

LAPORAN TUGAS AKHIR
PENGEMBANGAN MODUL NILAI TUKAR MATA UANG
PADA WEB PRODUK PENGIRIMAN UANG KE LUAR
NEGERI
DEVELOPMENT OF CURRENCY EXCHANGE RATE MODULE
IN REMITTANCE WEB PRODUCT



ELANG BAYU SEGARA
17/416345/SV/14083

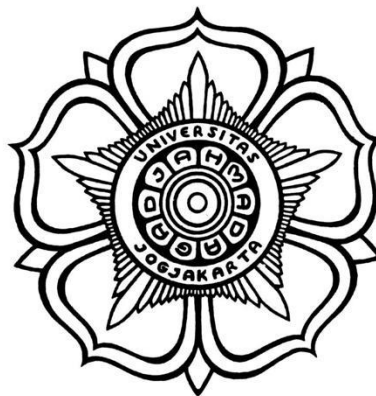
PROGRAM STUDI D3 KOMPUTER DAN SISTEM INFORMASI
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2020

LAPORAN TUGAS AKHIR

PENGEMBANGAN MODUL NILAI TUKAR MATA UANG
PADA WEB PRODUK PENGIRIMAN UANG KE LUAR
NEGERI

DEVELOPMENT OF CURRENCY EXCHANGE RATE MODULE
IN REMITTANCE WEB PRODUCT

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Ahli Madya
Komputer dan Sistem Informasi



ELANG BAYU SEGARA
17/416345/SV/14083

PROGRAM STUDI D3 KOMPUTER DAN SISTEM INFORMASI
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2020

HALAMAN PENGESAHAN

PERNYATAAN BEBAS PLAGIASI

KATA PENGANTAR

Puja dan puji syukur kehadiran Allah SWT, hanya dengan rahmat dan kuasaNya sehingga penulis mampu menyelesaikan tugas akhir yang berjudul “Pengembangan Modul Nilai Tukar Uang Pada Web Produk Pengiriman Uang Ke Luar Negeri”. Tugas akhir ini merupakan sebagai syarat untuk mendapatkan gelar Ahli Madya di fakultas Sekolah Vokasi, Universitas Gadjah Mada.

Banyak pengalaman dan pelajaran berharga yang penulis dapatkan pada pengerjaan tugas akhir ini, terutama semua cerita dan pengetahuan baru yang ada di tempat magang penulis yaitu PT Artajasa. Sebuah keluarga baru yang menjadi guru sekaligus pengayom bagi penulis.

Selanjutnya, tidak lupa penulis ucapkan rasa hormat dan terima kasih yang sebesar-besarnya kepada pihak-pihak yang mendukung penyelesaian tugas akhir ini, di antaranya:

1. Allah SWT yang telah memberikan nikmat dan karunia sehat dan sempat sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.
2. Ibu Widji Sriwahjuni, seseorang yang paling utama di hidup penulis, yang kasih sayang dan do'anya tidak pernah tidak akan putus untuk selalu memberi semangat, nasihat, bimbingan, dan motivasi kepada penulis. Serta seluruh anggota keluarga Bapak Ngatepiadji yang sudah mendukung dan memberikan do'anya kepada penulis.
3. Bapak Prof. Ir. Panut Mulyono, M.Eng., D.Eng., selaku Rektor Universitas Gadjah Mada.
4. Bapak Nur Rohman Rosyid, S.T., M.T., D.Eng., selaku kepala Program Studi D3 Komputer dan Sistem Informasi, Sekolah Vokasi, Universitas Gadjah Mada.
5. Bapak Muhammad Fakhrrurifqi, S.Kom., M.Cs., selaku dosen pembimbing yang dengan sabar dan telaten memberikan bimbingan, arahan dan semangat dalam penyusunan Tugas Akhir.

6. Bapak Hendi Fatria, selaku System Analyst PT Artajasa yang telah memberikan bimbingan dan ilmu-ilmu baru selama penulis melaksanakan magang.
7. Seluruh pengajar Program Studi D3 Komputer dan Sistem Informasi yang telah memberikan ilmu yang bermanfaat selama penulis menempuh masa belajar.
8. Keluarga PT Artajasa yang telah memberikan banyak pengetahuan baru, kesempatan, dan dukungan bagi penulis untuk belajar dan mengembangkan kemampuan penulis hingga Tugas Akhir ini selesai.
9. Sahabat penulis dari awal masuk kuliah, Dimas, Ilham Satriadi, Ilham Karyanto, Iyin yang menjadi keluarga seperantauan.
10. Teman-teman PKL ex-Telkom Sleman, Oliv, Icin, Afi; yang menemani berkeluh kesah dan menampung sambatan-sambatan penulis.
11. Teman-teman satu kontrakan, Dzakwan, Rifal, Wafi yang menemani keseharian penulis hidup di kontrakan.
12. Husni Ramdani, yang menjadi rujukan ketika penulis akan meminta bantuan terkait proyek tugas akhir ini.
13. Seluruh teman KOMSI atas kebersamaan, do'a, dukungan, dan semangat selama masa kuliah hingga pengerjaan tugas akhir selesai.
14. Seluruh pihak yang telah membantu dalam pengerjaan tugas akhir ini yang tidak dapat penulis sebutkan satu-persatu.

Pada akhirnya, penulis sangat menyadari bahwa tugas akhir ini masih jauh dari kata sempurna serta memerlukan pengembangan lebih lanjut lagi. Penulis mewakili apapun yang ada dan tertulis dalam laporan ini memohon maaf atas kesalahan dan ketidak-tepatan informasi yang ada. Maka dari itu, diperlukan kritik dan saran yang membangun demi kemajuan ilmu pengetahuan. Semoga tugas akhir ini bermanfaat bagi pembaca dan penulis.

Yogyakarta, 12 Mei 2020

Penulis

DAFTAR ISI

HALAMAN PENGESAHAN.....	iii
PERNYATAAN BEBAS PLAGIASI	iv
KATA PENGANTAR	v
DAFTAR ISI.....	vii
DAFTAR GAMBAR	x
DAFTAR TABEL.....	xii
INTISARI.....	xiii
ABSTRACT.....	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	2
1.4 Tujuan Penelitian	2
1.5 Manfaat Penelitian	2
1.6 Metodologi Penelitian	3
1.7 Sistematika Penulisan.....	4
BAB II TINJAUAN PUSTAKA.....	6
BAB III LANDASAN TEORI.....	9
3.1 Konsep Dasar Microservice	9
3.1.1 Elemen Dari Microservice	9
3.2 Siklus Pengembangan Sistem.....	9
3.2.1 Komponen Tim Pada Scrum	10
3.2.2 Komponen Tim Pada Scrum	11

3.2.3	Karakteristik Scrum	12
3.2.4	Keuntungan Dari Metodologi Scrum	12
3.3	Remittance	13
3.4	Application Programming Interface	13
3.5	Rest API	14
3.6	Perangkat Lunak Modul	15
3.6.1	Javascript.....	15
3.6.2	ReactJS	15
3.6.3	Golang	16
3.6.4	NATS	16
3.6.5	Docker	17
3.6.6	Kubernetes.....	17
3.6.7	PostgreSQL	17
3.7	Alat Implementasi	18
3.7.1	Visual Studio Code	18
3.7.2	Gitlab.....	18
3.7.3	Postman	19
3.7.4	DBeaver.....	19
BAB IV ANALISIS DAN PERANCANGAN		20
4.1	Analisis Modul	20
4.1.1	Alur Data dan Cara Kerja REST API	20
4.2	Analisis Masalah	22
4.3	Analisis Kebutuhan Fungsional.....	22
4.4	Analisis Kebutuhan Non Fungsional.....	23
4.5	Perancangan Modul	23

4.5.1	Perancangan Basis Data	23
4.5.2	Perancangan Proses	25
4.5.3	Perancangan Tampilan Antarmuka	28
BAB V IMPLEMENTASI.....		32
5.1	Implementasi Perangkat Lunak Pembangun	32
5.2	Implementasi Perangkat Lunak Pembangun	32
5.3	Implementasi Basis Data	33
5.4	Implementasi REST API (<i>Backend Service</i>).....	33
5.5	Implementasi Antarmuka Pengguna.....	49
5.5.1	Halaman Daftar Nilai Tukar Mata Uang.....	49
5.5.2	Halaman Rincian Nilai Tukar Mata Uang	53
5.5.3	Halaman <i>Update</i> Nilai Tukar Mata Uang	55
BAB VI PENGUJIAN DAN PEMBAHASAN		60
6.1	Skenario Pengujian REST API.....	60
6.2	Skenario Pengujian Antarmuka	61
6.3	Hasil Pengujian Modul	62
6.3.1	Hasil Pengujian REST API	62
6.3.2	Hasil Pengujian Antarmuka	68
BAB VII KESIMPULAN DAN SARAN		74
7.1	Kesimpulan.....	74
7.2	Saran	74
DAFTAR PUSTAKA		76

DAFTAR GAMBAR

Gambar 4.1 Alur Data Produk	21
Gambar 4.2 Flowchart daftar semua nilai tukar mata uang	26
Gambar 4.3 Flowchart rincian nilai tukar mata uang.....	27
Gambar 4.4 Flowchart update nilai tukar mata uang	28
Gambar 4.5 Rancangan halaman daftar nilai tukar mata uang	29
Gambar 4.6 Rancangan halaman rincian nilai tukar mata uang.....	30
Gambar 4.7 Rancangan halaman update nilai tukar mata uang	31
Gambar 5.1 Potongan kode mapper daftar nilai tukar mata uang.....	35
Gambar 5.2 Potongan kode repository daftar nilai tukar mata uang.....	35
Gambar 5.3 Potongan kode usecase daftar nilai tukar mata uang.....	36
Gambar 5.4 Potongan kode nats_jobs daftar nilai tukar mata uang.....	36
Gambar 5.5 Potongan kode mapper rincian nilai tukar mata uang	37
Gambar 5.6 Potongan kode repository rincian nilai tukar mata uang.....	37
Gambar 5.7 Potongan kode usecase rincian nilai tukar mata uang.....	38
Gambar 5.8 Potongan kode nats_jobs rincian nilai tukar mata uang	39
Gambar 5.9 Potongan kode mapper update nilai tukar mata uang	40
Gambar 5.10 Potongan kode repository update nilai tukar mata uang	40
Gambar 5.11 Potongan kode usecase update nilai tukar mata uang	41
Gambar 5.12 Potongan kode nats_jobs update nilai tukar mata uang	42
Gambar 5.13 Potongan kode mapper daftar profit.....	43
Gambar 5.14 Potongan kode repository daftar profit.....	43
Gambar 5.15 Potongan kode usecase daftar profit.....	43
Gambar 5.16 Potongan kode nats_jobs daftar profit.....	44
Gambar 5.17 Potongan kode mapper rincian profit	45
Gambar 5.18 Potongan kode repository rincian profit.....	45
Gambar 5.19 Potongan kode usecase rincian profit.....	45
Gambar 5.20 Potongan kode nats_jobs rincian profit.....	46
Gambar 5.21 Potongan kode mapper update profit	47
Gambar 5.22 Potongan kode repository update profit	47
Gambar 5.23 Potongan kode usecase update profit	48
Gambar 5.24 Potongan kode nats_jobs update profit	49
Gambar 5.25 Hasil implementasi halaman daftar nilai tukar mata uang	50
Gambar 5.26 Potongan kode halaman daftar nilai tukar mata uang	51
Gambar 5.27 Potongan kode fungsi halaman daftar nilai tukar mata uang	52
Gambar 5.28 Hasil implementasi halaman rincian nilai tukar mata uang	53
Gambar 5.29 Potongan kode halaman rincian nilai tukar mata uang.....	54
Gambar 5.30 Potongan kode fungsi halaman rincian nilai tukar mata uang	55

Gambar 5.31 Hasil implementasi halaman update nilai tukar mata uang.....	56
Gambar 5.32 Potongan kode halaman update nilai tukar mata uang.....	57
Gambar 5.33 Potongan kode fungsi halaman update nilai tukar mata uang.....	58
Gambar 6.1 Response daftar nilai tukar mata uang	63
Gambar 6.2 Response success rincian nilai tukar mata uang.....	64
Gambar 6.3 Response not found rincian nilai tukar mata uang	64
Gambar 6.4 Response success update nilai tukar mata uang	65
Gambar 6.5 Response not found update nilai tukar mata uang.....	65
Gambar 6.6 Response daftar profit	66
Gambar 6.7 Response success rincian profit.....	67
Gambar 6.8 Response not found rincian profit	67
Gambar 6.9 Response success update profit	68
Gambar 6.10 Response not found update profit.....	68
Gambar 6.11 URL daftar nilai tukar mata uang.....	69
Gambar 6.12 Tampilan halaman daftar nilai tukar mata uang.....	69
Gambar 6.13 URL rincian nilai tukar mata uang	70
Gambar 6.14 Tampilan halaman rincian nilai tukar mata uang	70
Gambar 6.15 URL update nilai tukar mata uang	71
Gambar 6.16 Tampilan halaman update nilai tukar mata uang	72
Gambar 6.17 Contoh masukan nilai tukar mata uang	72
Gambar 6.18 Contoh hasil data sudah diperbaharui	73

DAFTAR TABEL

Tabel 2.1 Perbandingan Tinjauan Pustaka Penelitian	7
Tabel 4.1 Rancangan Tabel rate_aj	24
Tabel 4.2 Rancangan Tabel profit.....	24
Tabel 6.1 Skenario Pengujian REST API	60
Tabel 6.2 Skenario Pengujian Antarmuka	61

INTISARI

**PENGEMBANGAN MODUL NILAI TUKAR MATA UANG
PADA WEB PRODUK PENGIRIMAN UANG KE LUAR
NEGERI**

Oleh:

Elang Bayu Segara

17/416345/SV/14083

Salah satu hal yang menjadi masalah dalam proses transaksi uang yang melibatkan dua negara atau lebih adalah nilai tukar mata uang yang tidak sama di masing-masing negara. Tidak terkecuali pada transaksi antar negara yang berbentuk *remittance*, transaksi ini sangat erat kaitannya dengan nilai mata uang pada negara tujuan. Dalam proses pengiriman uang ini, terdapat berbagai komponen yang berjalan demi menjamin kelancaran dan keamanan transaksi, antara lain yaitu pihak penyedia layanan *remittance* dari suatu negara dan mitra layanan yang berfungsi untuk mengatur alamat tujuan transaksi ke luar negeri.

Untuk mengatur agar transaksi yang dilakukan sudah sesuai dengan semua variabel yang terkait seperti nilai tukar mata uang dan *rate* transaksi pada negara tertentu, maka dibutuhkanlah suatu sistem untuk mengelola informasi variabel tersebut. Berdasarkan permasalahan tersebut, PT Artajasa membangun sistem *remittance* yang di dalamnya terdapat modul yang mampu mengatur informasi nilai tukar mata uang, dan *rate* transaksi dari setiap mitra yang ada. Modul nilai tukar mata uang ini dikembangkan dengan prinsip pengembangan arsitektur *microservices* dan dibangun dengan menggunakan bahasa pemrograman Go untuk bagian *backend* dan *framework* ReactJS untuk tampilan antarmukanya.

Modul yang telah dikembangkan ini mampu memenuhi kebutuhan bisnis dari produk atau layanan *remittance* dari PT. Artajasa serta menyelesaikan masalah yang ada pada proses transaksi uang antar negara tersebut dengan memudahkan pengelolaan informasi nilai tukar mata uang dan *margin profit* yang ada pada tiap negara.

Kata kunci: Nilai tukar mata uang, *microservices*, *remittance*, golang, reactjs

ABSTRACT

DEVELOPMENT OF CURRENCY EXCHANGE RATE MODULE IN REMITTANCE WEB PRODUCT

By:

Elang Bayu Segara

17/416345/SV/14083

One of the problems in the process of money transactions involving two or more countries is the exchange rates that are not the same in each country. No exception to transactions between countries in the form of remittances, these transactions are very closely related to the value of the currency in the destination country. In this money transfer process, there are various components that work to ensure the smoothness and security of transactions, including the remittance service provider and service partners whose function is to set the destination address for overseas transactions.

In order to arrange transactions to be in conformity with all related variables such as currency exchange rates and transaction rates in certain countries, a system is needed to manage the variable information. Based on these problems, PT Artajasa built a remittance system in which there is a module that is able to manage information on currency exchange rates, and transaction rates of each of the existing partners. This currency exchange module was developed with the principle of developing microservices architecture and was built using the Go programming language for the backend section and the ReactJS framework for displaying the interface.

This module is able to meet the business needs of remittance products or services from PT. Artajasa and solve the problems that exist in the process of money transactions between countries by facilitating the management of information on currency exchange rates and profit margins that exist in each country.

Keywords: exchange rate, microservices, remittance, golang, reactjs

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi pada masa sekarang atau lebih kita kenal dengan era industri 4.0 bergerak sangat cepat dan dinamis terutama pada bidang teknologi informasi. Di Indonesia sendiri banyak perusahaan yang bergerak di bidang teknologi informasi saat ini sudah memperbarui sistem dan mengadaptasi teknologi baru dalam mengembangkan suatu produk atau jasa. Salah satu teknologi yang sedang diminati dan menjadi pilihan utama bagi perusahaan tersebut yaitu model arsitektur *microservices*.

Tentu ada alasan mengapa model arsitektur *microservices* ini menjadi idaman dan solusi bagi mayoritas perusahaan teknologi informasi ini. Salah satu alasannya yaitu terletak pada manajemen produk yang jauh lebih rapi dan mudah untuk ditingkatkan lagi ke skala yang lebih besar. Dapat ditarik inti dari model *microservices* ini yaitu menjadikan produk lebih mudah dikembangkan karena akan dibagi dalam beberapa bagian atau yang lebih dikenal dengan modul yang lebih kecil dan setiap modul yang ada relatif independen, sehingga proses pengembangan dan pemeliharaan produk bisa lebih cepat dan tidak mengganggu layanan lain yang sedang berjalan.

Pada perusahaan tempat magang, penulis diberi kesempatan untuk ikut bergabung dalam tim di Yogyakarta untuk mengembangkan suatu produk berupa layanan pengiriman uang dari dan ke luar negeri. Penulis kemudian diberi suatu modul tentang manajemen nilai mata uang dan biaya pada produk layanan tersebut. Pengerjaan tugas ini meliputi bagian Front-End dan Back-End dari modul tersebut. Bagian Front-End dikembangkan dengan framework “ReactJS”, sedangkan di bagian Back-End dikembangkan dengan bahasa pemrograman “Go”.

Permasalahan yang kerap muncul dalam sebuah layanan terutama layanan *remittance* (pengiriman uang dalam valuta asing) adalah pada bagian manajemen kurs dan penyesuaian terhadap biaya pengirimannya. Semakin banyak kurs yang

didukung maka akan semakin menyulitkan jika akan dilakukan pembaruan nilainya, hal ini berbanding lurus dengan nilai biaya yang harus disesuaikan mengikuti perkembangan kurs. Dengan adanya modul ini, diharapkan mampu untuk mempermudah menyesuaikan nilai kurs dan biaya pada produk pengiriman uang dari dan ke luar negeri.

1.2 Rumusan Masalah

Rumusan masalah didapatkan dari latar belakang yang telah dijabarkan sebelumnya, tentang bagaimana mengembangkan sebuah modul dalam produk atau layanan yang memiliki arsitektur *Microservices*.

