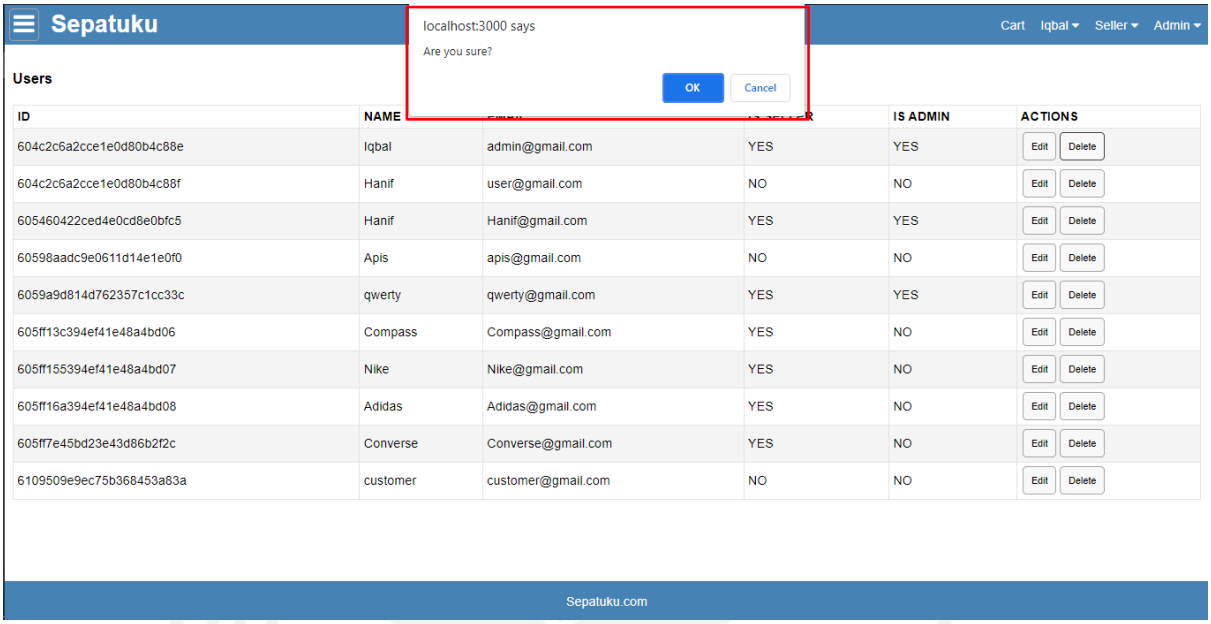
Halaman ini Gambar 4.27 digunakan *admin* untuk merubah status dari keseluruhan *user*.




Gambar 4.27 Halaman antarmuka *update user status*

C. **Fungsi *Delete User***

Pada halaman ini Gambar 4.28 antarmuka digunakan oleh *admin* untuk menghapus *user*, serta sebelum *user* benar-benar di hapus datanya, pada Gambar 4.28 *admin* akan menerima peringatan apakah pesanan benar-benar akan dihapus.



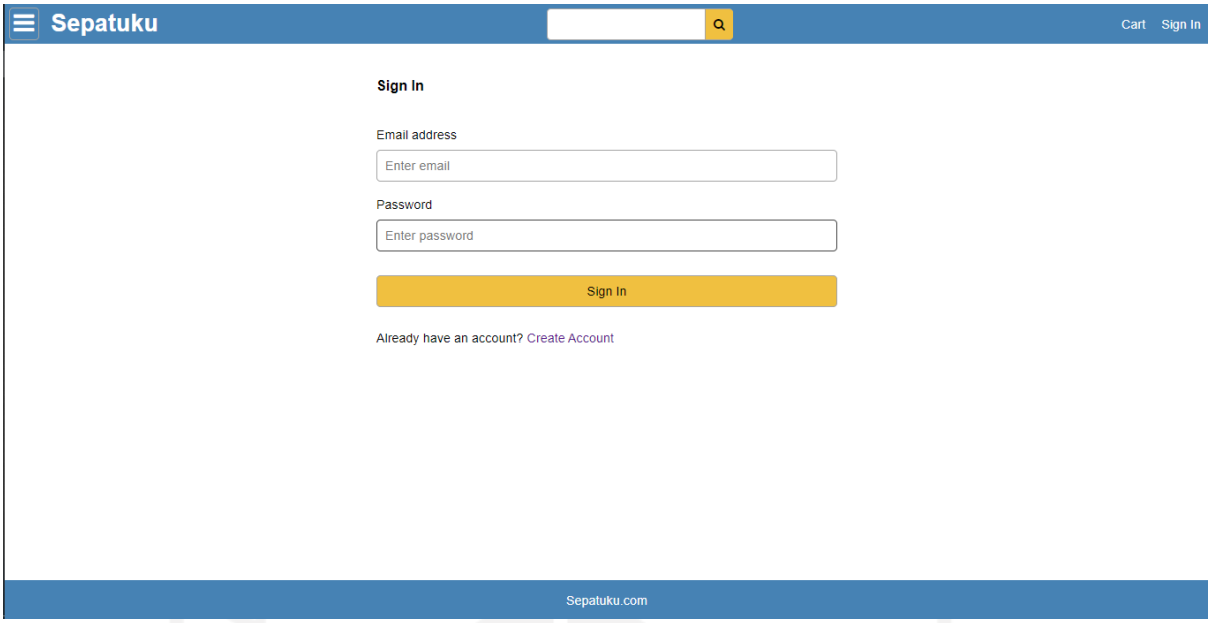

Gambar 4.28 Halaman antarmuka *delete user*

### 4.7.6 Implementasi Antarmuka *Manage My Profile*

Pada bagian antarmuka *manage my profile* terdapat terdapat empat fungsi, yaitu: *sign in, register, view info user, edit info user.*

A. **Fungsi *Sign In***

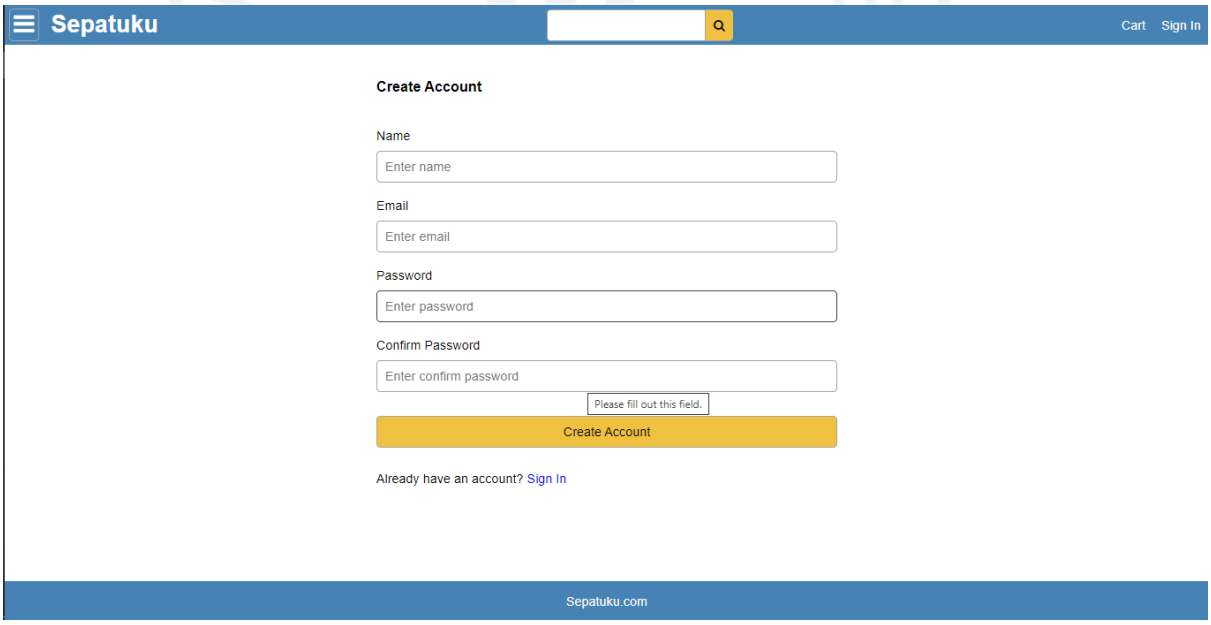
Halaman ini Gambar 4.29 digunakan pengguna untuk masuk kedalam aplikasi menggunakan akun yang sudah dibuat untuk dapat mengakses berbagai macam produk.




Gambar 4.29 Halaman antarmuka *Sign In*

B. **Fungsi *Register***

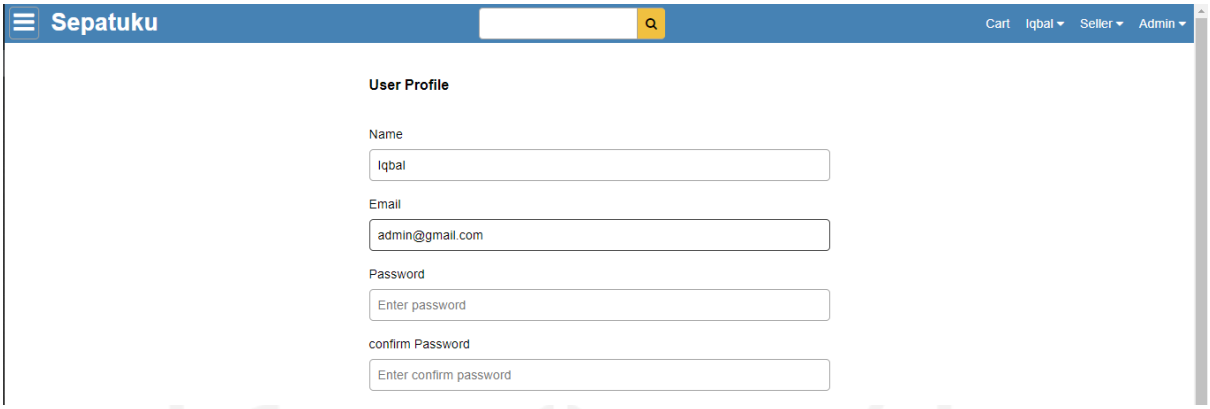
Halaman ini Gambar 4.30 digunakan pengguna untuk melakukan pendaftaran akun apabila tidak mempunyai akun.




Gambar 4.30 Halaman antarmuka *register*

C. **Fungsi *View Info User***

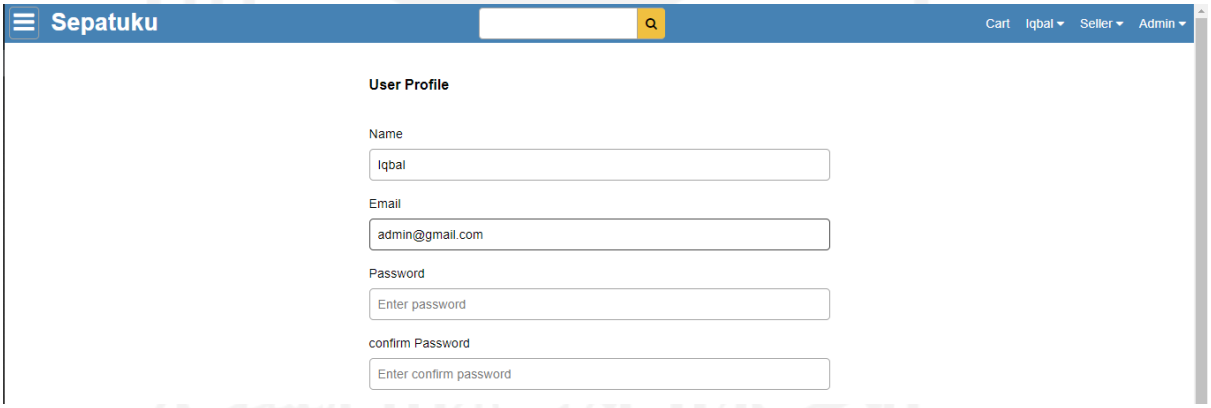
Halaman ini Gambar 4.31 digunakan pengguna untuk melihat informasi akun mereka.




Gambar 4.31 Halaman antarmuka *view info user*

D. **Fungsi *View Info User***

Halaman ini Gambar 4.32 digunakan pengguna juga untuk merubah informasi akun mereka.