1.3 Batasan Masalah

Demi menghindari meluasnya permasalahan yang diteliti, maka penulis membatasi permasalahan terkait modul nilai tukar uang dan biaya pada produk pengiriman uang dari dan ke luar negeri menjadi sebagai berikut:

- a. Modul nilai tukar uang dan biaya ini hanya dapat melakukan operasi CRUD (*Create-Read-Update-Delete*) konvensional atau pengolahan data secara biasa dan tidak terlalu kompleks.
- b. Modul tidak dapat melakukan modifikasi terhadap *database*, dengan kata lain hanya dapat melakukan *commands database* jenis DQL (*Data Query Language*) dan DML (*Data Manipulation Language*).

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah mengembangkan suatu modul untuk mengelola nilai tukar uang dan biaya pada produk pengiriman uang dari dan ke luar negeri.

1.5 Manfaat Penelitian

Manfaat dari penelitian modul ini adalah:

- a. Mempermudah pengelolaan nilai tukar uang pada produk pengiriman uang.
- b. Mempermudah pengelolaan biaya pada produk pengiriman uang.
- c. Mensinkronkan data nilai tukar uang dan biaya pada administrator produk dan pengguna.

1.6 Metodologi Penelitian

Pengembangan modul ini melibatkan 2 bagian metodologi penelitian, yaitu metode pengumpulan data dan pengembangan modul.

1. Metode Pengumpulan Data

a. Wawancara

Tahap wawancara dilakukan dengan beberapa anggota tim pengembang meliputi kepala divisi, *frontend programmer*, *backend programmer*, *dev ops*, dan *quality assurance*. Wawancara dimaksudkan agar modul yang dikembangkan penulis memenuhi kebutuhan fungsional, *business logic*, serta standar perusahaan.

b. Studi Literatur

Metode ini dilakukan dengan mencari berbagai panduan serta dokumentasi dari *framework* dan bahasa pemrograman yang digunakan sehingga dapat digunakan sebagai referensi untuk perancangan modul dan analisa alur sistem.

2. Perancangan dan Pengembangan Modul

Pengembangan modul nilai mata uang dan biaya ini menggunakan model pengembangan perangkat lunak yaitu model *Scrum*. Model pengembangan ini merupakan peningkatan dan pembaruan dari beberapa model pengembangan yang berorientasi objek atau iterasi yang sudah umum digunakan. (Schwaber, 1997)

Model pengembangan ini terbagi menjadi 3 tahap¹, antara lain:

¹ Schwaber, Ken. "Scrum development process." Business object design and implementation. Springer, London, 1997. 117-134.

a. Perencanaan dan perancangan

Pada tahap ini dilakukan pemetaan kebutuhan fungsional yang didasarkan pada *business logic* dari modul nilai mata uang dan biaya pada layanan pengiriman uang tersebut serta penentuan durasi pengerjaan modul.

b. *Sprint* pengembangan

Setelah kebutuhan fungsional sudah ditentukan, maka akan masuk pada tahap *Sprint* yaitu mulai mengembangkan modul dengan menggunakan *framework* dan bahasa pemrograman yang sudah ditentukan.

c. Penutupan

Tahap ini merupakan tahap di mana akan dilakukan ulasan dan pengecekan modul yang telah dikembangkan, apakah sesuai dengan rencana dan rancangan pada tahap awal pengembangan.

1.7 Sistematika Penulisan

Penulisan laporan tugas akhir ini tersusun atas tujuh bab. Sistem penulisan laporan diuraikan sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjabarkan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi penjabaran singkat tentang beberapa modul sejenis yang pernah dibuat sebelumnya dan digunakan sebagai referensi dalam penelitian tugas akhir ini.

BAB III LANDASAN TEORI

Bab ini berisi tentang teori, konsep, prinsip dan pendapat yang berasal dari buku dan sumber lain yang terkait dengan pengembangan modul yang dapat dipertanggungjawabkan secara ilmiah. Teori-teori yang dimuat dalam bab ini meliputi teori yang sesuai dan terkait dengan modul yang dikembangkan.

BAB IV ANALISA DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan tahapan-tahapan analisis dan perancangan sistem. Tahapan analisis terdiri atas deskripsi, *usecase* modul, dan alur olah data. Proses perancangan meliputi perancangan tampilan antarmuka pengguna, dan alur olah data.

BAB V IMPLEMENTASI MODUL

Bab ini menjelaskan tentang rincian penerapan atau pengimplementasian dari rancangan modul yang telah dibangun. Rincian implementasi berupa potongan *sourcecode* dan penjelasannya.

BAB VI PENGUJIAN DAN PEMBAHASAN

Pada bab ini membahas proses dan hasil pengujian modul yang telah dibangun, serta integrasi terhadap produk atau layanan pengiriman uang dari dan ke luar negeri.

BAB VII PENUTUP

Bab ini berisi kesimpulan dari pengembangan modul nilai tukar uang dan biaya. Pada bab ini juga terdapat saran untuk penelitian selanjutnya.

DAFTAR PUSTAKA

Pada bab ini memuat sumber literatur yang digunakan penulis sebagai acuan dalam mengembangkan modul.

LAMPIRAN

BAB II

TINJAUAN PUSTAKA

Guna mendapatkan hasil penelitian yang memiliki dasar yang kuat, maka dilakukanlah tinjauan pustaka yang mengacu pada beberapa sumber yang dapat dipertanggung-jawabkan kredibilitasnya. Di bawah tertera daftar sumber pustaka yang terkait dengan pengembangan modul nilai mata uang dan biaya dengan model *scrum* yang berbasis arsitektur *microservices*.

Dzarrin (2019), mengembangkan layanan yang berbasis arsitektur *Microservice* untuk manajemen tanda tangan digital di Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada. Layanan ini menggunakan *framework* ASP.NET dan SQL sebagai intinya, dan menghasilkan API yang dapat diakses oleh berbagai aplikasi web. Dengan memilih arsitektur *Microservices* ini maka akan memudahkan pengembang web yang ingin menggunakan layanan tanda tangan digital ini hanya dengan mengakses dan menggunakan API yang telah disediakan.

Utomo (2019), mengembangkan aplikasi IoT untuk memantau kualitas tidur seseorang. Platform IoT ini menggunakan arsitektur *Microservices* dan berbasis *event-driven*. Aplikasi pemantau kualitas tidur ini dibangun dengan menggunakan prinsip *containerization* sehingga akan membuat aplikasi mampu berjalan pada sistem operasi server apapun dengan *environment* yang tetap sama.

Jufry (2018), merancang dan mengembangkan arsitektur *Microservices* pada *Online Travel Agent*. Implementasi dari perancangan ini menggunakan bahasa pemrograman Microsoft .NET, UML, serta menggunakan Amazon Web Services sebagai layanan *cloud*. Hasil akhir dari penelitian ini menunjukkan bahwa lalu lintas data yang terjadi pada *Online Travel Agent* menjadi cepat dan mampu menangani jumlah *request* yang besar dengan relatif mudah. Hal ini dapat diartikan bahwa efisiensi dan efektivitas dari arsitektur *Microservices* ini berdampak besar pada suatu pengembangan produk.

Mengacu pada tinjauan pustaka di atas, penulis membangun modul nilai tukar mata uang dan biaya pada produk pengiriman uang dari dan ke luar negeri menggunakan *framework* ReactJS pada bagian *frontend* dan bahasa pemrograman Go pada bagian *backend*, serta menggunakan beberapa *tools* dalam komunikasi tersebut antara lain NATS sebagai *message broker* dan PostgreSQL sebagai basis data. Modul ini digunakan untuk mempermudah dalam mengatur nilai tukar mata uang dan biaya pada layanan pengiriman uang dari dan ke luar negeri. Modul ini jika pada tampilan antarmuka pengguna akan berupa tabel nilai mata uang dan biaya transfer dari semua negara yang didukung oleh layanan, sedangkan pada bagian *backend* akan berupa alur koneksi antara basis data dan antarmuka pengguna.

Perbandingan pada masing-masing penelitian dan proyek yang sudah dilakukan oleh para peneliti di atas dapat dicermati pada Tabel 2.1.

Tabel 2.1 Perbandingan Tinjauan Pustaka Penelitian

Penulis	Judul	Teknologi	Deskripsi
Nuwas Dzarrin T	Rancang Bangun Mikroservis Sebagai Pembangkit Tanda Tangan Digital Berbasis Cloud Computing	ASP.NET, SQL Server,\	<i>Service</i> atau layanan untuk menangani pengelolaan data tanda-tangan digital yang terhubung dengan <i>cloud</i> .
Oei Kurniawan Utomo	Arsitektur Platform IoT Untuk Pemantauan Kualitas Tidur Berbasis Microservices dan Event-driven	Internet of Things, Message Broker, Web App (Dashboard)	<i>Dashboard</i> dan layanan (<i>service</i>) yang berguna untuk memantau kualitas tidur seseorang.

Fachry Jufry	Perancangan Arsitektur Microservices Untuk Online Trvael Agent Pada PT. XYZ	Microsoft .NET, UML, dan Amazon Web Services	<i>Service engine</i> sebagai <i>endpoint</i> untuk memesan perjalanan wisata secara <i>online</i> .
--------------	--	---	--

BAB III

LANDASAN TEORI

3.1 Konsep Dasar Microservice

Larrucea, Santamaria, Colomo-Palacios, dan Ebert (2018) menyatakan bahwasannya *microservice* adalah suatu arsitektur aplikasi yang terdiri dari komponen-komponen yang dikelompokkan sesuai fungsi tertentu. Komponen-komponen tersebut berkomunikasi satu sama lain melalui jalur yang disebut *messages* (biasanya berupa API, maupun Rest API). Jenis arsitektur ini sendiri sejatinya merupakan bentuk penguraian dari arsitektur jenis *monolith* yang fungsinya tidak dapat berjalan secara independen. Hal yang dapat diharapkan dari pengembangan aplikasi menggunakan arsitektur *microservice* ini adalah berpusat pada efisiensi dan efektifitas mulai dari sumber daya manusia hingga sumber daya *digital*.

3.1.1 Elemen Dari Microservice

Yousif (2016) menuliskan bahwa *Microservice* dapat juga diartikan sebagai aplikasi yang komponen-komponennya bersifat modular. Karena *microservice* dibangun oleh komponen yang modular maka layanan yang ada di dalam arsitektur ini disebut dengan modul. Tiap modul memiliki fungsi dan spesifikasi yang berbeda dikelompokkan dan dibangun sesuai dengan kebutuhan bisnis.

3.2 Siklus Pengembangan Sistem

Jogiyanto (2005) menjabarkan bahwa pengembangan sistem pembuatan sistem baru yang lebih mutakhir dengan tujuan mengganti dan memperbaiki sistem yang sudah ada. Termasuk di dalamnya adalah metode pengembangan berjenis *agile development*. *Agile development* memiliki beberapa turunan lagi, salah satunya yaitu *Scrum*. Schwaber, dan Ken (1997) menjabarkan bahwa metode

pengembangan *Scrum* ini adalah manajemen, pemutakhiran, dan pemeliharaan untuk sebuah sistem yang sudah ada maupun yang masih dalam tahap prototipe produksi.

3.2.1 Komponen Tim Pada Scrum

Menurut Schwaber, dan Ken (1997), metode pengembangan *Scrum* dibagi menjadi 3 fase besar, yaitu:

1) Pregame

a. Planning

Pendefinisian tentang rilis baru berdasarkan *backlog* yang diketahui sekarang, dengan menyertakan estimasi dari jadwal serta biaya dari produk yang akan dikembangkan. Jika sedang mengembangkan produk baru, fase ini terdiri dari konseptualisasi dan analisis. Namun, jika sedang memperbaiki dan memutakhirkan produk yang sudah ada, maka fase ini terdiri dari analisis terbatas.

b. Architecture

Merancang bagaimana item yang sudah ada di *backlog* akan diimplementasikan. Fase ini meliputi modifikasi arsitektur produk atau sistem dan rancangan desain level tinggi.

2) Game

a. Development Sprints

Fase pengembangan dari perilsan fungsi baru, dengan tetap mengacu pada variabel waktu, kebutuhan, kualitas, biaya, dan kompetisi. Kronologis interaksi dengan variabel-variabel ini menentukan akhir dari fase ini. *Sprint* juga dapat diartikan sebagai aktivitas pengembangan yang dilakukan dengan jangka waktu yang sudah ditentukan sebelumnya oleh *Lead Developer*, biasanya interval dari *sprint* ini berkisar antara satu hingga empat minggu. Penentuan interval ini didasarkan pada kompleksitas produk, resiko yang dapat terjadi, dan tingkat pengawasan yang diinginkan. Di dalam *sprint* sendiri, terdapat beberapa tahap aktivitas seperti:

- *Develop*: Menentukan perubahan yang dibutuhkan untuk pengimplementasian dari kebutuhan *backlog* ke dalam modul, inisiasi modul, melakukan analisis domain, perancangan, pengembangan, implementasi, pengujian, dan pendokumentasian perubahan. Tahap *develop* sejatinya terdiri dari proses mikro dari sebuah penemuan, dan implementasi.

- *Wrap*: Menutup pengembangan modul, membuat versi *executable* dari perubahan yang sudah dikembangkan dan mendeskripsikan bagaimana cara mereka mengimplementasikan kebutuhan *backlog*.

- *Review*: Seluruh anggota tim bertemu dan mempresentasikan dan mengulas pekerjaan mereka, mengangkat dan menyelesaikan permasalahan yang muncul selama pengembangan, menambahkan item baru ke *backlog*.

- *Adjust*: Mengonsolidasikan informasi yang dikumpulkan dari pertemuan tersebut ke dalam modul yang dimaksudkan, termasuk perbedaan tampilan, dan properti baru.

3) Postgame

a. Closure

Ketika tim pengembang merasa bahwa variabel dari waktu, kompetisi, kebutuhan bisnis, biaya, dan kualitas sudah setuju untuk dirilis, mereka mendeklarasikan pengembangan dari rilis ini ditutup dan memasuki fase ini. Fase ini mempersiapkan produk yang sudah dikembangkan untuk dirilis secara umum. Aktivitas yang ada pada fase ini meliputi integrasi, pengujian sistem, dokumentasi untuk pengguna, persiapan material pelatihan, dan material pemasaran.

3.2.2 Komponen Tim Pada Scrum

- Manajemen: Dipimpin oleh seorang *Product Manager*, mendefinisikan konten atau kebutuhan awal dan jadwal dari produk yang akan dirilis, lalu mengelola perubahan mereka ketika proyek berjalan dan variabel berubah. Manajemen mengatur *backlog*, resiko, dan konten atau produk yang akan dirilis.

- Tim pengembang: Tim pengembang relatif sedikit, dengan terdiri dari pengembang, dokumenter, dan staf kontrol kualitas. Para pengembang dapat dibagi lagi menjadi beberapa tim yang terdiri dari tiga hingga enam orang. Setiap tim ditugaskan untuk mengembangkan suatu modul spesifik, termasuk *backlog* yang terkait dengan modul tersebut. Tim pengembang dikelompokkan berdasarkan keahlian dan spesifikasi masing-masing.

3.2.3 Karakteristik Scrum

Proyek *scrum* memiliki karakteristik sebagai berikut:

- Keluaran yang fleksibel – isi dari keluaran yang dihasilkan ditentukan oleh *environment*.
- Jadwal yang fleksibel – keluaran yang dihasilkan terkadang dapat dibutuhkan lebih cepat atau lebih lama dari yang sudah direncanakan.
- Tim yang kecil – tiap tim terdiri tidak lebih dari 6 anggota. Sangat dimungkinkan ada beberapa tim dalam suatu proyek.
- Ulasan yang sering – *progress* tim diulas sesering mungkin sesuai kompleksitas proyek (biasanya satu hingga empat minggu dalam satu siklus).
- Kolaborasi – intra dan inter-kolaborasi sangat diharapkan selama pengerjaan proyek.
- Berorientasi Objek – setiap tim akan diberi tugas dengan objek terkait, dengan spesifikasi dan kebutuhan yang jelas.

3.2.4 Keuntungan Dari Metodologi Scrum

Metodologi *scrum* dirancang sedemikian rupa untuk benar-benar fleksibel selama proses pengembangan produk. Metode ini menyediakan mekanisme kontrol untuk perencanaan rilis produk, lalu mengelola variabel selama proses pengembangan berjalan. Hal ini memungkinkan organisasi atau perusahaan untuk

mengubah proyek dan hasil keluaran, pada waktu kapanpun, demi memastikan rilis produk yang paling dikehendaki.

Metodologi *scrum* membebaskan pengembang untuk merancang dan menerapkan solusi terbaik yang mereka punyai terhadap proyek yang sedang dikerjakan, ini juga dikarenakan terjadinya pembelajaran di lingkup pengembang itu sendiri.

Teknologi yang berorientasi objek adalah basis dari metodologi *scrum*. Objek, fitur, atau kebutuhan bisnis, menawarkan lingkungan pengembangan yang terpisah dan mudah untuk dikelola.

3.3 Remittance

Untuk mengetahui asal mula dari remittance atau transfer dana ini, kita dapat mengutip dari jurnal yang ditulis oleh Anggraeni Primawati (2017) yang berjudul “Remitan Sebagai Dampak Migrasi Pekerja Ke Malaysia”, istilah remitan pada awalnya adalah uang maupun barang yang dikirim ke daerah asal oleh seorang migran yang masih berada di tempat tujuan. Definisi dari remitan ini mengalami perluasan seiring waktu menjadi tidak hanya uang dan barang saja, namun keterampilan dan ide juga termasuk remitan daerah asal.

Pada penelitian ini, transfer dana didefinisikan sebagai kegiatan yang bertujuan memindahkan sejumlah dana yang berasal dari pengirim kepada penerima dengan memanfaatkan jasa penyelenggara berupa lembaga bank maupun lembaga keuangan bukan bank. Transfer dana ini dapat dilakukan mulai dari antar individu maupun perusahaan. Sedangkan *remittance* merupakan bagian dari transfer dana yang umumnya dilakukan tanpa dasar/*underlying* pemenuhan suatu kewajiban ekonomi, bernilai kecil/*low value* dan dilakukan antar perorangan.

3.4 Application Programming Interface

Application Programming Interface atau yang lebih dikenal dengan singkatannya yaitu API merupakan kumpulan definisi dan protokol untuk membangun dan mengintegrasikan perangkat lunak aplikasi. API memungkinkan produk atau layanan saling berkomunikasi dengan produk dan layanan lainnya

tanpa harus mengetahui bagaimana penerepannya. API adalah cara sederhana untuk menghubungkan suatu infrastruktur melalui pengembangan *cloud-native*, serta memungkinkan untuk mendistribusikan dan berbagi data dengan pelanggan atau pengguna dan sumber eksternal lainnya.

3.5 Rest API

Rest API adalah web API yang menggunakan konsep dari arsitektur REST (*Representational State Transfer*). Karena Rest API adalah turunan dari arsitektur REST, maka tidak ada standar resminya. Didefinisikan oleh Roy Fielding (2000), bahwa API sudah tergolong Rest API selama memenuhi enam syarat yang dimiliki oleh sistem yang berarsitektur REST, yaitu:

- *Client-Server*

Arsitektur REST terdiri dari klien, server, dan sumber daya yang dihubungkan dengan *requests* melalui HTTP.