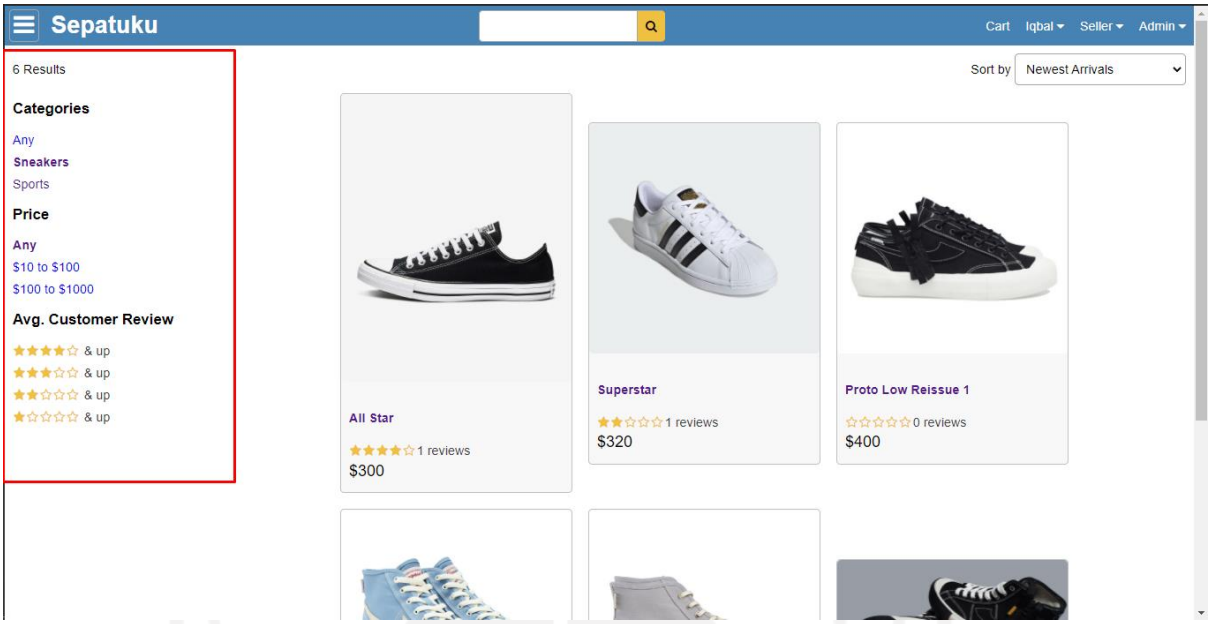

Gambar 4.32 Halaman antarmuka *edit info user*

### 4.7.7 Implementasi Antarmuka *Browse Product*

Pada bagian antarmuka *browse product* terdapat terdapat satu fungsi, yaitu: *filter product.*

A. **Fungsi *Filter Product***

Halaman ini Gambar 4.33 digunakan pengguna untuk mencari produk sesuai kategori, harga, serta ulasan produk yang mereka inginkan.



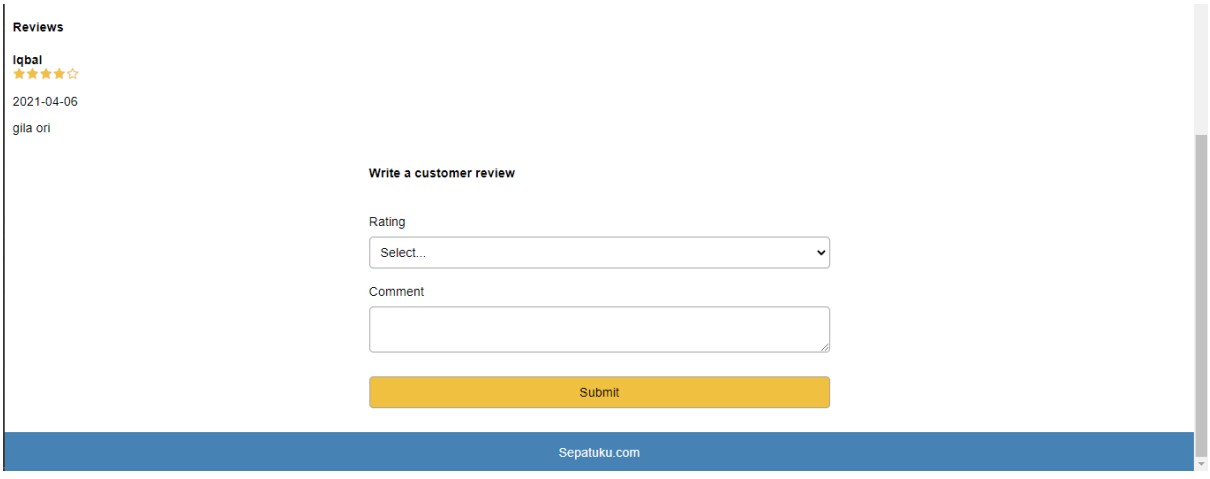

Gambar 4.33 Halaman antarmuka *filter product*

### 4.7.8 Implementasi Antarmuka *Get Statistic*

Pada bagian antarmuka *get statistic* terdapat terdapat dua fungsi, yaitu: *review product*, *view sales.*

A. **Fungsi *Review Product***

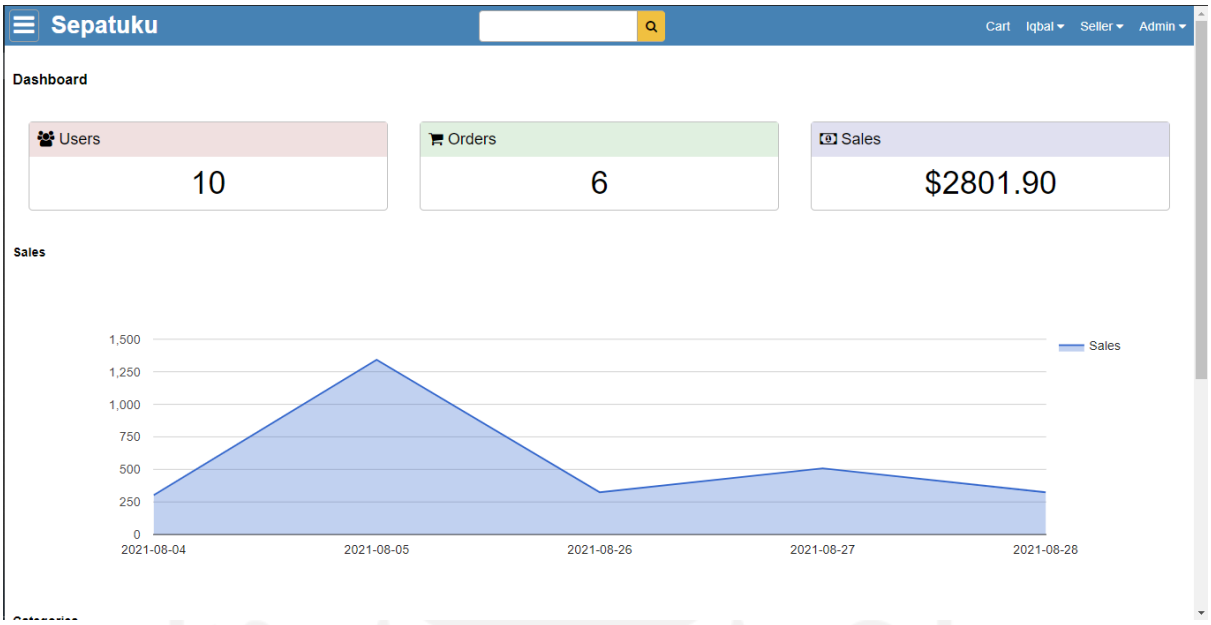
Halaman ini Gambar 4.34 digunakan pengguna untuk melakukan ulasan terhadap produk yang telah mereka pesan.