- *Stateless*

Tidak ada konten informasi klien yang disimpan di server saat *requests*. Informasi tentang sesi *requests* tetap disimpan oleh klien.

- *Cache*

Metode *caching* mampu mengeliminasi kebutuhan untuk terus berinteraksi antara klien dengan server.

- *Uniform Interface*

Dengan menerapkan prinsip generalisasi pengembangan aplikasi terhadap *interface* komponen, keseluruhan sistem arsitektur menjadi sederhana dan visibilitas interaksi meningkat.

- *Layered System*

Interaksi antara klien dengan server dapat diberi mediasi dengan *layer* tambahan. *Layer* tambahan ini dapat berupa *load balancing*, dan keamanan lanjutan.

- *Code on Demand*

Server dapat menambahkan ekstra fungsionalitas kepada klien dengan mengirim kode *executable*.

3.6 Perangkat Lunak Modul

Dalam mengembangkan modul nilai tukar mata uang dan biaya ini, penulis menggunakan beberapa *framework* dan layanan pendukung.

3.6.1 Javascript

JavaScript adalah bahasa pemrograman yang memungkinkan kita untuk mengimplementasikan fitur yang kompleks ke dalam sebuah halaman web seperti menampilkan konten yang diperbaharui sesuai waktu, peta interaktif, animasi 2 dimensi maupun 3 dimensi, dan lain sebagainya.

JavaScript sering dikaitkan dengan Java, namun kenyatannya JavaScript dan Java adalah dua hal yang berbeda. Penamaan JavaScript murni hanya dari aspek *marketing* oleh Netscape dan Sun (pertama kali disebut LiveScript, namun saat mendekati perilisan, mereka menggantinya dengan nama JavaScript).

JavaScript sendiri sudah melalui banyak perkembangan sejak pertama dirilis pada tahun 1995. Organisasi yang menetapkan standar untuk JavaScript yaitu ECMA (European Computer Manufacturers Association). Versi JavaScript yang paling populer digunakan untuk saat ini adalah versi ES2015 atau ES6.

3.6.2 ReactJS

ReactJS merupakan pustaka/*library* JavaScript yang digunakan untuk mengembangkan tampilan antarmuka pengguna. ReactJS dikembangkan dan dirilis oleh Facebook pada tahun 2013 dan bersifat *open source*. Jika dalam konsep aplikasi MVC (Model View Controller), ReactJS adalah *view layer* dari aplikasi tersebut. ReactJS memiliki fitur unggulan yaitu:

- Deklaratif

Yang dimaksud dari deklaratif yaitu dapat membuat tampilan antarmuka yang interaktif, sehingga memudahkan untuk membuat desain untuk setiap *state* dalam aplikasi.

- Berbasis komponen

ReactJS dibangun dengan prinsip modular, yang diimplementasikan dengan elemen kode yang dapat digunakan dan disesuaikan di manapun. Komponen dapat ditulis sekali dan diimpor di manapun jika dibutuhkan.

3.6.3 Golang

Bahasa pemrograman Go adalah bahasa pemrograman yang secara sintaks mirip seperti bahasa pemrograman C, namun ditambah dengan *memory safety*, *garbage collection*, penyusunan struktural, dan *concurrency*. Go dikelola dan dirilis pertama kali tahun 2009 di Google oleh Robert Griesemer, Rob Pike, dan Ken Thompson. Go dikenal cepat untuk menangani tugas-tugas berat karena secara prinsip sudah menerapkan *concurrency*. *Concurrency* adalah bagaimana cara untuk menghadapi banyak tugas atau pekerjaan sekaligus. *Concurrency* dalam Go diterapkan dengan menggunakan *goroutine*.

3.6.4 NATS

NATS merupakan sistem *messaging* atau yang lebih dikenal umum sebagai *message broker* yang sederhana, aman, dan memiliki performa tinggi untuk aplikasi *cloud native*, IoT, dan arsitektur *microservices*. NATS dibuat pertama kali pada tahun 2010 oleh Derek Collison. Inti dari NATS adalah sistem yang berupa *Publish/Subscribe (PubSub)*. Model *messaging* ini memungkinkan klien yang berada dalam sistem berkomunikasi tanpa harus menentukan letak akses *endpoint* pasti dari sebuah layanan yang ada di jaringan. Klien menjadi *subscribers* (atau konsumen) dengan mendaftarkan ketertarikannya ke sebuah subjek, lalu ketika seorang *publisher* (atau produsen) menyebarkan pesan baru ke subjek tersebut, maka NATS akan mengirim pesan itu ke semua *subscriber* yang sudah mendaftarkan pada topik atau subjek tersebut.

3.6.5 Docker

Docker merupakan teknologi virtualisasi *container*, dan dapat diartikan juga seperti VM (*Virtual Machine*) yang ringan. Untuk dapat menjabarkan Docker kita harus mengetahui dahulu konsep *container*. Mengutip dari Merkel (2014), *container* merupakan virtualisasi dari sebuah aplikasi yang mencakup hingga level sistem operasi, sehingga *container* ini mampu dijalankan pada sistem operasi apapun dengan *environment* yang sama dan tidak terpengaruh oleh jenis sistem operasinya selama sistem operasi tersebut didukung oleh Docker. Docker memecahkan beberapa masalah yang sering dijumpai saat pengembangan aplikasi seperti konflik *dependencies*, instalasi *dependencies*, dan perbedaan platform sistem operasi dengan cara mengemas suatu program ke dalam sebuah *container* yang mampu dijalankan pada semua platform yang didukung oleh Docker.

3.6.6 Kubernetes

Menurut Bernstein (2014), Kubernetes merupakan pengelompokan dan pengelolaan *container* aplikasi berdasarkan detail sistem yang dijalankan. Kubernetes juga bertugas untuk penjadwalan *container* untuk mengakses sumber daya yang dibutuhkan. Dalam konsep Kubernetes, setiap kelompok *container* disebut dengan Pod dan memiliki alamat IP yang unik dan dapat diakses oleh Pod lain dalam satu kluster. Kubernetes utamanya dikembangkan oleh Google, dan didukung oleh Microsoft, VMware, IBM, dan Red Hat.

3.6.7 PostgreSQL

Menurut Drake dan Worsley (2002) pada bukunya yang berjudul “Practical PostgreSQL”, PostgreSQL adalah sistem manajemen basis data *object-relational* yang telah dikembangkan dalam beberapa model sejak tahun 1977. Bermula dari sebuah proyek yang bernama Ingres di University of California di Berkeley, pada tahun 1986, Michael Stonebreaker menggunakan kode dari Ingres untuk membuat sistem basis data *object-relational*. Postgre menjadi *open source* pada tahun 1996 dan berubah nama menjadi PostgreSQL. PostgreSQL menjadi sangat

populer di dunia basis data karena memiliki fitur yang sangat mutakhir yang kebanyakan hanya dapat ditemukan di basis data komersil seperti Oracle. Salah satu fitur unggulan dari PostgreSQL yaitu deklaratif kueri SQL, dukungan untuk *concurrency*, mendukung multi pengguna, dan optimasi kueri.

3.7 Alat Implementasi

Untuk membantu implementasi dari hasil perencanaan, analisis, dan desain modul nilai tukar mata uang dan biaya pada produk layanan pengiriman uang ini, penulis menggunakan beberapa alat yang dapat disimak penjelasan singkatnya di bawah ini.

3.7.1 Visual Studio Code

Visual Studio Code adalah editor kode gratis yang mendukung *debugging*, pengelolaan repositori Git, *syntax highlighting*, dan penyelesaian saran kode yang pintar. Keunggulan utama dari Visual Studio Code adalah adanya fitur ekstensi yang berguna untuk mengganti tema, dan pintasan *keyboard* serta dapat menambahkan fitur baru yang belum ada. Visual Studio Code dirilis pertama kali tahun 2015 oleh Microsoft dan bersifat *open source*. Menurut survey yang dilakukan Stack Overflow pada tahun 2019, Visual Studio Code menjadi editor kode yang paling populer dengan 50,7% dari 80 ribu lebih responden yang menggunakannya.

3.7.2 Gitlab

Gitlab merupakan salah satu VCS (*Version Control System*) yang bersifat *open source* dan dapat digunakan serta diinstal mandiri pada server perusahaan sehingga memudahkan untuk pengelolaan kode pada pengembangan produk atau aplikasi secara internal perusahaan. Gitlab pertama kali dirilis pada tahun 2014.

3.7.3 Postman

Postman merupakan sebuah aplikasi yang berfungsi sebagai klien REST untuk menguji REST API yang sudah dikembangkan. Postman memiliki keunggulan utama dibandingkan dengan aplikasi sejenis lainnya karena memiliki fitur *workspace* yang memungkinkan para pengembang untuk berkolaborasi dan menguji API secara *real-time*.

3.7.4 DBeaver

DBeaver merupakan alat manajemen basis data gratis dan *open source* yang mendukung berbagai macam jenis basis data seperti MySQL, PostgreSQL, SQLite, dan lain sebagainya. DBeaver dapat dijalankan pada sistem operasi Linux, Windows, ataupun Mac. DBeaver pertama kali dirilis pada tahun 2010.

BAB IV

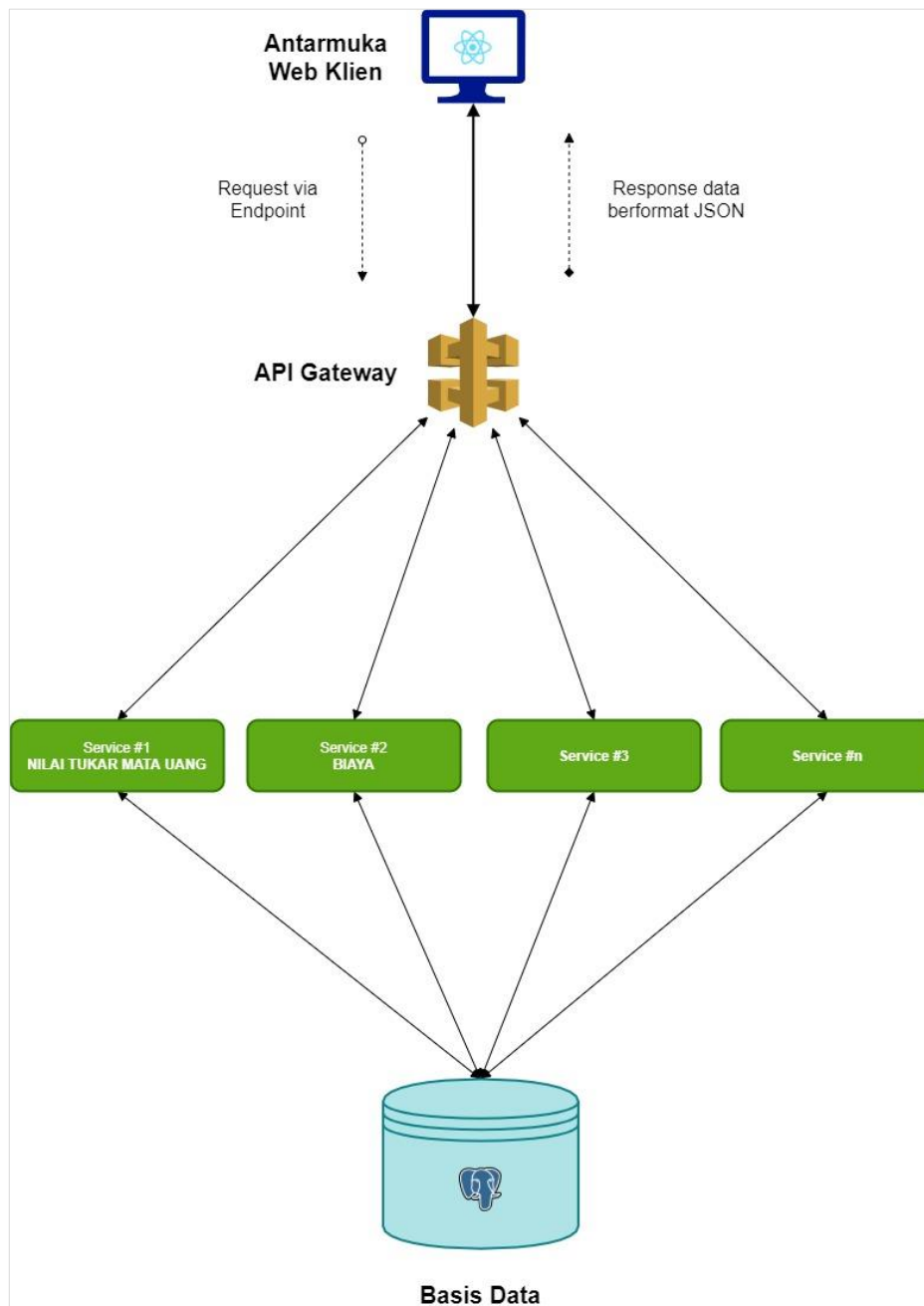
ANALISIS DAN PERANCANGAN

4.1 Analisis Modul

Idealnya, dalam proses pengembangan suatu modul ataupun sistem, dibutuhkan tahap analisis, yaitu tahap untuk mendalami dan memetakan masalah-masalah yang terjadi. Hal ini berguna untuk mengetahui pendekatan dan solusi apa yang akan digunakan dalam modul atau sistem untuk menyelesaikan masalah dan memenuhi kebutuhan tersebut. Dalam tahap ini dilakukan analisis terhadap masalah produk atau layanan, kebutuhan fungsional, dan non fungsional modul.

4.1.1 Alur Data dan Cara Kerja REST API

Dalam mengelola data yang ada, modul ini menggunakan komunikasi dengan memanfaatkan *endpoint* yang dihasilkan oleh sebuah *backend service*. *Backend service* ini sendiri tidak lain merupakan sebuah *gateway API* yang bertugas menghubungkan antarmuka pengguna dengan *service* pengelola data. *Backend service* ini bekerja dengan cara mendapatkan *request* dari pengguna melalui antarmuka web kemudian mengembalikan *response* berupa data sesuai dengan yang diminta. *Endpoint* digunakan oleh antarmuka web sebagai alamat untuk meminta suatu data atau aksi tertentu yang akan dijalankan oleh *backend service*. Aksi yang dapat dilakukan oleh pengguna meliputi GET dan POST data ke basis data. Untuk melihat secara jelas bagaimana alur data dan komunikasi yang terjadi antara antarmuka web dan *backend service* dapat dilihat pada Gambar 4.1.



Gambar 4.1 Alur Data Produk

Dari Gambar 4.1, dapat kita lihat bahwa klien (melalui antarmuka web) mengakses data dengan cara mengirim *request* menggunakan protokol HTTP baik dalam metode GET atau UPDATE yang akan diterima oleh API *gateway* lalu akan dihubungkan ke *service* nilai tukar mata uang untuk menjalankan *query* pada basis data. Setelah *query* berhasil dilakukan, *service* akan mengirimkan *response* berupa

data dalam format JSON yang nantinya akan diteruskan oleh API *gateway* kepada klien melalui antarmuka web.

4.2 Analisis Masalah

Pada dasarnya, model bisnis dari perusahaan tempat penulis magang yaitu PT Artajasa adalah *Business to Business* atau lebih kita kenal sebagai B2B yang berarti bahwa produk yang dihasilkan akan digunakan oleh perusahaan lain yang menjadi relasi dari produk atau layanan yang dihasilkan. Pada produk atau layanan pengiriman uang dari dan ke luar negeri ini, PT Artajasa menjadi perantara dan penyedia layanan transaksi antara mitra (*collecting agent*) dan bank tujuan pengiriman uang. Permasalahan yang timbul ketika kita mengirim uang terutama antar negara adalah informasi nilai tukar mata uang atau yang sering disebut kurs. Tentunya setiap negara memiliki kurs yang berbeda, dan untuk memudahkannya dalam mengelola informasi kurs tersebut, dibuatlah suatu modul untuk memudahkan pengelolaan informasi tersebut.

Setelah melalui forum diskusi bersama tim pengembang dari PT Artajasa, disetujui untuk membuat suatu modul yang berfungsi mengelola informasi nilai tukar mata uang dari negara-negara yang didukung oleh produk layanan pengiriman uang tersebut. Modul ini juga memiliki suatu informasi yaitu margin (selisih) yaitu selisih nilai tukar mata uang. Margin ini sendiri digunakan produk untuk menentukan partner layanan mana yang akan digunakan untuk transaksi berdasarkan besaran margin dan nilai keuntungan (profit). Modul ini memiliki bagian antarmuka pengguna yang merupakan panel untuk mengelola informasi, serta bagian *backend* untuk menjalankan perintah yang dimasukkan oleh pengguna, mulai dari menampilkan data dari basis data, serta memperbaharui data yang ada di basis data.

4.3 Analisis Kebutuhan Fungsional

Berdasarkan hasil analisis masalah di atas, modul nilai tukar mata uang ini memiliki kebutuhan fungsional yaitu meliputi:

- 1) Modul dapat menampilkan semua data berupa nilai tukar mata uang, margin, dan profit
- 2) Modul dapat menampilkan rincian salah satu data nilai tukar mata uang, margin, dan profit sesuai *id* negara masukan dari pengguna.
- 3) Modul dapat memperbaharui *value* salah satu data nilai tukar mata uang, margin, dan profit sesuai *id* negara masukan dari pengguna.

4.4 Analisis Kebutuhan Non Fungsional

Setelah mengetahui hasil analisis kebutuhan fungsionalnya, maka akan dilakukan analisis kebutuhan non fungsional. Kebutuhan non fungsional dapat diartikan sebagai hal pendukung sistem atau modul agar mampu berjalan dengan baik. Modul ini memiliki kebutuhan non fungsional sebagai berikut:

1. Modul membutuhkan koneksi ke jaringan internet.
2. Modul hanya dapat diakses di dalam aplikasi antarmuka pengguna.
3. Modul hanya dapat diakses dan digunakan ketika pengguna sudah berhasil masuk ke dalam antarmuka pengguna.
4. Implementasi REST API diakses melalui *endpoint* yang akan diproses via antarmuka pengguna.

4.5 Perancangan Modul

Untuk mendapatkan gambaran yang jelas dan mempermudah proses pengembangan, maka dilakukanlah perancangan modul. Perancangan ini berfokus pada tabel basis data dan tampilan tabel antarmuka pengguna.

4.5.1 Perancangan Basis Data

Pada implementasi modul nilai tukar mata uang pada produk layanan pengiriman uang dari dan ke luar negeri ini disediakan basis data yang sudah dikembangkan oleh tim pengembang di pusat untuk dapat diakses melalui *backend service* nantinya. Adapun tabel yang digunakan dan diakses oleh modul ini

meliputi, tabel *rate_aj*, dan tabel *profit*. Struktur detail dari tabel-tabel tersebut dapat dilihat pada tabel di bawah ini.

1. Tabel *rate_aj*

Tabel *rate_aj* adalah tabel yang memuat data nilai tukar mata uang. Tabel *rate_aj* terdiri dari atribut *destination_currency*, *rate_idr_dest*, dan *margin*. Rincian tabel *rate_aj* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Rancangan Tabel *rate_aj*

Atribut	Tipe Data	Keterangan
<i>destination_currency</i>	string	Kode negara tujuan maupun asal transaksi.
<i>rate_idr_dest</i>	string	Nilai kurs mata uang negara tujuan/asal.
<i>margin</i>	integer	Selisih nilai tukar mata uang.