Gambar 4.34 Halaman antarmuka *review product*

B. **Fungsi *View Sales***

Halaman ini Gambar 4.35 digunakan seller untuk melihat keseluruhan penjualan perbulan.




Gambar 4.35 Halaman antarmuka *view sales*

## 4.8 Pengujian Antarmuka

Setelah antarmuka diimplementasikan kemudian antarmuka diuji apakah keluaran dari tiap antarmuka telah memenuhi harapan.

### 4.8.1 Pengujian Antarmuka *Manage Product*

Berikut hasil dari pengujian halaman antarmuka *manage product* seperti yang ditunjukkan pada tabel 4.14.

Tabel 4.14 Pengujian antarmuka *manage product*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *create product*** | | | | |
| 1. | *Unit* | Menggunakan tombol *create product* untuk menambahkan produk beserta informasi produk | Data *product* pada basis data *product* bertambah dan produk berhasil ditampilkan | Valid |
| **Fungsi *view all product*** | | | | |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 2. | *Unit* | Menggunakan tombol *product* pada *seller* untuk melihat keseluruhan produk | Seluruh data *product* ditampilkan termasuk informasi *seller* | Valid |
| **Fungsi *delete product*** | | | | |
| 3. | *Unit* | Menggunakan tombol *delete* yang ada pada informasi produk dan menerima peringatan untuk meyakinkan bahwa produk akan dihapus | Data *product* pada basis data *product* terhapus serta produk yang dihapus hilang | Valid |
| **Fungsi *Edit info product*** | | | | |
| 4. | *Unit* | Menggunakan tombol *edit info product* untuk mengubah informasi produk | Data pada basis data *product* berubah sesuai dengan parameter yang telah dimasukkan | Valid |

### 4.8.2 Pengujian Antarmuka *Place Order*

Berikut hasil dari pengujian halaman antarmuka *place order* seperti yang ditunjukkan pada tabel 4.15.

Tabel 4.15 Pengujian antarmuka *place order*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *create order*** | | | | |
| 1. | *Unit* | Menggunakan tombol *add to cart* untuk menyimpan produk pada keranjang pembelian beserta informasi jumlah yang akan dipesan | Menampilkan produk dan jumlah yang akan dipesan pada halaman *add to cart* | Valid |
| 2. | *Unit* | Menggunakan tombol *buy* untuk melakukan pemesanan | Menampilkan jumlah dan harga produk yang | Valid |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| | | | akan dipesan pada halaman *place order* | |
| 3. | *Unit* | Mengisi informasi alamat pengiriman dan menekan tombol *submit* | Menampilkan alamat pengiriman pada halaman *place order* | Valid |
| 4. | *Unit* | Memilih dan mengisi metode pengiriman yang akan digunakan dan menekan tombol *submit* | Menampilkan metode pengiriman pada halaman *place order* | Valid |
| 5. | *Unit* | Memilih dan mengisi metode pembayaran yang akan digunakan dan menekan tombol *submit* | Menampilkan metode pembayaran pada halaman *place order* | Valid |
| 6. | *Unit* | Menampilkan keseluruhan informasi pemesanan mulai dari jumlah produk yang dipesan, harga, alamat pengiriman, metode pengiriman, metode pembayaran, dan menekan tombol *order* untuk melakukan pemesanan | Data *order* pada basis data *order* bertambah | Valid |

### 4.8.3 Pengujian Antarmuka *Manage Order*

Berikut hasil dari pengujian halaman antarmuka *manage order* seperti yang ditunjukkan pada tabel 4.16.

Tabel 4.16 Pengujian antarmuka *manage order*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *update status payment*** | | | | |
| 1. | *Unit* | Menggunakan tombol *pay* untuk melakukan pembayaran | Data status *payment* pada basis data *order* berubah menjadi sukses dibayarkan dan menampilkan waktu pembayaran | Valid |
| **Fungsi *update deliver order status*** | | | | |
| 2. | *Unit* | Menggunakan tombol *deliver* untuk mengirimkan barang ke pemesan | Data *status deliver order* pada basis data *order* berubah menjadi sukses dikirimkan dan menampilkan waktu pengiriman | Valid |
| **Fungsi *create payment*** | | | | |
| 3. | *Unit* | Menggunakan tombol *order* untuk membuat pembayaran | Data *payment* pada basis data *order* ditambahkan dan menampilkan informasi harga yang harus dibayarkan | Valid |

### 4.8.4 Pengujian Antarmuka *View Order*

Berikut hasil dari pengujian halaman antarmuka *view order* seperti yang ditunjukkan pada tabel 4.17.

Tabel 4.17 Pengujian antarmuka *view order*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *view all order*** | | | | |

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| 1. | *Unit* | Menggunakan tombol *order* pada *seller* untuk melihat keseluruhan pesanan | Seluruh data *order* ditampilkan | Valid |
| **Fungsi *delete order*** | | | | |
| 2. | *Unit* | Menggunakan tombol delete pada order untuk menghapus pesanan | Data *order* pada basis order *user* terhapus dan pesanan hilang | Valid |

### 4.8.5 Pengujian Antarmuka *Manage User*

Berikut hasil dari pengujian halaman antarmuka *manage user* seperti yang ditunjukkan pada tabel 4.18.

Tabel 4.18 Pengujian antarmuka *manage user*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *view all user*** | | | | |
| 1. | *Unit* | Menggunakan tombol *user* pada *admin* untuk melihat keseluruhan pengguna | Seluruh data *user* ditampilkan | Valid |
| **Fungsi *update user status*** | | | | |
| 2. | *Unit* | Menggunakan tombol *edit* pada *user info* untuk merubah status pengguna | Data *user status* pada basis data *user* berubah dan *user* mendapatkan hak akses yang diubah | Valid |
| **Fungsi *delete user*** | | | | |
| 3. | *Unit* | Menggunakan tombol *delete* pada *user info* untuk menghapus pengguna | Data *user* pada basis data *user* terhapus dan pengguna tidak memiliki hak akses ke aplikasi lagi | Valid |

### 4.8.6 Pengujian Antarmuka *Manage My Profile*

Berikut hasil dari pengujian halaman antarmuka *manage user* seperti yang ditunjukkan pada tabel 4.19.

Tabel 4.19 Pengujian antarmuka *manage my profile*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *sign in*** | | | | |
| 1. | *Unit* | Menggunakan tombol *sign in* untuk masuk ke dalam aplikasi | Membaca data *user* dan membuat akun tersebut *login* | Valid |
| **Fungsi *register*** | | | | |
| 2. | *Unit* | Menggunakan tombol *register* untuk melakukan pendaftaran akun | Data *user* pada basis data *user* bertambah | Valid |
| **Fungsi *view info user*** | | | | |
| 3. | *Unit* | Menggunakan tombol *user profile* pada akun untuk melihat informasi akun | data *user profile* ditampilkan | Valid |
| **Fungsi *view info user*** | | | | |
| 4. | *Unit* | Menggunakan tombol *update* pada *user profile* untuk merubah informasi akun | Data *user* info pada basis data *user* berubah serta menampilkan informasi akun yang baru | Valid |

### 4.8.7 Pengujian Antarmuka *Browse Product*

Berikut hasil dari pengujian halaman antarmuka *browse product* seperti yang ditunjukkan pada tabel 4.20.

Tabel 4.20 Pengujian antarmuka *browse product*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi browse product** | | | | |
| 1. | *Unit* | Menggunakan tombol *side bar menu* untuk memilih kategori yang diinginkan | Membaca data *product* yang dicari dan menampilkan produk yang diinginkan | Valid |

### 4.8.8 Pengujian Antarmuka *Get Statistic*

Berikut hasil dari pengujian halaman antarmuka *get statistic* seperti yang ditunjukkan pada tabel 4.21.

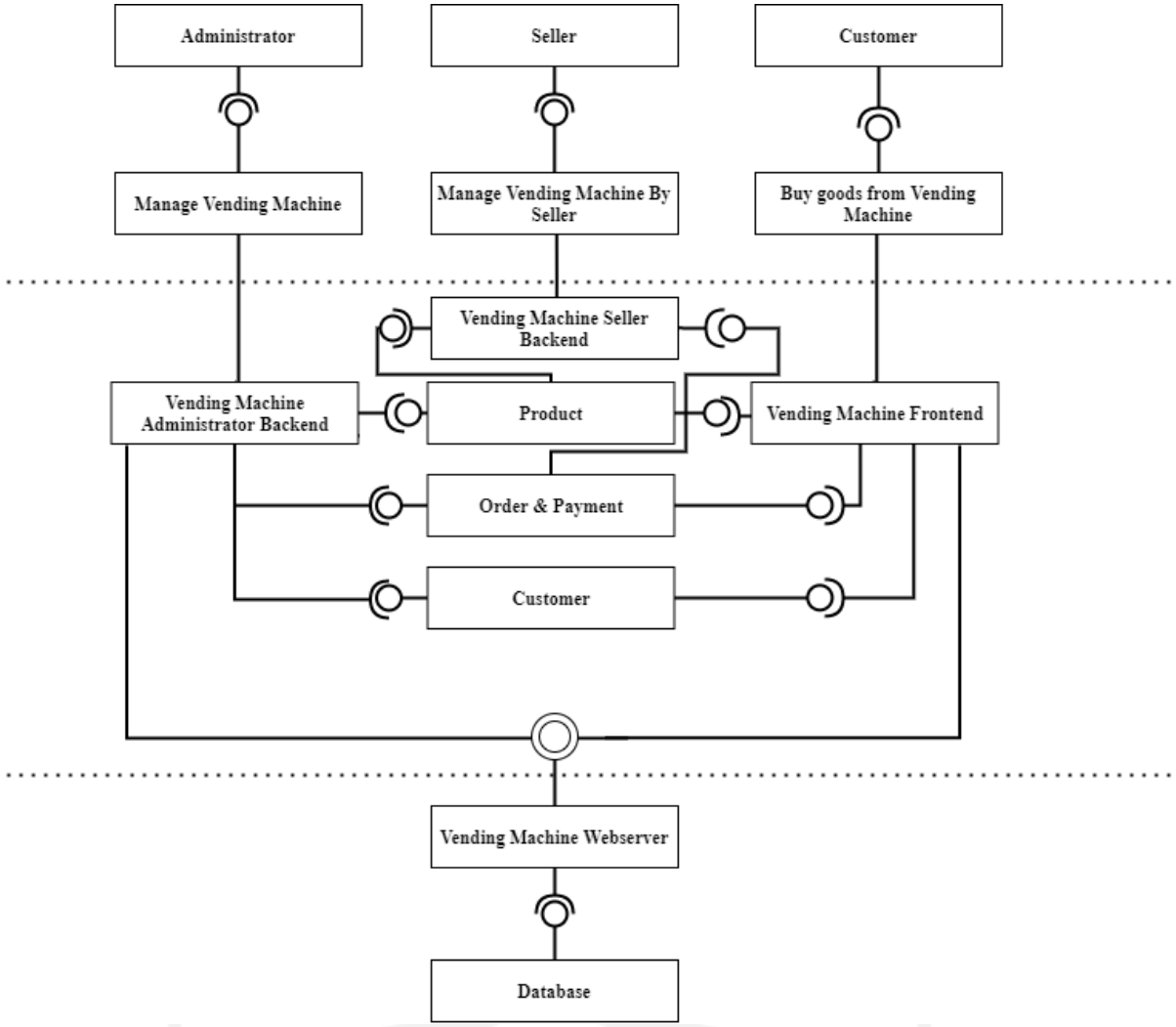
Tabel 4.21 Pengujian antarmuka *get statistic*

| No. | Lingkup | *Test case* | Hasil yang diharapakan | Hasil pengujian |
|---|---|---|---|---|
| **Fungsi *review product*** | | | | |
| 1. | *Unit* | Menggunakan tombol *review* pada produk yang telah dipesan | Data *review product* pada basis data *product* bertambah dan menampilkan *review* dari produk tersebut | Valid |
| **Fungsi *view sales*** | | | | |
| 2. | *Unit* | Menggunakan tombol dashboard untuk melihat keseluruhan penjualan | Menampilkan keseluruhan data penjualan beserta info keuntungan penjualan | Valid |