2. Tabel *profit*

Tabel *profit* adalah tabel yang memuat data keuntungan minimal dari kurs mata uang setiap negara yang didukung oleh produk atau layanan pengiriman uang ini. Tabel *profit* terdiri dari atribut *currency_destination*, dan *min_profit*. Rincian tabel *profit* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Rancangan Tabel *profit*

Atribut	Tipe Data	Keterangan
<i>currency_destination</i>	string	Kode negara tujuan maupun asal transaksi.
<i>min_profit</i>	float	Nilai keuntungan minimal dari transaksi pada mata uang negara yang diacu.

4.5.2 Perancangan Proses

Demi terwujudnya modul dan fungsi yang dibutuhkan sesuai hasil analisis, maka dilakukanlah perancangan proses. Perancangan proses merupakan penjelasan gambaran alur yang berjalan dalam modul nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri ini. Perancangan proses dibuat dengan menggunakan model UML dan digambarkan dengan bagan alir (*flowchart*).

Bagan alir (*flowchart*) berguna untuk penggambaran alur proses pengolahan data yang terjadi pada modul ini, yaitu memperlihatkan bagaimana alur algoritma yang benar, mulai dari masukan perintah hingga keluaran data. Di bawah ini merupakan deksripsi dari setiap proses pengolahan data dalam modul nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri.

1) *Flowchart* daftar semua nilai tukar mata uang.

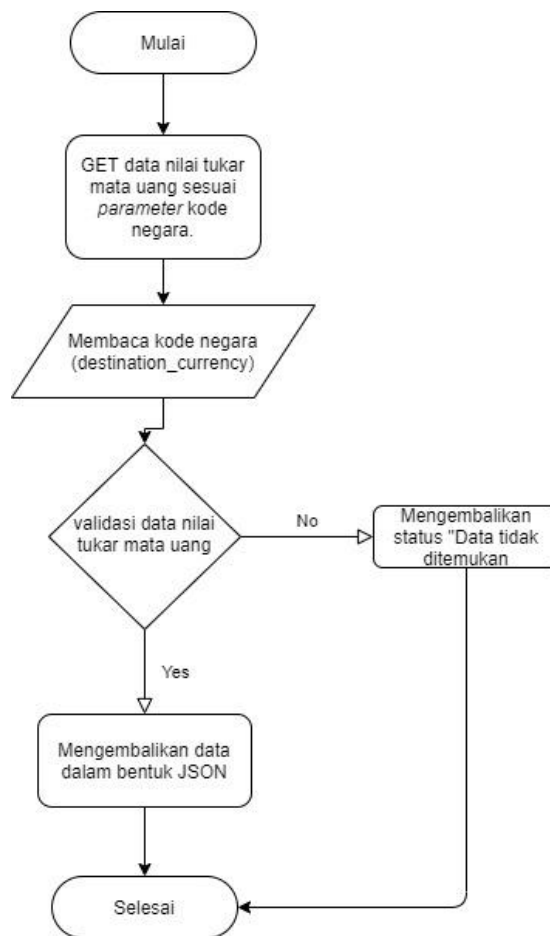
Proses API untuk menampilkan daftar nilai tukar mata uang tidak memerlukan masukan *parameter* apapun. Saat *endpoint* API dipanggil, maka *backend service* akan melakukan *request* dengan tipe GET yang selanjutnya akan melakukan *query* SELECT ALL terhadap tabel *rate_aj* dan *profit* yang nantinya akan mengembalikan hasil *query* dalam bentuk JSON.



Gambar 4.2 Flowchart daftar semua nilai tukar mata uang

- 2) Flowchart menampilkan rincian data nilai tukar mata uang sesuai kode negara (*destination_currency*).

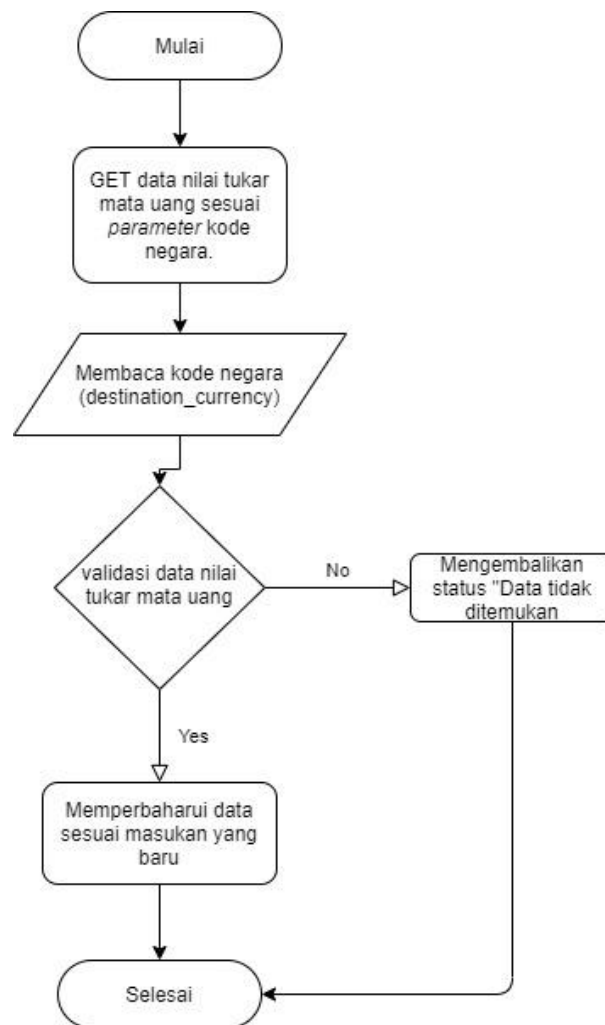
Proses API untuk menampilkan rincian salah satu nilai tukar mata uang berdasarkan kode negara memerlukan suatu *parameter* yaitu kode negara (*destination_currency*). Saat *endpoint* API dipanggil, maka *backend service* akan melakukan *request* dengan tipe GET yang selanjutnya akan melakukan *query* SELECT WHERE terhadap tabel *rate_aj* dan *profit* yang nantinya akan mengembalikan hasil *query* dalam bentuk JSON.



Gambar 4.3 Flowchart rincian nilai tukar mata uang

- 3) Flowchart memperbaharui data nilai tukar mata uang sesuai kode negara (*destination_currency*).

Proses API untuk memperbaharui salah satu nilai tukar mata uang berdasarkan kode negara memerlukan suatu *parameter* yaitu kode negara (*destination_currency*). Saat *endpoint* API dipanggil, maka *backend service* akan melakukan *request* dengan tipe POST yang selanjutnya akan melakukan *query* UPDATE WHERE terhadap tabel *rate_aj* dan *profit* yang nantinya akan mengembalikan hasil *query* dalam bentuk JSON.



Gambar 4.4 Flowchart update nilai tukar mata uang

4.5.3 Perancangan Tampilan Antarmuka

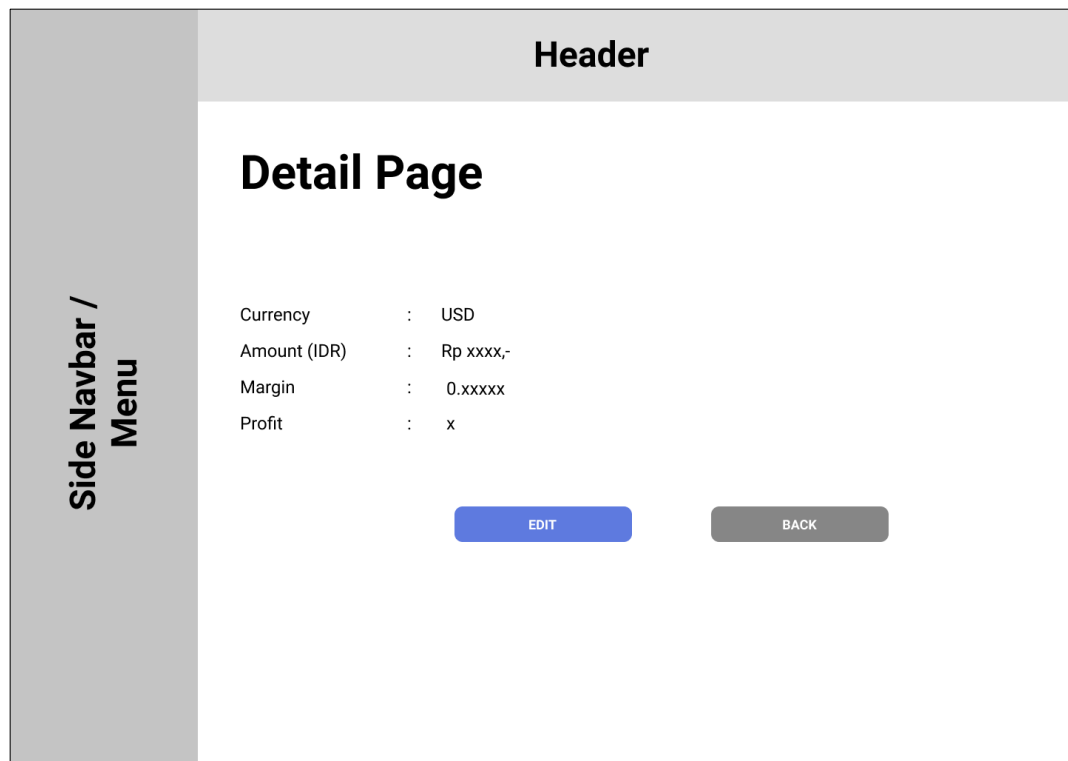
Antarmuka merupakan aspek yang penting karena merupakan hal yang akan digunakan oleh pengguna untuk menjalankan modul nilai tukar mata uang tersebut. Antarmuka pengguna ini berupa tabel yang berisi daftar nilai tukar mata uang, rincian nilai tukar mata uang per kode negara, serta halaman *edit* data yang berupa *form*. Rancangan tabel ini dibuat menggunakan aplikasi desain tampilan antarmuk Figma.

1. Rancangan tampilan tabel daftar nilai tukar mata uang.

Side Navbar / Menu	Header				
	Currency	Amount (IDR)	Margin	Profit	Action
	Currency #1	Rp xxxxx,-	0.xxxxx	x	Detail
	Currency #2	Rp xxxxx,-	0.xxxxx	x	Detail
	Currency #3	Rp xxxxx,-	0.xxxxx	x	Detail
	Currency #4	Rp xxxxx,-	0.xxxxx	x	Detail
					◀ ▶

Gambar 4.5 Rancangan halaman daftar nilai tukar mata uang

- Rancangan tampilan rincian nilai tukar mata uang sesuai kode negara yang dipilih.



Gambar 4.6 Rancangan halaman rincian nilai tukar mata uang

3. Rancangan tampilan *edit data* nilai tukar mata uang sesuai kode negara yang dipilih.

Side Navbar /
Menu

Header

Edit Page

Currency

:

USD

Amount (IDR)

:

Rp xxxx,-

Margin

:

Profit

:

SAVE

BACK

Gambar 4.7 Rancangan halaman update nilai tukar mata uang

BAB V

IMPLEMENTASI

Sebagai bentuk tindak lanjut dari proses analisis dan perancangan modul nilai tukar mata uang di atas, maka dalam bab ini akan dijelaskan proses penerapan atau implementasi dari analisis dan rancangan tersebut. Penjelasan dalam bab ini meliputi tahap-tahap pembuatan kode program sehingga dapat menjadi sebuah *backend service* yang sesuai dan dapat digunakan pada produk atau layanan pengiriman uang dari dan ke luar negeri.

5.1 Implementasi Perangkat Lunak Pembangun

Daftar perangkat lunak yang digunakan untuk mengembangkan modul nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri meliputi:

1. Windows 10 Pro 64 Bit sebagai sistem operasi.
2. Microsoft Visual Studio Code sebagai *code editor*.
3. Bahasa pemrograman Go sebagai bahasa di *backend service*.
4. ReactJS sebagai *framework* antarmuka pengguna.
5. Google Chrome/Mozilla Firefox sebagai *web browser*.
6. Postman sebagai alat uji coba REST API.
7. DBeaver sebagai alat manajemen basis data.
8. Docker sebagai *virtual machine* untuk menjalankan NATS (*messaging service*).

5.2 Implementasi Perangkat Lunak Pembangun

Daftar perangkat keras yang digunakan untuk mengembangkan modul nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri meliputi:

1. Prosesor Intel® Core i5-7200U @2.5GHz

2. RAM 12.00 GB
3. *Harddisk Drive* 1 TB
4. Monitor LCD 14"
5. *Keyboard*
6. Tetikus

5.3 Implementasi Basis Data

Pada tahap implementasi basis data ini, penulis tidak akan menjelaskan tahapan pembuatan basis data dikarenakan seperti yang sudah dibahas pada bab sebelumnya perihal analisis dan perancangan di bagian rancangan basis data, penulis hanya sebatas menjadi pengguna biasa dengan hak akses terbatas pada tabel yang sudah dibuat oleh tim pengembang perusahaan. Oleh karena itu, rancangan basis data tersebut akan menjadi acuan dan digunakan dalam tahap implementasi pengembangan modul ini.

5.4 Implementasi REST API (*Backend Service*)

Pada subbab ini akan dijelaskan mengenai tahapan dalam pengimplentasian rancangan REST API yang sudah dibuat dengan mengacu pada *flowchart* sebagai pedoman algoritma dan alur datanya. REST API ini merupakan *backend service* pada modul nilai tukar mata uang yang memiliki peran sebagai perantara dari antarmuka pengguna dan basis data.

Untuk memudahkan manajemen kode dan memperjelas alur data dari REST API ini, maka *backend service* akan dibuat dengan struktur atau *layer* seperti berikut:

1. *Mapper*

Layer ini berperan untuk memindahkan variabel data yang ada pada *struct* di model *database* ke *struct* yang ada di model *data transfer object* (DTO) sehingga dapat digunakan sesuai alur yang bekerja.

2. *Repository*

Layer ini memiliki tugas untuk melakukan *query* ke basis data sesuai *usecase* yang ada. *Query* yang dilakukan bentuknya akan berbeda dengan *query* bawaan SQL karena menggunakan *library* ORM yaitu go-pg.

3. *Usecase*

Layer ini merupakan tempat di mana logika bisnis terjadi, seperti untuk mendapatkan daftar semua nilai tukar mata uang, melihat detail per kode negara, dan memperbaharui data nilai tukar mata uang per kode negara.

4. *Config*

Layer ini berperan penting sebagai inisiator atau penyambung koneksi ke basis data, koneksi ke *message broker* (NATS), dan juga inisiasi pengaturan variabel *service* seperti IP, port, username, password, dan nama basis data.

5. *Messaging*

Layer ini merupakan tempat *message broker* yang dalam hal ini yaitu NATS untuk melakukan proses manajemen komunikasinya yang dapat berupa *Publish-Subscribe*.

6. *Model*

Layer ini adalah tempat menyimpan beberapa *struct* yang akan digunakan dalam *service* supaya dapat diakses secara global.

Sedikit catatan mengenai pembagian API, dikarenakan ada salah satu data² yang lokasinya berada pada basis data yang berbeda dan mengacu pada rancangan REST API serta logika bisnis yang dibutuhkan, maka akan dibagi menjadi 3 poin utama, yaitu:

- a. API data rate_aj
 - 1) API daftar nilai tukar mata uang

² Data profit.

API daftar nilai tukar mata uang merupakan *service* yang digunakan untuk melihat semua data nilai tukar mata uang yang tersimpan dalam basis data pada tabel *rate_aj* dalam bentuk *list*. Pengimplementasian API ini terdiri dari beberapa *layer*. Dapat dilihat kode program daftar nilai tukar mata uang untuk *layer Mapper* pada Gambar 5.1.

```
// ToListRateAJViewPayload ...
func (m *MapperRate) ToListRateAJViewPayload(listRateAJ []model.RateAJ) []model.RateAJViewPayload {
    listPayload := []model.RateAJViewPayload{}
    for _, rowRateAJ := range listRateAJ {
        payload := model.RateAJViewPayload{
            DestinationCurrency: rowRateAJ.DestinationCurrency,
            RateIDRDest:         rowRateAJ.RateIDRDest,
            Margin:              rowRateAJ.Margin,
        }
        listPayload = append(listPayload, payload)
    }
    return listPayload
}
```

Gambar 5.1 Potongan kode mapper daftar nilai tukar mata uang

Fungsi tersebut memiliki nama *ToListRateAJViewPayload* yang memiliki satu parameter yaitu *listRateAJ* yang berupa sebuah array dari *model.RateAJ*, dan fungsi ini mengembalikan data olahan yang berupa array juga dengan nama *listPayload* yang berbentuk array dari *model.RateAJViewPayload*. Untuk kode program daftar nilai tukar mata uang pada *layer* selanjutnya yaitu *layer Repository* dapat dilihat pada Gambar 5.2.

```
//GetAllRateAJDB is used to get List ALL Rate AJ
func (r *RateRepo) GetAllRateAJDB() []model.RateAJ {
    RateAJList := []model.RateAJ{}
    if errGetAll := r.DatabaseBKUTrxWeb.Model(&RateAJList).Select(); errGetAll != nil {
        log.Error(errGetAll.Error())
    }
    return RateAJList
}
```

Gambar 5.2 Potongan kode repository daftar nilai tukar mata uang

Fungsi tersebut memiliki nama *GetAllRateAJDB* dan tidak memiliki parameter apapun namun memiliki *return value* yaitu *RateAJList* yang berupa array dari *model.RateAJ*. Untuk kode program daftar nilai tukar mata uang pada *layer Usecase* dapat dilihat pada Gambar 5.3.

```
// GetAllRateAJ ...
func (c *RateUsecase) GetAllRateAJ(refNumber string, message model.Message) interface{} {
    var (
        yyyymmddhhmmss = "2006-01-02 15:04:05"
    )
    currentTime := time.Now()
    username := message.Header.DataUser.Username
    roleName := message.Header.DataUser.RoleName
    listRateAJ := c.rateRepo.GetAllRateAJDB()
    listPayload := c.rateMapper.ToListRateAJViewPayload(listRateAJ)
    response := util.ResponseSuccess_ListData(listPayload, len(listPayload))
    userAudit := model.UserAudit{
        Username: username,
        RoleName: roleName,
    }
    c.interCon.LogActivity(refNumber, currentTime.Format(yyyymmddhhmmss), "GET LIST RATE AJ", message.Header.URL, "SUCCESS", nil, userAudit)
    return response
}
```

Gambar 5.3 Potongan kode usecase daftar nilai tukar mata uang

Fungsi tersebut memiliki nama GetAllRateAJ dengan dua parameter yaitu refNumber yang berbentuk string dan message yang berbentuk model.Message dan mempunyai *return value* bernama response yang berbentuk *interface*. Sedangkan untuk kode program daftar nilai tukar mata uang pada *layer Messaging* (NATS) dapat dilihat pada Gambar 5.4.

```
//GetAllRateAJ for get list of Rate AJ
func (n *NatsJobs) GetAllRateAJ(subject, reply string, msg map[string]interface{}) {
    var (
        refNumber = util.GetReffNumber(subject)
        message    model.Message
    )
    resultAsJSON, _ := json.Marshal(msg)
    json.Unmarshal(resultAsJSON, &message)
    log.Message(
        refNumber,
        "IN",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(resultAsJSON),
    )
    response := n.rateUsecase.GetAllRateAJ(refNumber, message)
    n.natsEngine.Publish(reply, response)
    responseJSON, _ := json.Marshal(response)
    log.Message(
        refNumber,
        "OUT",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(responseJSON),
    )
}
```

Gambar 5.4 Potongan kode nats_jobs daftar nilai tukar mata uang

Fungsi tersebut memiliki nama `GetAllRateAJ` dengan tiga parameter yaitu `subject`, `reply`, dan `msg`. Fungsi tersebut tidak memiliki *return value* namun bekerja untuk menghubungkan antara service dengan *message broker* dalam hal ini NATS.