## 4.9 Arsitektur Aktual

Dalam proses implementasi terdapat penambahan proses bisnis aktor yaitu *seller* untuk mengelola produk yang dibuat oleh *seller* dan mengelola *order* dari produk yang dibuat oleh *seller*. Serta terdapat sebuah layanan yang tidak diimplementasikan yaitu *Send email to customer.*

### 4.9.1 Pandangan Holistic




Gambar 4.36 Pandangan *holistic*

Pada saat implementasi terjadi perubahan pada layanan yang ada pada pandangan *holistic* yang ditunjukkan pada Gambar 4.36.

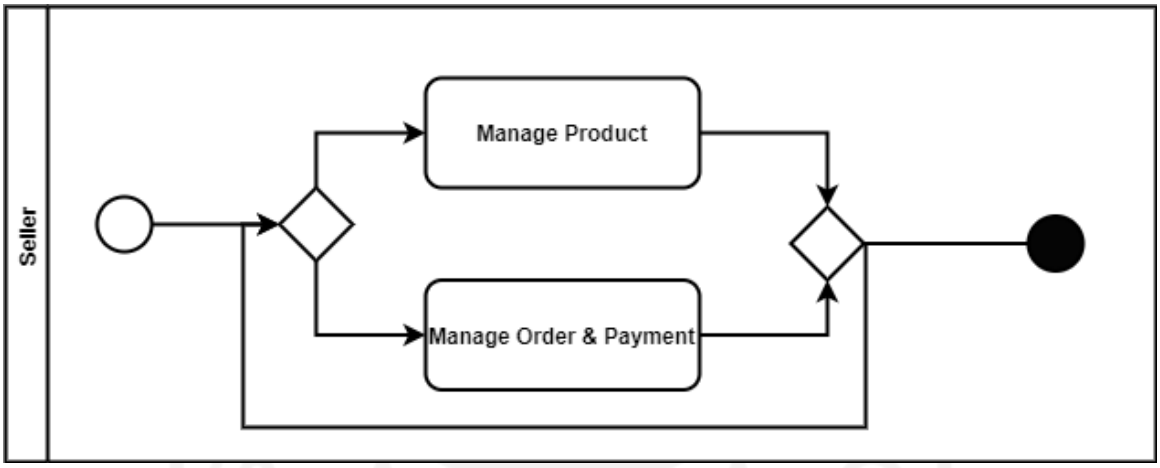
*a. Manage Product*

Layanan ini menggabungkan layanan *manage catalog* dan *manage articles* karena fungsinya yang saling berkaitan, dimana di dalam sebuah katalog produk biasanya telah tersedia harga produk, jumlah produk yang tersedia serta deskripsi dari sebuah produk. Deskripsi produk termasuk ke dalam artikel.

b. *Manage Order & Payment*

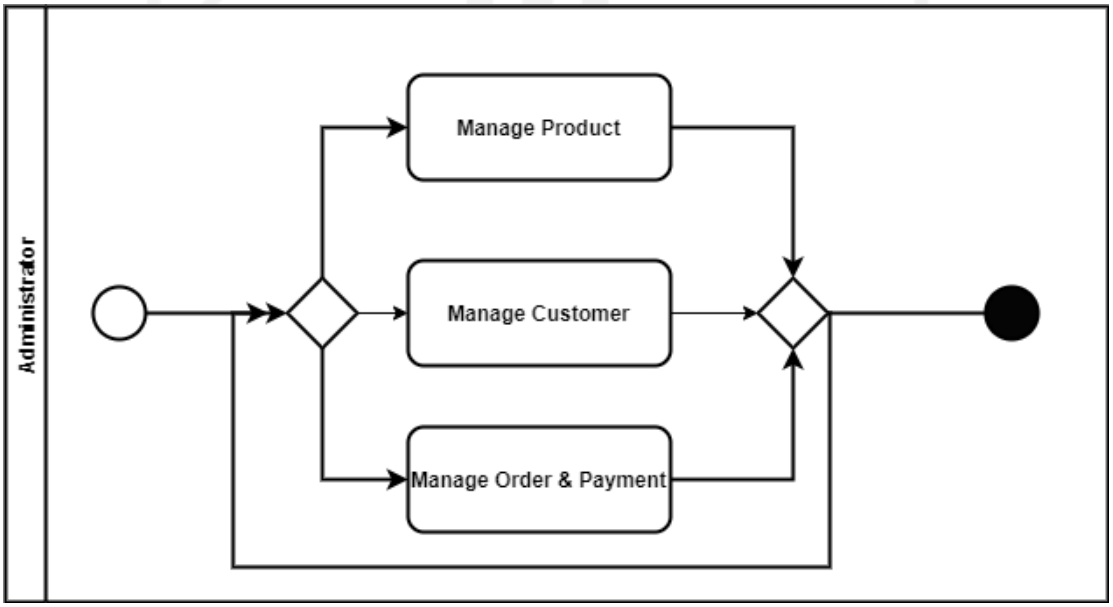
Layanan ini menggabungkan layanan *manage order* dan *manage invoices* karena fungsi yang digunakan sama dengan *manage order* yaitu melihat keseluruhan informasi *order*.