2) API rincian nilai tukar mata uang

API rincian nilai tukar mata uang merupakan *service* yang digunakan untuk melihat rincian data nilai tukar mata uang yang tersimpan dalam basis data pada tabel `rate_aj` per kode negara. Pengimplementasian API ini terdiri dari beberapa *layer*. Dapat dilihat kode program daftar nilai tukar mata uang untuk *layer Mapper* pada Gambar 5.5.

```
// ToRateAJDetailViewPayload ...
func (m *MapperRate) ToRateAJDetailViewPayload(rateaj model.RateAJ) model.RateAJDetailViewPayload {
    RateAJDetailViewPayload := model.RateAJDetailViewPayload{
        DestinationCurrency: rateaj.DestinationCurrency,
        RateIDRDest:         rateaj.RateIDRDest,
        Margin:              rateaj.Margin,
    }
    return RateAJDetailViewPayload
}
```

Gambar 5.5 Potongan kode mapper rincian nilai tukar mata uang

Fungsi tersebut memiliki nama `ToRateAJDetailViewPayload` dengan satu parameter yaitu `rateaj` yang berbentuk `model.RateAJ` dan memiliki *return value* yaitu `RateAJDetailViewPayload` yang berbentuk `model.RateAJDetailViewPayload`. Untuk kode program rincian nilai tukar mata uang pada *layer* selanjutnya yaitu *layer Repository* dapat dilihat pada Gambar 5.6.

```
//GetSingleRateAJDB is used to get RateAJ by ID
func (r *RateRepo) GetSingleRateAJDB(rateajID string) model.RateAJ {
    var RateAJData model.RateAJ
    if errGet := r.DatabaseBKUTrxWeb.Model(&RateAJData).Where("destination_currency = ?", rateajID).Select(); errGet != nil {
        log.Error(errGet.Error())
    }
    return RateAJData
}
```

Gambar 5.6 Potongan kode repository rincian nilai tukar mata uang

Fungsi tersebut memiliki nama `GetSingleRateAJDB` dengan satu parameter yaitu `rateajID` berbentuk *string* dan memiliki *return value* yaitu `RateAJData` yang berbentuk `model.RateAJ`. Untuk kode program rincian nilai tukar mata uang pada *layer Usecase* dapat dilihat pada Gambar 5.7.

```

// GetRateAJByID ...
func (c *RateUsecase) GetRateAJByID(refNumber string, message model.Message) interface{} {
    var (
        yyyymmddhhmmss = "2006-01-02 15:04:05"
    )
    currentTime := time.Now()
    username := message.Header.DataUser.Username
    roleName := message.Header.DataUser.RoleName
    RateAJData := c.rateRepo.GetSingleRateAJDB(message.PathVariable[0])
    if RateAJData.DestinationCurrency == "" {
        return util.ResponseError_NotFound()
    }
    payload := c.rateMapper.ToRateAJDetailViewPayload(RateAJData)
    userAudit := model.UserAudit{
        Username: username,
        RoleName: roleName,
    }
    c.interCon.LogActivity(
        refNumber,
        currentTime.Format(yyyymmddhhmmss),
        "GET DETAIL RATE AJ", message.Header.URL,
        "SUCCESS",
        "GET DETAIL RATE AJ : "+payload.DestinationCurrency,
        userAudit
    )
    return util.ResponseSuccess_Data(payload)
}

```

Gambar 5.7 Potongan kode usecase rincian nilai tukar mata uang

Fungsi tersebut memiliki nama `GetRateAJByID` dengan dua parameter yaitu `refNumber` yang berbentuk *string* dan `message` yang berbentuk `model.Message` serta memiliki *return value* dari hasil `util.ResponseSuccess_Data(payload)`. Sedangkan untuk kode program rincian nilai tukar mata uang pada *layer Messaging* (NATS) dapat dilihat pada Gambar 5.8.

```

//GetSingleRateAJ for get single of RateAJ
func (n *NatsJobs) GetSingleRateAJ(subject, reply string, msg map[string]interface{}) {
    var (
        refNumber = util.GetReffNumber(subject)
        message    model.Message
    )
    JSONByte, _ := json.Marshal(msg)
    json.Unmarshal(JSONByte, &message)
    log.Message(
        refNumber,
        "IN",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(JSONByte),
    )
    response := n.rateUsecase.GetRateAJByID(refNumber, message)
    n.natsEngine.Publish(reply, response)
    resJSON, _ := json.Marshal(response)
    log.Message(
        refNumber,
        "OUT",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(resJSON),
    )
}

```

Gambar 5.8 Potongan kode nats_jobs rincian nilai tukar mata uang

Fungsi tersebut memiliki nama `GetSingleRateAJ` dengan tiga parameter yaitu `subject`, `reply`, dan `msg`. Fungsi tersebut tidak memiliki *return value* namun bekerja untuk menghubungkan antara service dengan *message broker* dalam hal ini NATS.

3) API *update* nilai tukar mata uang

API *update* nilai tukar mata uang merupakan *service* yang digunakan untuk memperbaharui data nilai tukar mata uang yang tersimpan dalam basis data pada tabel `rate_aj` sesuai kode negara yang dipilih. Pengimplementasian API ini terdiri dari beberapa *layer*. Dapat dilihat kode program daftar nilai tukar mata uang untuk *layer Mapper* pada Gambar 5.9.

```
// ToRateAJ_Update ...
func (m *MapperRate) ToRateAJ_Update(reqUpdateRateAJ model.RateAJUpdatePayload) model.RateAJ_Update {
    RateAJData := model.RateAJ_Update{
        DestinationCurrency: reqUpdateRateAJ.DestinationCurrency,
        RateIDRDest:         reqUpdateRateAJ.RateIDRDest,
        Margin:              reqUpdateRateAJ.Margin,
    }
    return RateAJData
}
```

Gambar 5.9 Potongan kode mapper update nilai tukar mata uang

Fungsi tersebut memiliki nama `ToRateAJ_Update` dengan satu parameter yaitu `reqUpdateRatAJ` yang berbentuk `model.RateAJUpdatePayload` dan memiliki *return value* yaitu `RateAJData` yang berbentuk `model.RateAJ_Update`. Untuk kode program *update* nilai tukar mata uang pada *layer* selanjutnya yaitu *layer Repository* dapat dilihat pada Gambar 5.10.

```
//UpdateRateAJDB is used to update RateAJ by ID
func (r *RateRepo) UpdateRateAJDB(rateajID string, RateAJData model.RateAJ_Update) error {
    RateAJData.DestinationCurrency = rateajID
    res, errUpdate := r.DatabaseBKUTrxWeb.Model(&RateAJData).Where("destination_currency = ?", rateajID).Update()
    if errUpdate != nil {
        log.Error("error update rateaj repo : " + errUpdate.Error())
        return errUpdate
    } else if res.RowsAffected() < 1 {
        return errors.New("destination_currency not found")
    }
    return nil
}
```

Gambar 5.10 Potongan kode repository update nilai tukar mata uang

Fungsi tersebut memiliki nama `UpdateRateAJDB` dengan dua parameter yaitu `rateajID` yang berbentuk *string* dan `RateAJData` yang berbentuk `model.RateAJ_Update`. Untuk kode program *update* nilai tukar mata uang pada *layer Usecase* dapat dilihat pada Gambar 5.11.

```

// UpdateRateAJDB ...
func (c *RateUsecase) UpdateRateAJDB(refNumber string, message model.Message) interface{} {
    var (
        reqBodyUpdateRateAJ model.RateAJUpdatePayload
        yyyyymmddhhmmss      = "2006-01-02 15:04:05"
    )
    currentTime := time.Now()
    username := message.Header.DataUser.Username
    roleName := message.Header.DataUser.RoleName
    reqBody, _ := json.Marshal(message.Body)
    err := json.Unmarshal(reqBody, &reqBodyUpdateRateAJ)
    if err != nil {
        log.Error("error marshall on update RateAJ : " + err.Error())
        return util.ResponseError_GenericCustom(err.Error())
    }
    RateAJData := c.rateMapper.ToRateAJ_Update(reqBodyUpdateRateAJ)
    if err := c.rateRepo.UpdateRateAJDB(message.PathVariable[0], RateAJData); err != nil {
        log.Error(err.Error())
        return util.ResponseError_GenericCustom(err.Error())
    }
    responseUpdateRateAJ := util.ResponseSuccess_Generic()
    // idArr := []string{RateAJData.DestinationCurrency}
    // c.interCon.NotifChangeToServiceVATrx("UPDATE", idArr, referenceNumber)
    userAudit := model.UserAudit{
        Username: username,
        RoleName: roleName,
    }
    c.interCon.LogActivity(
        refNumber,
        currentTime.Format(yyyyymmddhhmmss),
        "UPDATE RATE AJ", message.Header.URL,
        "SUCCESS", "UPDATE RATE AJ : "+RateAJData.DestinationCurrency,
        userAudit
    )
    return responseUpdateRateAJ
}

```

Gambar 5.11 Potongan kode usecase update nilai tukar mata uang

Fungsi tersebut memiliki nama UpdateRateAJDB dengan dua parameter yaitu refNumber yang berbentuk *string* dan message yang berbentuk model.Message dan *return value* bernama responseUpdateRateAJ yang berbentuk interface. Sedangkan untuk kode program *update* nilai tukar mata uang pada *layer Messaging* (NATS) dapat dilihat pada Gambar 5.12.

```

//UpdateRateAJ to update single RateAJ
func (n *NatsJobs) UpdateRateAJ(subject, reply string, msg map[string]interface{}) {
    var (
        refNumber = util.GetReffNumber(subject)
        message    model.Message
    )
    asJSON, _ := json.Marshal(msg)
    json.Unmarshal(asJSON, &message)
    log.Message(
        refNumber,
        "IN",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(asJSON),
    )
    response := n.rateUsecase.UpdateRateAJDB(refNumber, message)
    n.natsEngine.Publish(reply, response)
    resJSON, _ := json.Marshal(response)
    log.Message(
        refNumber,
        "OUT",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(resJSON),
    )
}

```

Gambar 5.12 Potongan kode nats_jobs update nilai tukar mata uang

Fungsi tersebut memiliki nama UpdateRateAJ dengan tiga parameter yaitu subject, reply, dan msg. Fungsi tersebut tidak memiliki *return value* namun bekerja untuk menghubungkan antara service dengan *message broker* dalam hal ini NATS.

b. API data profit

Implementasi API data profit ini sejatinya tidak memiliki perbedaan dengan API data rate_aj dalam segi alur, namun perbedaan hanya terdapat pada bagian penamaan fungsi dan variabel di dalamnya. Berikut penjabaran implementasi API data profit.

1) API daftar profit

API daftar profit merupakan *service* yang digunakan untuk melihat semua data profit yang tersimpan dalam basis data pada tabel *profit* dalam bentuk *list*. Pengimplementasian API ini terdiri dari beberapa *layer*. Dapat dilihat kode program daftar profit untuk *layer Mapper* pada Gambar 5.13.

```
// ToListRateAJProfitViewPayload ...
func (m *MapperRate) ToListRateAJProfitViewPayload(listRateAJProfit []model.RateAJ_Profit) []model.RateAJProfitViewPayload {
    listProfitPayload := []model.RateAJProfitViewPayload{}
    for _, rowRateAJProfit := range listRateAJProfit {
        payload := model.RateAJProfitViewPayload {
            CurrencyDestination: rowRateAJProfit.CurrencyDestination,
            MinProfit:           rowRateAJProfit.MinProfit,
        }
        listProfitPayload = append(listProfitPayload, payload)
    }
    return listProfitPayload
}
```

Gambar 5.13 Potongan kode mapper daftar profit

Fungsi tersebut memiliki nama `ToListRateAJProfitViewPayload` dengan satu parameter yaitu `listRateAJProfit` yang berbentuk *array* `model.RateAJ_Profit` dengan *return value* `listProfitPayload` yang berbentuk *array* `model.RateAJProfitViewPayload`. Untuk kode program daftar profit pada *layer* selanjutnya yaitu *layer Repository* dapat dilihat pada Gambar 5.14.

```
// GetAllRateAJProfit is used to get list all profit
func (r *RateRepo) GetAllRateAJProfit() []model.RateAJ_Profit {
    RateAJProfit := []model.RateAJ_Profit{}
    if errGetProfit := r.DatabaseBKUMaster.Model(&RateAJProfit).Select(); errGetProfit != nil {
        log.Error(errGetProfit.Error())
    }
    return RateAJProfit
}
```

Gambar 5.14 Potongan kode repository daftar profit

Fungsi tersebut memiliki nama `GetAllRateAJProfit` tanpa parameter apapun dan dengan *return value* `RateAJProfit` yang berbentuk *array* `model.RateAJ_Profit`. Untuk kode program daftar profit pada *layer Usecase* dapat dilihat pada Gambar 5.15.

```
// GetAllRateAJProfit ...
func (c *RateUsecase) GetAllRateAJProfit(refNumber string, message model.Message) interface{} {
    var (
        yyyyymmddhhmmss = "2006-01-02 15:04:05"
    )
    currentTime := time.Now()
    username := message.Header.DataUser.Username
    roleName := message.Header.DataUser.RoleName
    listRateAJProfit := c.rateRepo.GetAllRateAJProfit()
    listProfitPayload := c.rateMapper.ToListRateAJProfitViewPayload(listRateAJProfit)
    response := util.ResponseSuccess_listData(listProfitPayload, len(listProfitPayload))
    userAudit := model.UserAudit{
        Username: username,
        RoleName: roleName,
    }
    c.interCon.LogActivity(refNumber, currentTime.Format(yyyyymmddhhmmss), "GET LIST PROFIT", message.Header.URL, "SUCCESS", nil, userAudit)
    return response
}
```

Gambar 5.15 Potongan kode usecase daftar profit

Fungsi tersebut memiliki nama GetAllRateAJProfit dengan dua parameter *refNumber* yang berbentuk *string* dan *message* yang berbentuk *model.Message* dan *return value* response yang berbentuk *interface*. Sedangkan untuk kode program daftar profit pada *layer Messaging* (NATS) dapat dilihat pada Gambar 5.16.

```
//GetAllRateAJProfit for get list of Rate AJ
func (n *NatsJobs) GetAllRateAJProfit(subject, reply string, msg map[string]interface{}) {
    var (
        refNumber = util.GetReffNumber(subject)
        message     model.Message
    )
    resultAsJSON, _ := json.Marshal(msg)
    json.Unmarshal(resultAsJSON, &message)
    log.Message(
        refNumber,
        "IN",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(resultAsJSON),
    )
    response := n.rateUsecase.GetAllRateAJProfit(refNumber, message)
    n.natsEngine.Publish(reply, response)
    responseJSON, _ := json.Marshal(response)
    log.Message(
        refNumber,
        "OUT",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(responseJSON),
    )
}
```

Gambar 5.16 Potongan kode nats_jobs daftar profit

Fungsi tersebut memiliki nama GetAllRateAJProfit dengan tiga parameter yaitu *subject*, *reply*, dan *msg*. Fungsi tersebut tidak memiliki *return value* namun bekerja untuk menghubungkan antara *service* dengan *message broker* dalam hal ini NATS.

2) API rincian profit

API rincian profit merupakan *service* yang digunakan untuk melihat data profit yang tersimpan dalam basis data pada tabel *profit* per kode negara yang dipilih. Pengimplementasian API ini terdiri dari beberapa *layer*. Dapat dilihat kode program rincian profit untuk *layer Mapper* pada Gambar 5.17.

```
// ToRateAJProfitDetailViewPayload ...
func (m *MapperRate) ToRateAJProfitDetailViewPayload(rateajprofit model.RateAJ_Profit) model.RateAJProfitDetailViewPayload {
    RateAJProfitDetailViewPayload := model.RateAJProfitDetailViewPayload{
        CurrencyDestination: rateajprofit.CurrencyDestination,
        MinProfit:            rateajprofit.MinProfit,
    }
    return RateAJProfitDetailViewPayload
}
```

Gambar 5.17 Potongan kode mapper rincian profit

Fungsi tersebut memiliki nama `ToRateAJProfitDetailViewPayload` dengan satu parameter yaitu `rateajprofit` yang berbentuk `model.RateAJ_Profit` dan memiliki *return value* `RateAJProfitDetailViewPayload` yang berbentuk `model.RateAJProfitDetailViewPayload`. Untuk kode program rincian profit pada *layer* selanjutnya yaitu *layer Repository* dapat dilihat pada Gambar 5.18.

```
//GetSingleRateAJProfitDB is used to get RateAJProfit by ID
func (r *RateRepo) GetSingleRateAJProfitDB(rateajprofitID string) model.RateAJ_Profit {
    var RateAJProfitData model.RateAJ_Profit
    if errGet := r.DatabaseBKUMaster.Model(&RateAJProfitData).Where("currency_destination = ?", rateajprofitID).Select(); errGet != nil {
        log.Error(errGet.Error())
    }
    return RateAJProfitData
}
```

Gambar 5.18 Potongan kode repository rincian profit

Fungsi tersebut memiliki nama `GetSingleRateAJProfitDB` dengan satu parameter yaitu `rateajprofitID` yang berbentuk *string* dan memiliki *return value* `RateAJProfitData` yang berbentuk `model.RateAJ_Profit`. Untuk kode program rincian profit pada *layer Usecase* dapat dilihat pada Gambar 5.19.

```
// GetRateAJProfitByID ...
func (c *RateUsecase) GetRateAJProfitByID(refNumber string, message model.Message) interface{} {
    var (
        yyyymmddhhmmss = "2006-01-02 15:04:05"
    )
    currentTime := time.Now()
    username := message.Header.DataUser.Username
    roleName := message.Header.DataUser.RoleName
    RateAJProfitData := c.rateRepo.GetSingleRateAJProfitDB(message.PathVariable[0])
    if RateAJProfitData.CurrencyDestination == "" {
        return util.ResponseError_NotFound()
    }
    payload := c.rateMapper.ToRateAJProfitDetailViewPayload(RateAJProfitData)
    userAudit := model.UserAudit{
        Username: username,
        RoleName: roleName,
    }
    c.interCon.LogActivity(
        refNumber,
        currentTime.Format(yyyymmddhhmmss),
        "GET DETAIL PROFIT", message.Header.URL,
        "SUCCESS", "GET DETAIL PROFIT : "+payload.CurrencyDestination,
        userAudit
    )
    return util.ResponseSuccess_Data(payload)
}
```

Gambar 5.19 Potongan kode usecase rincian profit

Fungsi tersebut memiliki nama `GetRateAJProfitByID` dengan dua parameter yaitu `refNumber` yang berbentuk *string* dan `message` yang berbentuk `model.Message` dan memiliki *return value* yaitu hasil dari `util.ResponseSuccess_Data(payload)` yang berbentuk interface. Sedangkan untuk kode program rincian profit pada *layer Messaging* (NATS) dapat dilihat pada Gambar 5.20.

```
//GetSingleRateAJProfit for get single of RateAJProfit
func (n *NatsJobs) GetSingleRateAJProfit(subject, reply string, msg map[string]interface{}) {
    var (
        refNumber = util.GetReffNumber(subject)
        message    model.Message
    )
    JSONByte, _ := json.Marshal(msg)
    json.Unmarshal(JSONByte, &message)
    log.Message(
        refNumber,
        "IN",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(JSONByte),
    )
    response := n.rateUsecase.GetRateAJProfitByID(refNumber, message)
    n.natsEngine.Publish(reply, response)
    resJSON, _ := json.Marshal(response)
    log.Message(
        refNumber,
        "OUT",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(resJSON),
    )
}
```

Gambar 5.20 Potongan kode nats_jobs rincian profit

Fungsi tersebut memiliki nama `GetSingleRateAJProfit` dengan tiga parameter yaitu `subject`, `reply`, dan `msg`. Fungsi tersebut tidak memiliki *return value* namun bekerja untuk menghubungkan antara service dengan *message broker* dalam hal ini NATS.

3) API *update* profit

API *update* profit merupakan *service* yang digunakan untuk memperbaharui data profit yang tersimpan dalam basis data pada tabel *profit* sesuai kode negara

yang dipilih. Pengimplementasian API ini terdiri dari beberapa *layer*. Dapat dilihat kode program *update* profit untuk *layer Mapper* pada Gambar 5.21.

```
// ToRateAJProfit_Update ...
func (m *MapperRate) ToRateAJProfit_Update(reqUpdateRateAJProfit model.RateAJProfitUpdatePayload) model.RateAJ_Profit_Update {
    RateAJProfitData := model.RateAJ_Profit_Update{
        CurrencyDestination: reqUpdateRateAJProfit.CurrencyDestination,
        MinProfit:            reqUpdateRateAJProfit.MinProfit,
    }
    return RateAJProfitData
}
```

Gambar 5.21 Potongan kode mapper update profit

Fungsi tersebut memiliki nama `ToRateAJProfit_Update` dengan satu parameter yaitu `reqUpdateRateAJProfit` yang berbentuk `model.RateAJProfitUpdatePayload` dan memiliki *return value* `RateAJProfitData` yang berbentuk `model.RateAJ_Profit_Update`. Untuk kode program *update* profit pada *layer* selanjutnya yaitu *layer Repository* dapat dilihat pada Gambar 5.22.

```
//UpdateRateAJProfitDB is used to update RateAJ by ID
func (r *RateRepo) UpdateRateAJProfitDB(rateajprofitID string, RateAJProfitData model.RateAJ_Profit_Update) error {
    RateAJProfitData.CurrencyDestination = rateajprofitID
    res, errUpdate := r.DatabaseBKUMaster.Model(6RateAJProfitData).Where("currency_destination = ?", rateajprofitID).Update()
    if errUpdate != nil {
        log.Error("error update rateajprofit repo : " + errUpdate.Error())
        return errUpdate
    } else if res.RowsAffected() < 1 {
        return errors.New("currency_destination not found")
    }
    return nil
}
```

Gambar 5.22 Potongan kode repository update profit

Fungsi tersebut memiliki nama `UpdateRateAJProfitDB` dengan dua parameter yaitu `rateajprofitID` yang berbentuk *string* dan `RateAJProfitData` yang berbentuk `model.RateAJ_Profit_Update`. Untuk kode program *update* profit pada *layer Usecase* dapat dilihat pada Gambar 5.23.

```

// UpdateRateAJProfitDB ...
func (c *RateUsecase) UpdateRateAJProfitDB(refNumber string, message model.Message) interface{} {
    var (
        reqBodyUpdateRateAJProfit model.RateAJProfitUpdatePayload
        yyyyymmddhhmmss           = "2006-01-02 15:04:05"
    )
    currentTime := time.Now()
    username := message.Header.DataUser.Username
    roleName := message.Header.DataUser.RoleName
    reqBody, _ := json.Marshal(message.Body)
    err := json.Unmarshal(reqBody, &reqBodyUpdateRateAJProfit)
    if err != nil {
        log.Error("error marshall on update RateAJ : " + err.Error())
        return util.ResponseError_GenericCustom(err.Error())
    }
    RateAJProfitData := c.rateMapper.ToRateAJProfit_Update(reqBodyUpdateRateAJProfit)
    if err := c.rateRepo.UpdateRateAJProfitDB(message.PathVariable[0], RateAJProfitData); err != nil {
        log.Error(err.Error())
        return util.ResponseError_GenericCustom(err.Error())
    }
    responseUpdateRateAJProfit := util.ResponseSuccess_Generic()
    // idArr := []string{RateAJData.DestinationCurrency}
    // c.interCon.NotifChangeToServiceVATrx("UPDATE", idArr, referenceNumber)
    userAudit := model.UserAudit{
        Username: username,
        RoleName: roleName,
    }
    minProfit := fmt.Sprintf("%f", RateAJProfitData.MinProfit)
    c.interCon.LogActivity(
        refNumber,
        currentTime.Format(yyyyymmddhhmmss),
        "UPDATE PROFIT", message.Header.URL,
        "SUCCESS", "UPDATE PROFIT : "+RateAJProfitData.CurrencyDestination+" : "+minProfit,
        userAudit
    )
    return responseUpdateRateAJProfit
}

```

Gambar 5.23 Potongan kode usecase update profit

Fungsi tersebut memiliki nama `UpdateRateAJProfitDB` dengan dua parameter yaitu `refNumber` yang berbentuk *string* dan `message` yang berbentuk `model.Message` dan *return value* `responseUpdateRateAJProfit` yang berbentuk `interface`. Sedangkan untuk kode program *update* profit pada *layer Messaging* (NATS) dapat dilihat pada Gambar 5.24.

```

//UpdateRateAJProfit to update single RateAJProfit
func (n *NatsJobs) UpdateRateAJProfit(subject, reply string, msg map[string]interface{}) {
    var (
        refNumber = util.GetReffNumber(subject)
        message     model.Message
    )
    asJSON, _ := json.Marshal(msg)
    json.Unmarshal(asJSON, &message)
    log.Message(
        refNumber,
        "IN",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(asJSON),
    )
    response := n.rateUsecase.UpdateRateAJProfitDB(refNumber, message)
    n.natsEngine.Publish(reply, response)
    resJSON, _ := json.Marshal(response)
    log.Message(
        refNumber,
        "OUT",
        "FAST-HTTP",
        "",
        "TOPIC",
        subject,
        "",
        "",
        string(resJSON),
    )
}

```

Gambar 5.24 Potongan kode nats_jobs update profit

Fungsi tersebut memiliki nama UpdateRateAJProfit dengan tiga parameter yaitu subject, reply, dan msg. Fungsi tersebut tidak memiliki *return value* namun bekerja untuk menghubungkan antara service dengan *message broker* dalam hal ini NATS.

5.5 Implementasi Antarmuka Pengguna

Pada subbab ini akan dijabarkan mengenai proses pengimplementasian rancangan dari antarmuka pengguna dalam bentuk gambar tampilan halaman dan potongan kode program halaman tersebut.

5.5.1 Halaman Daftar Nilai Tukar Mata Uang

Halaman daftar nilai tukar mata uang ini adalah halaman bawaan yang ditampilkan ketika pengguna menuju bagian *Rate* pada navigasi bar di samping kiri.

Untuk dapat mengakses halaman ini, pengguna diharuskan untuk masuk dahulu ke dalam sistem. Data yang ditampilkan pada halaman ini didapat melalui panggilan API dengan metode HTTP GET menggunakan *library* Axios yang memiliki keluaran berbentuk JSON yang nantinya akan digunakan oleh komponen *table* untuk ditampilkan ke pengguna. Hasil implementasi halaman daftar nilai tukar mata uang dapat dilihat pada Gambar 5.25.

The screenshot shows a web application interface for 'Bersama Kirim Uang'. The main content area displays a table titled 'Rate Artajasa'. The table has the following data:

Currency	Amount (IDR)	Margin Kurs	Profit	Actions
PHP	308.11	153	0.5	Detail
USD	15444.00	2001	2	Detail
KRW	12.84	50	2	Detail
MYR	3627.57	700	2.5	Detail
GBP	19794.31	2000	1	Detail

The interface also includes a sidebar on the left with various icons, a top navigation bar with a search bar and user profile, and a footer with the text '2020 © Bersama Kirim Uang'.

Gambar 5.25 Hasil implementasi halaman daftar nilai tukar mata uang

Tabel tersebut memiliki 5 kolom yaitu Currency, Amount (IDR), Margin Kurs, Profit, dan Actions. Potongan kode program pada komponen *table* dapat dilihat pada Gambar 5.26.


```

return (
  <div>
    <MaterialTable
      title="Rate Artajasa"
      isLoading={props.isLoading}
      columns={[
        {
          title: 'Currency',
          field: 'destination_currency',
          editable: 'never'
        },
        { title: 'Amount (IDR)', field: 'rate_idr_dest', editable: 'never' },
        { title: 'Margin Kurs', field: 'margin' },
        { title: 'Profit', field: 'min_profit' },
        { title: 'Actions', field: "editrateaj", editable: 'never',
          render: item => {
            return (
              <Link
                onClick={() => handleDetail(item.destination_currency)}
                to={` /rate/page-rate/detail-rate-artajasa/${item.destination_currency}`}
                className="text-dark"
              >Detail</Link>
            )
          }
        }
      ]}
      data={props.Data.map(item => {
        return {
          destination_currency: item.destination_currency,
          rate_idr_dest: item.rate_idr_dest,
          margin: item.margin,
          min_profit: item.min_profit,
          editrateaj: item
        }
      })}
      options={{
        headerStyle: {
          backgroundColor: "#576574",
          color: "#FFF",
          maxWidth: 170,
          textAlign: "left"
        },
        rowStyle: {
          textAlign: "center"
        },
        actionsColumnIndex: 5,
        paginationType: "stepped",
        showTextRowsSelected: false,
        pageSizeOptions: [5, rowsHalf, props.Data.length],
        draggable: false
      }}
    </div>
  );

```

Gambar 5.26 Potongan kode halaman daftar nilai tukar mata uang

Tampilan tabel daftar nilai tukar mata uang ini menggunakan *library* Material-Table untuk memudahkan *styling* dan manajemen data yang ada. Dapat kita lihat di atas bahwa material tabel memiliki beberapa atribut seperti *title* untuk memberi nama tabel, *isLoading* yang berguna untuk menampilkan efek *loading* saat

data belum selesai didapatkan, *columns* yang berguna untuk membuat kolom dan nama kolomnya, *data* yang berfungsi sebagai tempat data yang akan dimasukkan ke baris sesuai kolom, dan *options* yang berfungsi sebagai pengaturan *styling* pada tabel.

Sedangkan untuk potongan kode program pada komponen yang berbentuk fungsi untuk memanggil API dapat dilihat pada Gambar 5.27.

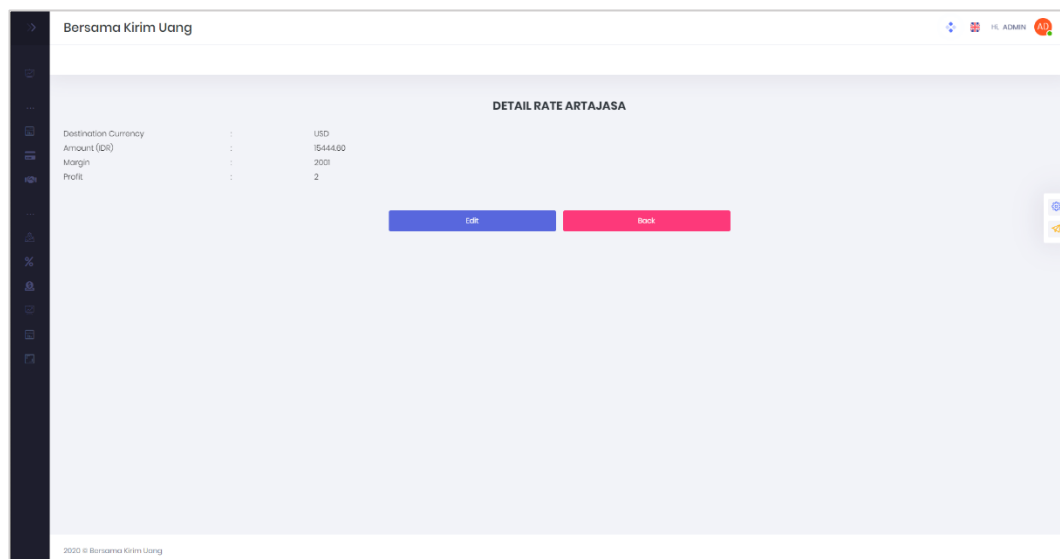
```
export const getDataRateAJPayload = () => {
  return {
    type: "GET_RATE_AJ",
    payload: new Promise((resolve, reject) => {
      const headers = {
        "x-header": `${Cookies.get("header")}`,
        "x-signature": `${Cookies.get("signature")}`,
        "x-refreshToken": `${Cookies.get("refreshToken")}`
      };
      Http.get(`/bku/v1/web/rateaj`, {
        headers: headers
      })
        .then(result => {
          resolve(result);
          if (result.headers["x-refreshToken"] !== undefined) {
            var expired = new Date(new Date().getTime() + 3600 * 1000);
            Cookies.set("header", `${result.headers["x-header"]}`, {
              expires: expired
            });
            Cookies.set("signature", `${result.headers["x-signature"]}`, {
              expires: expired
            });
            Cookies.set("refreshToken", `${result.headers["x-refreshToken"]}`, {
              expires: expired
            });
          }
        })
        .catch(error => {
          reject(error);
        });
    });
  };
};
```

Gambar 5.27 Potongan kode fungsi halaman daftar nilai tukar mata uang

Seperti yang dapat kita lihat pada Gambar 5.27, fungsi `getDataRateAJPayload` akan melakukan *request* data dengan metode HTTP GET ke *endpoint* API yang sesuai. Setelah data sudah didapatkan, maka akan langsung dikembalikan ke komponen tabel untuk divisualisasikan ke pengguna.

5.5.2 Halaman Rincian Nilai Tukar Mata Uang

Halaman rincian nilai tukar mata uang ini adalah halaman yang akan ditampilkan ketika pengguna mengeklik tombol *detail* pada salah satu baris mata uang. Halaman ini digunakan untuk memberi rincian informasi secara terfokus pada mata uang yang dipilih. Data yang ditampilkan pada halaman ini didapat melalui panggilan API dengan metode HTTP GET menggunakan *library* Axios yang memiliki keluaran berbentuk JSON yang nantinya akan digunakan oleh komponen *form* untuk ditampilkan ke pengguna. Hasil implementasi halaman rincian nilai tukar mata uang dapat dilihat pada Gambar 5.28.



Gambar 5.28 Hasil implementasi halaman rincian nilai tukar mata uang

Pada halaman rincian tersebut dapat kita lihat ada beberapa baris yaitu Destination Currency, Amount (IDR), Margin, dan Profit. Potongan kode program pada komponen *form* dapat dilihat pada Gambar 5.29.

```

return (
  <div>
    <Row className="show-grid">
      <Col xs={4} md={2}>
        <div className="list-form-name">Destination Currency</div>
      </Col>
      <Col xs={1} md={1} style={{ textAlign: "left" }}>
        <div className="list-form-name">:</div>
      </Col>
      <Col xs={7} md={9}>
        <div className="list-form-name">{props.DetailRateAJ.destination_currency}</div>
      </Col>
    </Row>
    <Row className="show-grid">
      <Col xs={4} md={2}>
        <div className="list-form-name">Amount (IDR)</div>
      </Col>
      <Col xs={1} md={1} style={{ textAlign: "left" }}>
        <div className="list-form-name">:</div>
      </Col>
      <Col xs={7} md={9}>
        <div className="list-form-name">{props.DetailRateAJ.rate_idr_dest}</div>
      </Col>
    </Row>
    <Row className="show-grid">
      <Col xs={4} md={2}>
        <div className="list-form-name">Margin</div>
      </Col>
      <Col xs={1} md={1} style={{ textAlign: "left" }}>
        <div className="list-form-name">:</div>
      </Col>
      <Col xs={7} md={9}>
        <div className="list-form-name">{props.DetailRateAJ.margin}</div>
      </Col>
    </Row>
    <Row className="show-grid">
      <Col xs={4} md={2}>
        <div className="list-form-name">Profit</div>
      </Col>
      <Col xs={1} md={1} style={{ textAlign: "left" }}>
        <div className="list-form-name">:</div>
      </Col>
      <Col xs={7} md={9}>
        <div className="list-form-name">{props.DetailRateAJ.min_profit}</div>
      </Col>
    </Row>
  </div>
);

```

Gambar 5.29 Potongan kode halaman rincian nilai tukar mata uang

Tabel rincian nilai tukar mata uang ini akan memiliki beberapa baris seperti yang dapat dilihat pada gambar potongan kode di atas, baris tersebut meliputi *Destination Currency* sebagai kode negara, *Amount (IDR)* sebagai nilai tukar mata uang, *Margin* sebagai margin, dan *Profit* sebagai profit.

Sedangkan untuk potongan kode program pada komponen yang berbentuk fungsi untuk memanggil API dapat dilihat pada Gambar 5.30.

```

export const getDetailRateAJPayload = (params) => {
  return {
    type: "GET_DETAIL_RATE_AJ",
    payload: new Promise((resolve, reject) => {
      const { id = "" } = params;
      const headers = {
        "x-header": `${Cookies.get("header")}`,
        "x-signature": `${Cookies.get("signature")}`,
        "x-refreshToken": `${Cookies.get("refreshToken")}`,
      };
      Http.get(`/bku/v1/web/rateaj/${id}`, {
        headers: headers,
      })
        .then((result) => {
          resolve(result);
          if (result.headers["x-refreshToken"] !== undefined) {
            var expired = new Date(new Date().getTime() + 3600 * 1000);
            Cookies.set("header", `${result.headers["x-header"]}`, {
              expires: expired,
            });
            Cookies.set("signature", `${result.headers["x-signature"]}`, {
              expires: expired,
            });
            Cookies.set("refreshToken", `${result.headers["x-refreshToken"]}`, {
              expires: expired,
            });
          }
        })
        .catch((error) => {
          reject(error);
        });
    }),
  };
};

```

Gambar 5.30 Potongan kode fungsi halaman rincian nilai tukar mata uang

Seperti yang dapat kita lihat pada Gambar 5.30, fungsi `getDetailRateAJPayload` akan melakukan *request* data dengan metode HTTP GET ke *endpoint* API yang sesuai dengan ditambahkan parameter *id* sebagai kode negara yang dipilih. Setelah data sudah didapatkan, maka akan langsung dikembalikan ke komponen tabel untuk divisualisasikan ke pengguna.