### 4.9.2 Proses Bisnis




Gambar 4.37 Proses bisnis *seller*

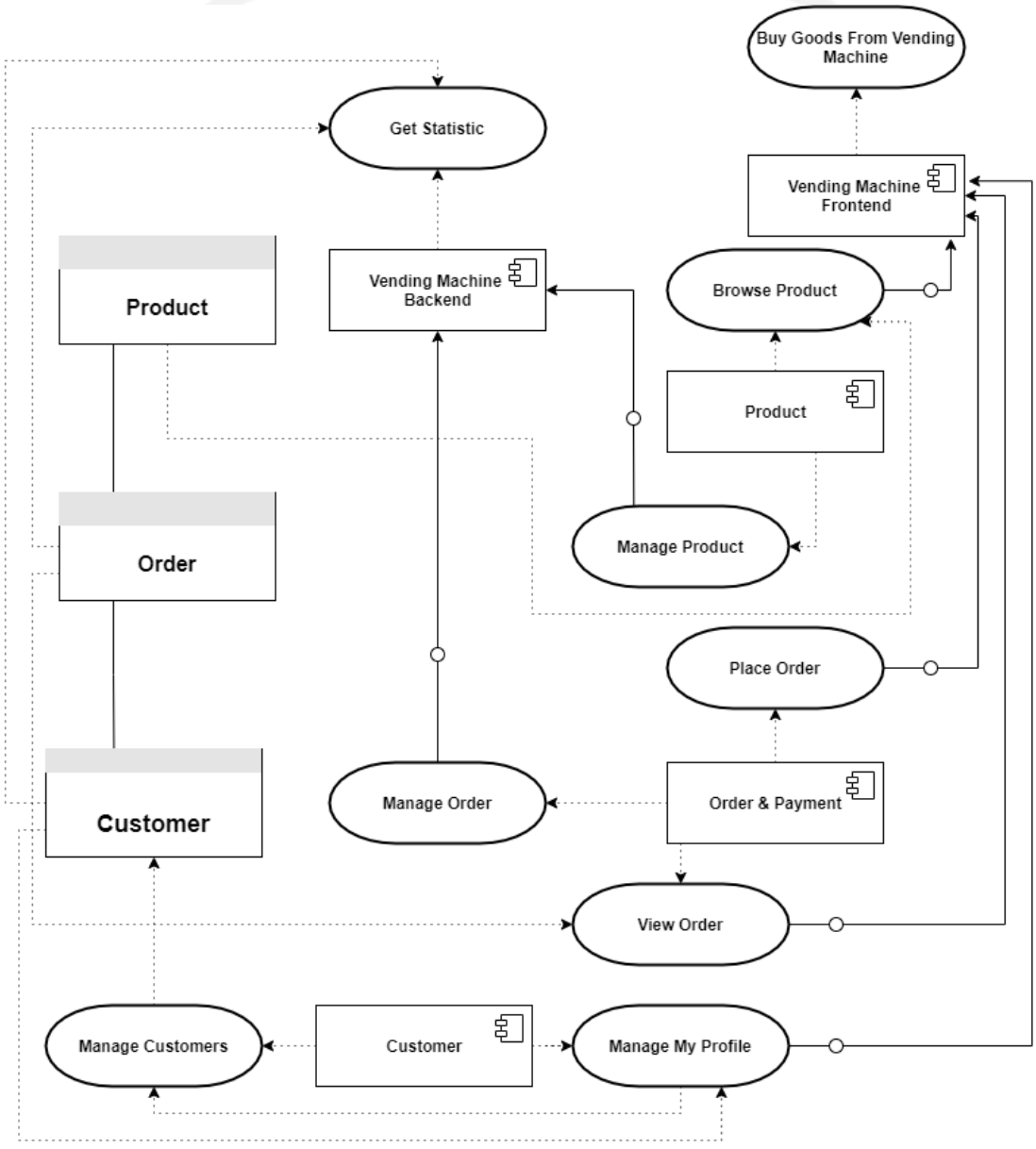
Gambar 4.37 memuat bagian penambahan proses bisnis aktor yaitu *seller*. Fungsi dari proses bisnis ini adalah mengelola produk yang telah ditawarkan *seller* beserta *order* yang masuk.




Gambar 4.38 Proses bisnis *admin*

Pada proses bisnis *admin* terjadi penyesuaian terhadap perubahan yang terjadi pada saat penggabungan beberapa layanan yaitu layanan *manage articles* dan *manage catalog* yang digabungkan menjadi *manage product*. Kemudian penggabungan layanan *manage order* dan *manage invoices* menjadi *manage order* & *payment* pada pandangan *holistic* seperti yang ditunjukkan pada gambar 4.38.

### 4.9.3 Struktur Data dan Aplikasi




Gambar 4.39 Arsitektur data dan aplikasi

Gambar 4.39 merupakan arsitektur data dan aplikasi yang telah disesuaikan dengan perubahan proses bisnis serta perubahan basis data. Berdasarkan pengamatan pada saat implementasi dilakukan perubahan pada arsitektur data dan aplikasi dengan menggabungkan beberapa basis data serta menggabungkan beberapa layanan menjadi satu agar lebih efektif dan efisien.

### 4.9.4 Basis Data

Dari pola yang sudah diberikan dari awal terjadi beberapa penggabungan beberapa objek data dalam basis data. Tabel 4.22 menyediakan uraian beberapa basis data yang masing-masing berdiri sendiri di tiap layanannya.

Tabel 4.22 Basis data di tiap layanan

| **Layanan** | **Basis data** |
|---|---|
| Manage Product | Product |
| Browse Product | |
| Get Statistic | |
| Place Order | Order |
| Manage Order | |
| View Order | |
| Manage User | User |
| Manage My Profile | |

*a. Product*

Basis data ini berisi keseluruhan informasi produk mulai dari nama, harga, gambar, berat, kategori merek, jumlah stok serta deskripsi produk.

*b. Order*

Basis data ini berisi seluruh informasi pemesanan mulai dari *invoices*, *payment*, *payment method* serta *shipping method*.

*c. User*

Basis data ini berisi informasi keseluruhan pengguna mulai dari *admin*, *customer* dan juga *seller*.

## 4.10 Layanan API

Terdapat 8 layanan API yang dihasilkan. Seluruh layanan diimplementasikan menggunakan Node.js dan Axios untuk saling berkomunikasi. Berikut 8 layanan tersebut.