5.5.3 Halaman *Update* Nilai Tukar Mata Uang

Halaman *update* nilai tukar mata uang ini adalah halaman yang akan ditampilkan ketika pengguna mengeklik tombol *edit* pada halaman rincian mata uang. Halaman ini memiliki bentuk yang sama dengan halaman rincian nilai tukar mata uang, hanya saja yang membedakan adalah kemampuan untuk memberi masukan atau mengubah data yang ada yaitu data margin dan profit. Data yang

ditampilkan pada halaman ini didapat melalui panggilan API dengan metode HTTP GET menggunakan *library* Axios yang memiliki keluaran berbentuk JSON yang nantinya akan digunakan oleh komponen *form* untuk ditampilkan ke pengguna, serta data yang akan dikirim untuk memperbaharui data di basis data akan melalui panggilan API dengan metode HTTP POST menggunakan *library* AXIOS. Hasil implementasi halaman *update* nilai tukar mata uang dapat dilihat pada Gambar 5.31.

The screenshot shows a web application interface for 'Bersama Kirim Uang'. The main content area is titled 'EDIT RATE ARTAJASA (Margin)'. It contains a form with the following fields:

Field	Value
Destination Currency	USD
Amount (IDR)	15444.60
Margin	200
Profit	2

Below the form are two buttons: 'Save' (blue) and 'Back' (pink). The interface also features a sidebar with various icons and a top bar with the user name 'PR. ADMIN'.

Gambar 5.31 Hasil implementasi halaman update nilai tukar mata uang

Halaman *update* nilai tukar mata uang memiliki beberapa baris yang memiliki kolom masukan yaitu Destination Currency, Amount (IDR), Margin, dan Profit. Potongan kode program pada komponen *form* dapat dilihat pada Gambar 5.32.

```

return (
  <div>
    <Row className="show-grid">
      <Col xs={4} md={3}>
        <div className="list-form-name">Destination Currency</div>
      </Col>
      <Col xs={1} md={1} style={{textAlign:"left"}}>
        <div className="list-form-name"></div>
      </Col>
      <Col xs={12} md={8}>
        <input className="form-control form-value-create-ca" value={props.DestinationCurrency} readOnly/>
      </Col>
    </Row>
    <Row className="show-grid">
      <Col xs={4} md={3}>
        <div className="list-form-name">Amount (IDR)</div>
      </Col>
      <Col xs={1} md={1} style={{textAlign:"left"}}>
        <div className="list-form-name"></div>
      </Col>
      <Col xs={12} md={8}>
        <input className="form-control form-value-create-ca" value={props.RateIDRDest} readOnly/>
      </Col>
    </Row>
    <Row className="show-grid">
      <Col xs={4} md={3}>
        <div className="list-form-name">Margin</div>
      </Col>
      <Col xs={1} md={1} style={{textAlign:"left"}}>
        <div className="list-form-name"></div>
      </Col>
      <Col xs={12} md={8}>
        <input className="form-control form-value-create-ca" value={props.Margin} onChange={event => props.setMargin(event.target.value)}/>
      </Col>
    </Row>
    <Row className="show-grid">
      <Col xs={4} md={3}>
        <div className="list-form-name">Profit</div>
      </Col>
      <Col xs={1} md={1} style={{textAlign:"left"}}>
        <div className="list-form-name"></div>
      </Col>
      <Col xs={12} md={8}>
        <input className="form-control form-value-create-ca" value={props.MinProfit} onChange={event => props.setMinProfit(event.target.value)}/>
      </Col>
    </Row>
  </div>
);

```

Gambar 5.32 Potongan kode halaman update nilai tukar mata uang

Tabel *update* nilai tukar mata uang ini akan memiliki beberapa baris seperti yang dapat dilihat pada gambar potongan kode di atas, baris tersebut meliputi *Destination Currency* sebagai kode negara, *Amount (IDR)* sebagai nilai tukar mata uang, *Margin* sebagai margin, dan *Profit* sebagai profit. Baris yang dapat diubah datanya adalah baris *Margin* dan *Profit*.

Sedangkan untuk potongan kode program pada komponen yang berbentuk fungsi memanggil API untuk mengirim data ke basis data dapat dilihat pada Gambar 5.33.

```

// Post RateAJ
const postRateAJ = (headers) => {
  var idRateAJ = data.destination_currency
  Http.post(`/bku/v1/web/rateaj/${idRateAJ}`, toPostRateAJ, {
    headers: headers,
  })
  .then(result => {
    // localStorage.removeItem("x-cache-edit");
    if (result.headers["x-header"] !== undefined) {
      var expired = new Date(new Date().getTime() + 3600 * 1000);
      Cookies.set("header", `${result.headers["x-header"]}`, {
        expires: expired
      });
      Cookies.set("signature", `${result.headers["x-signature"]}`, {
        expires: expired
      });
      Cookies.set("refreshToken", `${result.headers["x-refreshToken"]}`, {
        expires: expired
      });
    }
  })
  .catch(error => {
    console.log(error);
    postCache();
  });
}

// Post RateAJ (Profit)
const postProfit = (headers) => {
  var idProfit = data.currency_destination
  Http.post(`/bku/v1/web/rateajprofit/${idProfit}`, toPostProfit, {
    headers: headers,
  })
  .then(result => {
    // localStorage.removeItem("x-cache-edit");
    if (result.headers["x-header"] !== undefined) {
      var expired = new Date(new Date().getTime() + 3600 * 1000);
      Cookies.set("header", `${result.headers["x-header"]}`, {
        expires: expired
      });
      Cookies.set("signature", `${result.headers["x-signature"]}`, {
        expires: expired
      });
      Cookies.set("refreshToken", `${result.headers["x-refreshToken"]}`, {
        expires: expired
      });
    }
  })
  .catch(error => {
    console.log(error);
    postCache();
  });
}

```

Gambar 5.33 Potongan kode fungsi halaman update nilai tukar mata uang

Seperti yang dapat kita lihat pada Gambar 5.33, fungsi `postRateAJ` dan `postProfit` akan melakukan *request* data dengan metode HTTP POST ke *endpoint* API yang sesuai dengan ditambahkan parameter *id* sebagai kode negara yang dipilih. Setelah data sudah berhasil ditambahkan ke basis data, maka halaman akan mengembalikan pengguna ke halaman daftar nilai tukar mata uang guna melihat perubahan yang terjadi.

BAB VI

PENGUJIAN DAN PEMBAHASAN

Setelah implementasi rancangan modul nilai tukar mata uang telah selesai, maka kita sampai pada bab pengujian dan pembahasan yang akan menjabarkan tahap pengujian modul untuk mengetahui apakah modul yang telah dibuat telah sesuai dengan rancangan yang ada dan apakah performa dari modul ini juga telah sesuai dengan standar produksi dari perusahaan. Pengujian ini difokuskan kepada fungsionalitas modul meliputi sisi REST API dan antarmuka pengguna.

6.1 Skenario Pengujian REST API

Skenario pengujian dibuat untuk menjelaskan tahap-tahap yang dilakukan untuk menguji REST API atau *backend service* dalam modul nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri ini agar diketahui keluaran apa yang dihasilkan dari beberapa kasus pengujian. Skenario pengujian yang dilakukan dapat dilihat pada Tabel 6.1.

Tabel 6.1 Skenario Pengujian REST API

Pengujian	Materi
Melakukan proses <i>get all data</i> pada tabel <i>rate_aj</i> .	Mengirim <i>request</i> API <i>get all data rate_aj</i> .
Melakukan proses <i>get single data</i> pada tabel <i>rate_aj</i> .	Mengirim <i>request</i> API <i>get single data rate_aj</i> dengan parameter kode negara (<i>destination_currency</i>).
Melakukan proses <i>update single data</i> pada tabel <i>rate_aj</i> .	Mengirim <i>request</i> API <i>update single data rate_aj</i> dengan parameter kode negara (<i>destination_currency</i>).
Melakukan proses <i>get all data</i> pada tabel <i>profit</i> .	Mengirim <i>request</i> API <i>get all data profit</i> .

Melakukan proses <i>get single data</i> pada tabel <i>profit</i>	Mengirim <i>request API get single data profit</i> dengan parameter kode negara (<i>currency_destination</i>).
Melakukan proses <i>update single data</i> pada tabel <i>profit</i>	Mengirim <i>request API update single data profit</i> dengan parameter kode negara (<i>currency_destination</i>).

6.2 Skenario Pengujian Antarmuka

Skenario pengujian antarmuka dibuat dengan tujuan untuk menjelaskan tahap-tahap pengujian terhadap implementasi antarmuka yang sudah ada. Dalam skenario ini terdapat satu kolom tambahan yaitu hasil yang diharapkan yang berfungsi sebagai acuan apakah implementasi antarmuka ini sudah sesuai dengan rancangan dan memenuhi syarat fungsionalitas modul.

Tabel 6.2 Skenario Pengujian Antarmuka

Pengujian	Materi	Hal yang diharapkan
Halaman daftar nilai tukar mata uang.	Mengakses URL halaman daftar modul nilai tukar mata uang.	Menampilkan halaman daftar nilai tukar mata uang beserta data lengkapnya.
Halaman rincian nilai tukar mata uang.	Mengakses URL halaman rincian nilai tukar mata uang dengan parameter salah satu kode negara.	Menampilkan halaman rincian daftar nilai tukar mata uang beserta data lengkapnya sesuai kode negara.
Halaman <i>edit</i> nilai tukar mata uang.	Mengakses URL halaman <i>edit</i> nilai tukar mata uang dengan parameter salah satu kode negara.	Menampilkan halaman <i>edit</i> nilai tukar mata uang beserta data lengkapnya sesuai kode negara.

6.3 Hasil Pengujian Modul

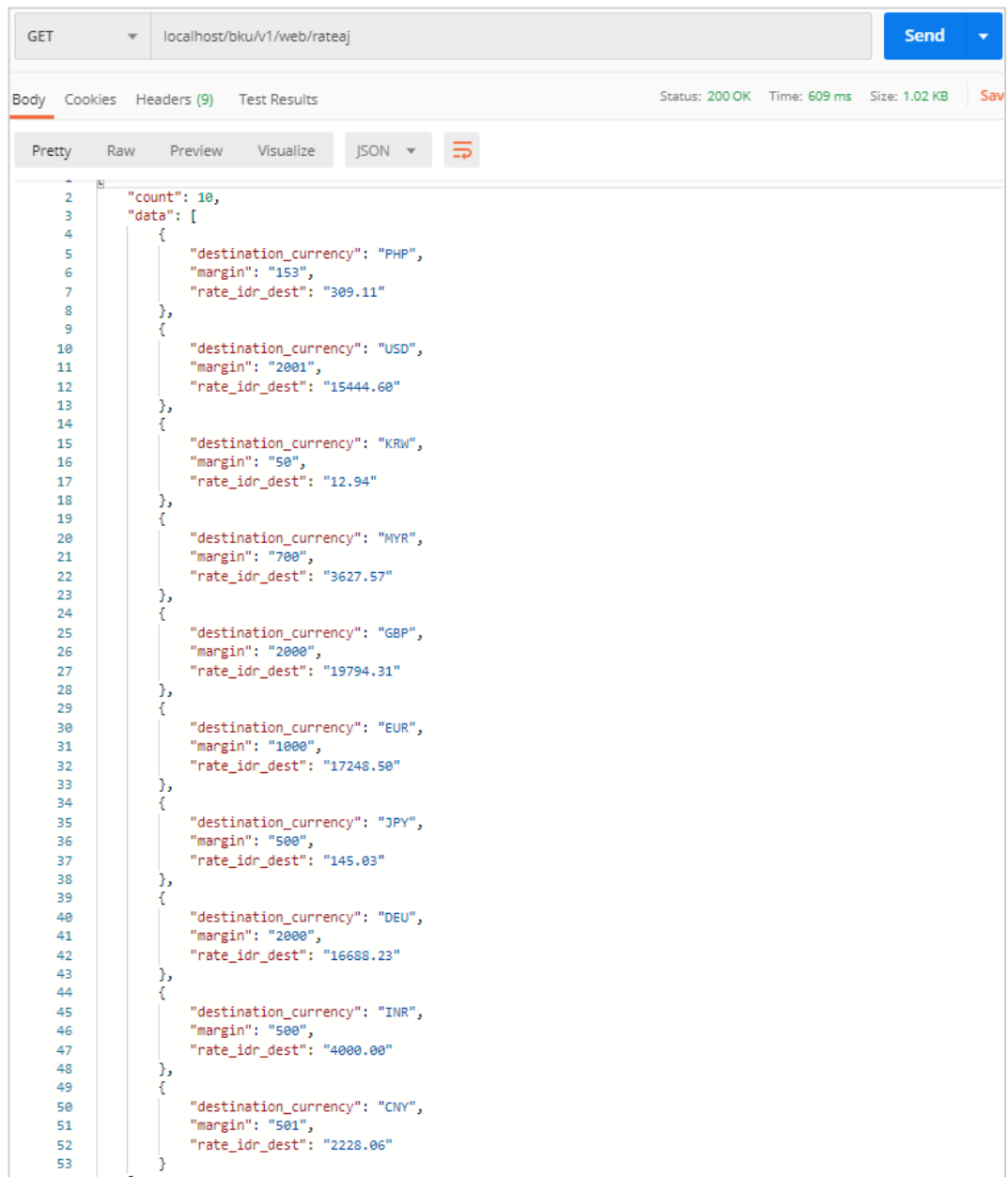
6.3.1 Hasil Pengujian REST API

Pengujian REST API dilakukan dengan menggunakan aplikasi REST Client API yaitu Postman. Pada aplikasi ini nanti akan dituliskan alamat API dan pemilihan metode HTTPnya seperti GET, POST, PUT, dan lain sebagainya. Ketika kita mengirimkan *request*, Postman akan otomatis menampilkan informasi hasil *response* API mulai dari data *response* hingga status *response*. Terdapat beberapa jenis status *response* dari REST API ini antara lain yaitu HTTP Status 200 yang berarti data berhasil didapatkan dan tidak ada masalah dalam proses *query*, HTTP 401 Unauthorized yang berarti tidak dapat dilakukan *query data* karena belum melakukan proses verifikasi masuk sistem.

a. Pengujian REST API *rate_aj*

1) Melakukan proses *get all data rate_aj*

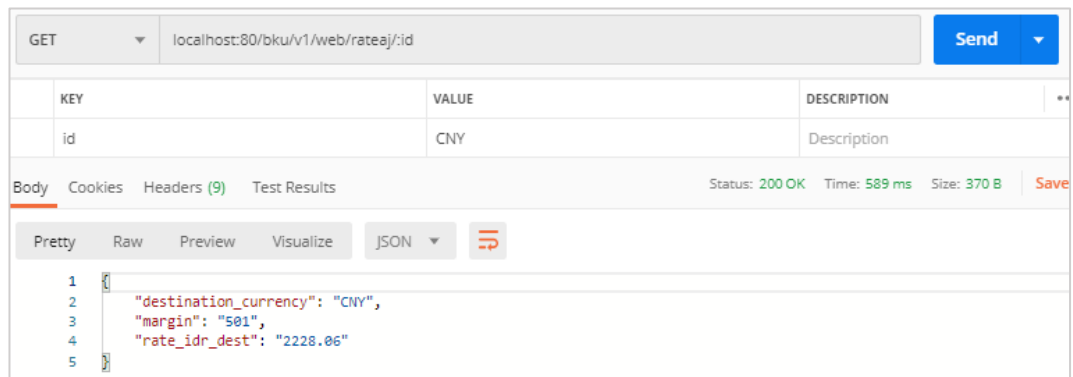
Pengujian proses *get all data rate_aj* dilakukan dengan menjalankan *query* SELECT ALL pada tabel *rate_aj*. Tujuan dari pengambilan data tersebut adalah untuk dapat menampilkan informasi daftar nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri. *Request* pada proses *get all data rate_aj* dapat dilihat pada Gambar 6.1.



Gambar 6.1 Response daftar nilai tukar mata uang

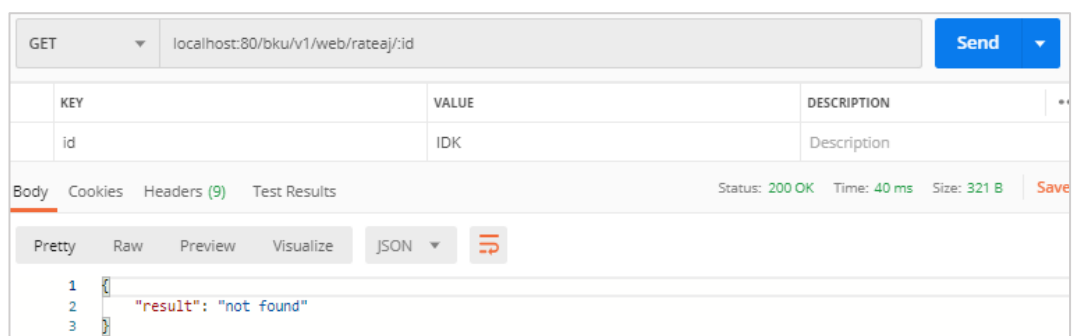
2) Melakukan proses *get single data rate_aj*

Pengujian proses *get single data rate_aj* dilakukan dengan menjalankan *query* SELECT - WHERE pada tabel *rate_aj*. Tujuan dari pengambilan data tersebut adalah untuk dapat menampilkan rincian informasi nilai tukar mata uang berdasarkan kode negara yang menjadi parameter *request*. *Request* pada proses *get single data rate_aj* dapat dilihat pada Gambar 6.2.



Gambar 6.2 Response success rincian nilai tukar mata uang

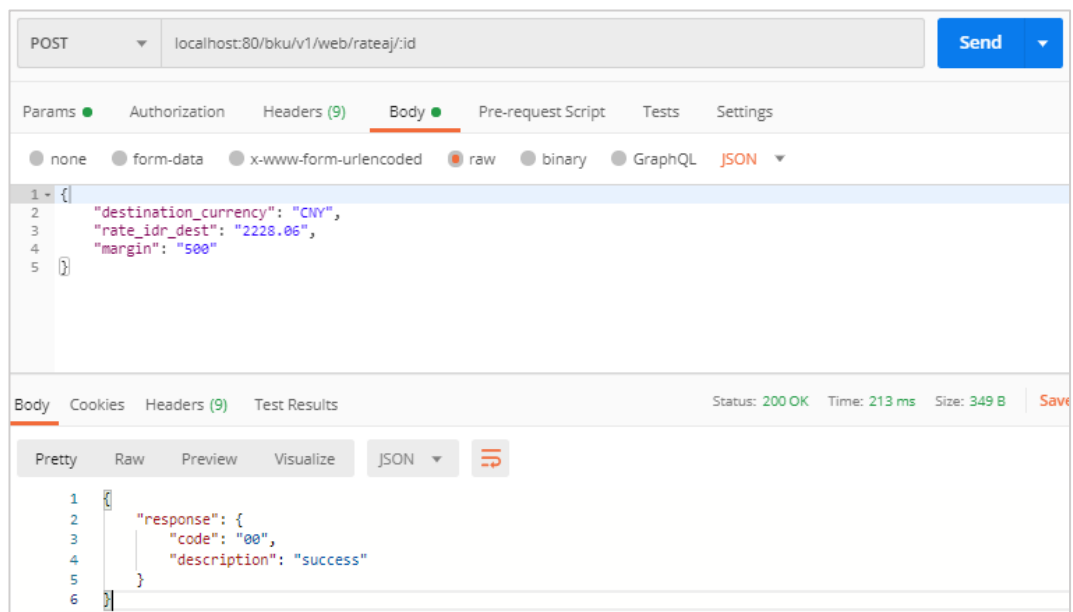
Sedangkan ketika parameter id kode negara yang dimasukkan salah ataupun tidak ditemukan saat *query* pada basis data, maka *response* yang didapat akan menjadi seperti yang ada pada Gambar 6.3.