### 4.10.1 Layanan *Manage Product*

Layanan ini ditujukan untuk mengelola seluruh produk yang memiliki fungsi yaitu:

- *Create product*
- *View all product*
- *Edit info product*
- *Delete product.*

### 4.10.2 Layanan *Place Order*

Layanan ini memasukkan produk yang akan dipesan ke dalam keranjang beserta alamat pemesan dan pemilihan metode pembayaran yang akan digunakan. Fungsi *pada* layanan *place order* adalah *create order.*

### 4.10.3 Layanan *Manage Order*

Pada bagian layanan ini ditujukan untuk memeriksa informasi order apakah transaksi telah dilakukan oleh pemesan produk, apabila produk tersebut telah dilunasi maka *admin/seller* terkait dapat mengirimkan produk, serta pengguna mendapatkan informasi bahwa produk yang dipesan telah dikirim. Berikut fungsi dari layanan *manage order*:

- *Pay via client id paypal*
- *Update payment*
- *Update deliver order status.*

### 4.10.4 Layanan *View Order*

Layanan ini digunakan pengguna untuk melihat pesanan yang telah dibuatnya. *Admin* dapat melihat dan menghapus keseluruhan pesanan yang telah dibuat oleh pengguna. Berikut fungsi dari layanan *view order*:

- *View all order*
- *Delete order.*

### 4.10.5 Layanan *Manage User*

Layanan ini digunakan pengguna untuk melihat pesanan yang telah dibuatnya. *Admin* dapat melihat dan menghapus keseluruhan pesanan yang telah dibuat oleh pengguna. Berikut fungsi dari layanan *Manage user*:

- *View all user*
- *Update user status*
- *Delete user.*

### 4.10.6 Layanan *Manage My Profile*

Layanan ini ditujukan untuk menyimpan segala informasi akun yang dibutuhkan pengguna mulai dari nama, *email*, hingga *password*. apabila pengguna belum mempunyai akun, maka pengguna harus membuat akunnya terlebih dahulu. Apabila pengguna telah memiliki akun pengguna hanya perlu masuk menggunakan akun yang telah ada. Berikut fungsi dari layanan *manage my profile*:

- *Sign in*
- *Register*
- *View Info User*
- *Edit Info User.*

### 4.10.7 Layanan *Browse Product*

Layanan ini untuk memudahkan pengguna dalam mencari produk yang diinginkan sesuai dengan nama, penjual, harga, kategori, maupun rating produk. Fungsi dari layanan *browse product* adalah *filter product*.

### 4.10.8 Layanan *Get Statistic*

Untuk mengetahui kualitas produk yang dijual, diperlukan ulasan tiap produk dari *customer* yang telah memesan produk. Selain itu, informasi mengenai jumlah penjualan produk yang masuk per bulan dapat dilihat pada layanan ini. Berikut fungsi yang ada dalam layanan *get statistic:*

- *Review Product*
- *View sales.*

## 4.11 Pembahasan

Rancangan yang setara dengan pola arsitektur *enterprise vending machine* dibutuhkan saat mengembangkan aplikasi toko *online* berbasis teknologi *microservices*. Rancangan dibuat menggunakan kerangka kerja untuk memastikan arsitektur dirancang secara bertahap untuk meyakinkan bahwa kebutuhan bisnis *enterprise* dibantu sepenuhnya oleh sistem teknologi informasi *enterprise* mulai dari ranah arsitektur data, aplikasi hingga teknologi untuk mendukung kebutuhan bisnis *enterprise* oleh sistem yang akan dibangun. Kerangka kerja juga dapat menghasilkan pemodelan di tiap ranah terperinci. Terutama pemilihan teknologi menggunakan teknologi *microservices* yang mempermudah pengembangan dikarenakan tiap layanan dapat dikembangkan tanpa mengganggu layanan lain, serta tidak ada hambatan untuk menggunakan teknologi baru. Hal ini mampu memberikan dampak positif terutama dalam kasus yang diangkat, yakni berupa toko *online*.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Setelah dilakukan pengujian, implementasi pola arsitektur *enterprise vending machine* dalam kasus konkret berupa toko *online* pada teknologi *microservices* dapat disimpulkan sebagai berikut.

1. Rancangan aplikasi yang setara dengan pola arsitektur *enterprise vending machine* dibutuhkan saat mengembangkan aplikasi berbasis teknologi *microservices*.
2. Pengembangan pemodelan pola arsitektur *enterprise vending machine* dapat menghasilkan arsitektur bisnis, data, serta teknologi yang terperinci.

## 5.2 Saran

Penelitian ini berisikan deskripsi implementasi arsitektur *enterprise* pola *vending machine* pada teknologi *microservices.* Dikarenakan sedikitnya literatur mengenai pembahasan pengembangan serupa yang didasari oleh arsitektur *enterprise* penelitian ini memerlukan penelitian lebih lanjut untuk memvalidasi serta menyempurnakan.

# DAFTAR PUSTAKA

Loukides, M., & Swoyer, S. (2020). *Microservices Adoption in 2020 – O'Reilly*. 2021-04-30. https://www.oreilly.com/radar/microservices-adoption-in-2020/

Mufrizal, R., & Indarti, D. (2019). Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *5*(1), 57–68. https://doi.org/10.25077/teknosi.v5i1.2019.57-68

Perroud, T., & Inversini, R. (2013). Enterprise Architecture Patterns. In *Enterprise Architecture Patterns*. https://doi.org/10.1007/978-3-642-37561-3

Purnama, Heri, I. Y. B. (2010). *APLIKASI PENGELOLAAN SKRIPSI DI STMIK AKAKOM YOGYAKARTA MENGGUNAKAN ARSITEKTUR MICROSERVICE DENGAN Node.js*. 1–28. http://perpus.akakom.ac.id/

Putra, S. D., & Saputra, K. (2019). *Desain Dan Implementasi Microservices Studi Kasus Pada Layanan Taking Order ( Aplikasi E-Commerce Pt Xyz )*.

Rafiqi, M. D., Subyantoro, E., & W, D. K. (n.d.). *Implementasi Arsitektur Microservice Pada Aplikasi Online Travel Tourinc*. *1*, 1–10.

Richardson, C. (2018). *Microservice Architecture pattern*. Microservices.Io. https://microservices.io/patterns/microservices.html

Shah, H., & El Kourdi, M. (2007). Frameworks for enterprise architecture. *IT Professional*, *9*(5), 36–41. https://doi.org/10.1109/MITP.2007.86

Yunis, R., & Surendro, K. (2009). Model Enterprise Architecture Untuk Perguruan Tinggi di Indonesia. *Seminar Nasional Informatika 2009*, *I*(semnasIF), E72–E79. http://jurnal.upnyk.ac.id/index.php/semnasif/article/view/909/783

# LAMPIRAN