Gambar 6.3 Response not found rincian nilai tukar mata uang

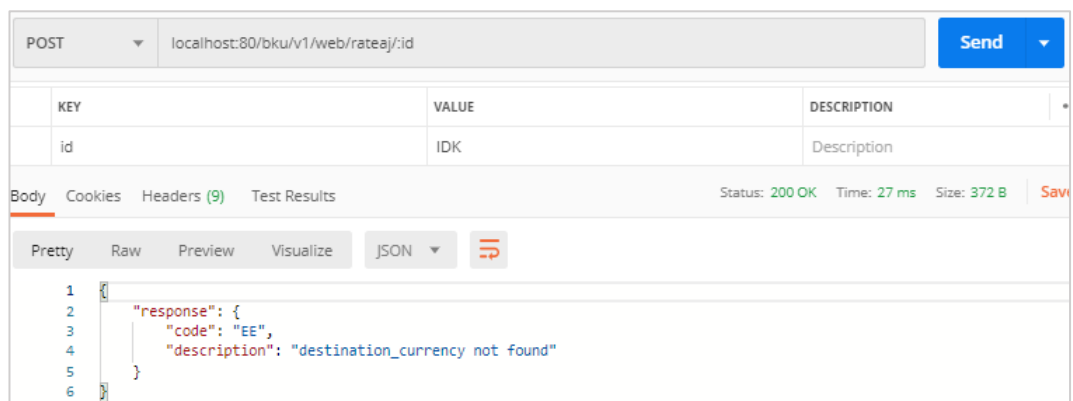
3) Melakukan proses *update single data rate_aj*

Pengujian proses *update single data rate_aj* dilakukan dengan menjalankan *query* UPDATE pada tabel *rate_aj*. Tujuan dari proses tersebut adalah untuk memperbarui data margin pada tabel nilai tukar mata uang berdasarkan kode negara yang menjadi parameter *request*. *Request* pada proses *update single data rate_aj* dapat dilihat pada Gambar 6.4.



Gambar 6.4 Response success update nilai tukar mata uang

Sedangkan ketika parameter id kode negara yang dimasukkan salah ataupun tidak ditemukan saat *query* pada basis data, maka *response* yang didapat akan menjadi seperti yang ada pada Gambar 6.5.

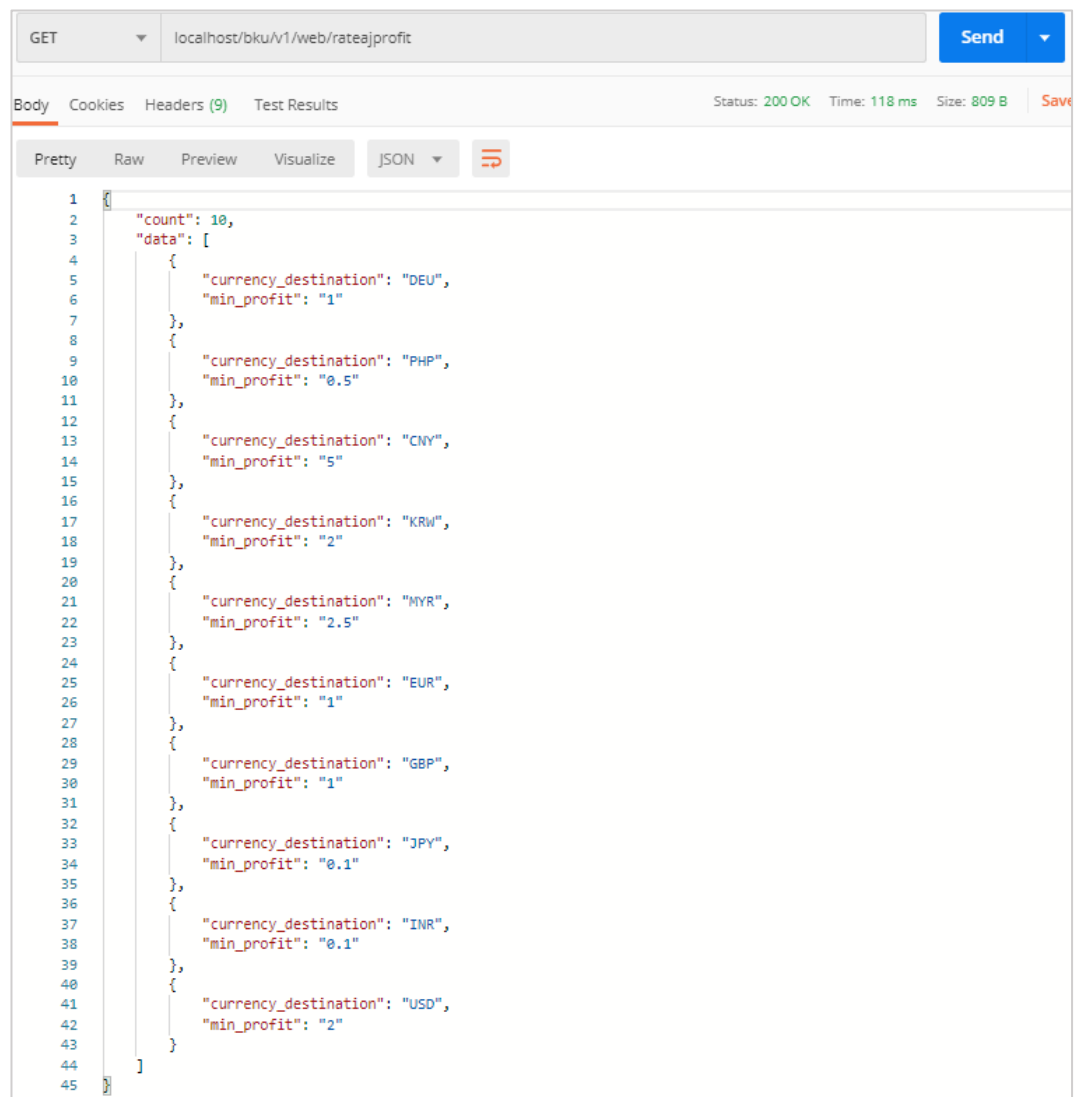


Gambar 6.5 Response not found update nilai tukar mata uang

b. Pengujian REST API *profit*

1) Melakukan proses *get all data profit*

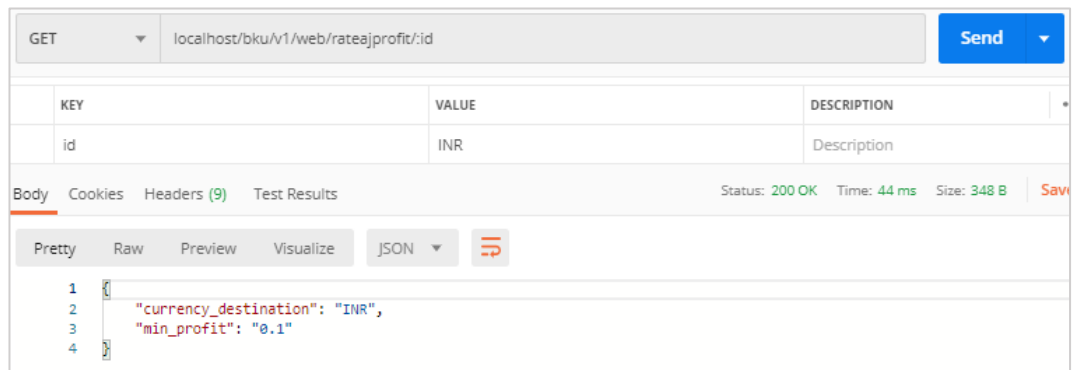
Pengujian proses *get all data profit* dilakukan dengan menjalankan *query* SELECT ALL pada tabel *profit*. Tujuan dari pengambilan data tersebut adalah untuk dapat menampilkan informasi daftar profit pada produk atau layanan pengiriman uang dari dan ke luar negeri. *Request* pada proses *get all data profit* dapat dilihat pada Gambar 6.6.



Gambar 6.6 Response daftar profit

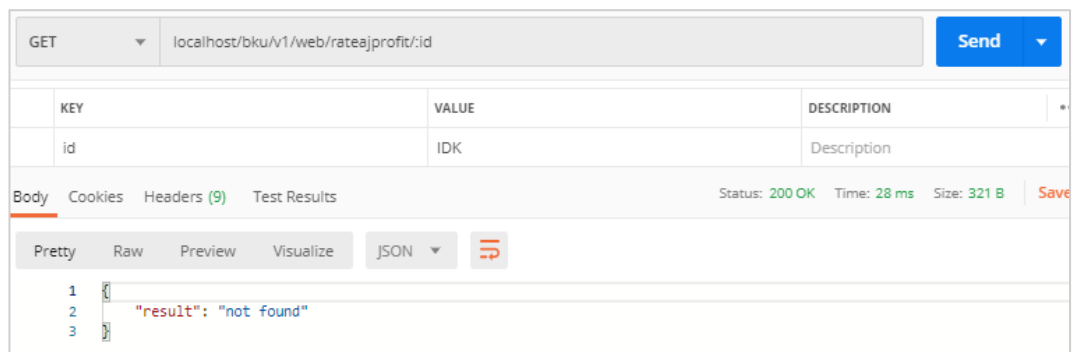
2) Melakukan proses *get single data profit*

Pengujian proses *get single data profit* dilakukan dengan menjalankan *query* SELECT - WHERE pada tabel *profit*. Tujuan dari pengambilan data tersebut adalah untuk dapat menampilkan rincian informasi profit berdasarkan kode negara yang menjadi parameter *request*. *Request* pada proses *get single data profit* dapat dilihat pada Gambar 6.7.



Gambar 6.7 Response success rincian profit

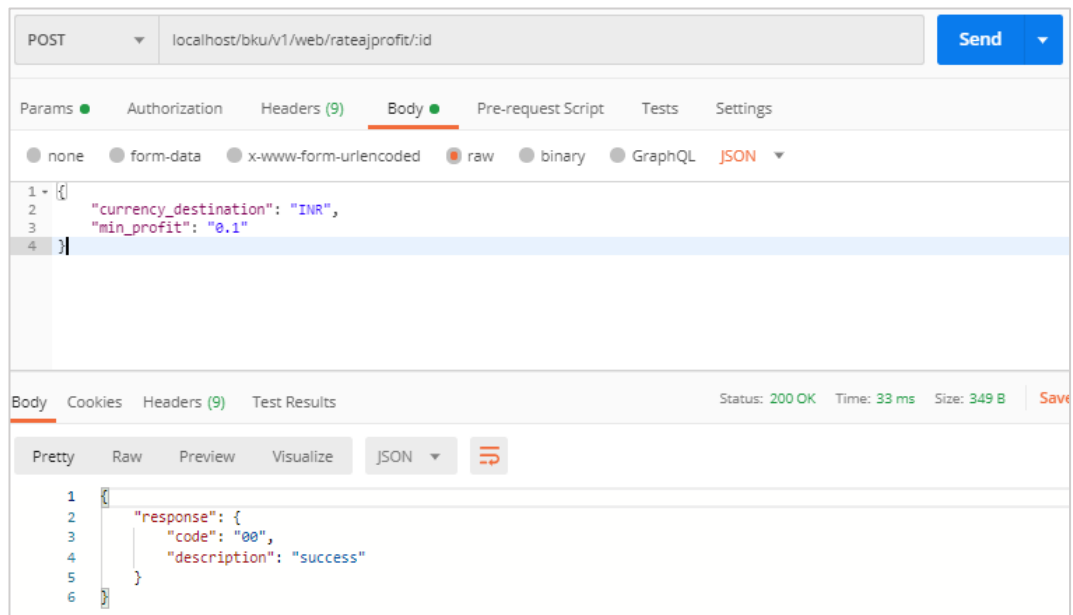
Sedangkan ketika parameter id kode negara yang dimasukkan salah ataupun tidak ditemukan saat *query* pada basis data, maka *response* yang didapat akan menjadi seperti yang ada pada Gambar 6.8.



Gambar 6.8 Response not found rincian profit

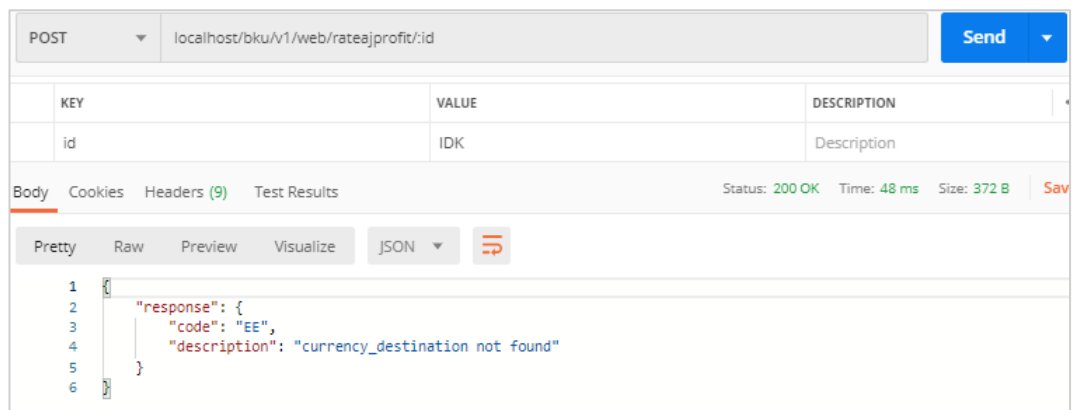
3) Melakukan proses *update single data profit*

Pengujian proses *update single data profit* dilakukan dengan menjalankan *query* UPDATE pada tabel *profit*. Tujuan dari proses tersebut adalah untuk memperbarui data *min_profit* pada tabel profit berdasarkan kode negara yang menjadi parameter *request*. *Request* pada proses *update single data profit* dapat dilihat pada Gambar 6.9.



Gambar 6.9 Response success update profit

Sedangkan ketika parameter id kode negara yang dimasukkan salah ataupun tidak ditemukan saat *query* pada basis data, maka *response* yang didapat akan menjadi seperti yang ada pada Gambar 6.10.



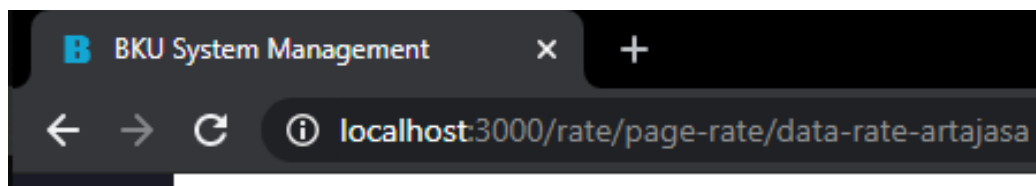
Gambar 6.10 Response not found update profit

6.3.2 Hasil Pengujian Antarmuka

Pengujian antarmuka ditujukan untuk melihat apakah hasil implementasi antarmuka sudah sesuai dengan rancangan yang telah dibuat. Penjelasan tiap halaman yang diuji yaitu sebagai berikut:

- 1) Pengujian Halaman Daftar Nilai Tukar Mata Uang

Halaman daftar nilai tukar mata uang ini berupa tabel yang berisi informasi tiap mata uang dari negara-negara yang didukung oleh produk atau layanan pengiriman uang dari dan ke luar negeri ini. Halaman ini hanya bisa diakses ketika pengguna sudah berhasil melakukan proses verifikasi masuk sistem. Alamat URL dari halaman ini yaitu “hostname/rate/page-rate/data-rate-artajasa”, karena pada pengujian ini menggunakan *local environment* maka URLnya berbentuk seperti yang ada pada Gambar 6.11.



Gambar 6.11 URL daftar nilai tukar mata uang

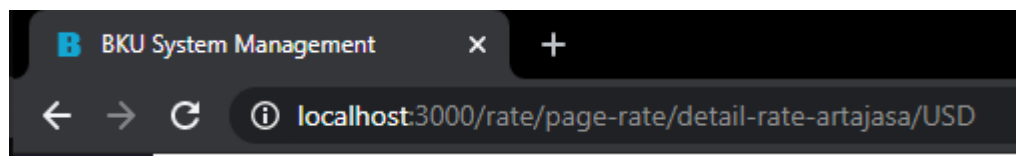
Setelah pengguna menekan tombol navigasi yang mengarah ke halaman daftar nilai tukar mata uang tersebut, maka sistem akan otomatis menampilkan halaman daftar nilai tukar mata uang tanpa harus memuat kembali halaman yang ada. Hal ini dimungkinkan karena bagian antarmuka menggunakan *library* ReactJS yang memiliki konsep *component-based*, yaitu hanya akan memuat komponen yang sedang diminta dan tetap menjaga agar komponen yang tidak diminta tidak perlu dimuat ulang.

Currency	Amount (IDR)	Margin Rate	Profit	Actions
PHP	329.11	153	0.5	Detail
USD	15444.60	2001	2	Detail
KRW	12.94	96	2	Detail
MYR	3627.57	760	2.5	Detail
GBP	19794.31	2000	1	Detail

Gambar 6.12 Tampilan halaman daftar nilai tukar mata uang

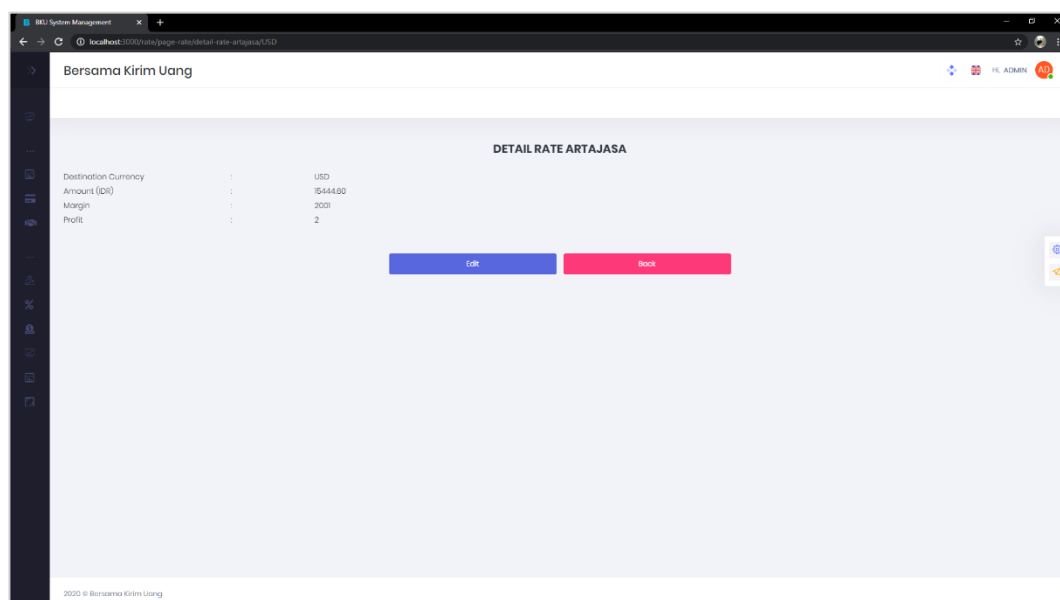
2) Pengujian Halaman Rincian Nilai Tukar Mata Uang

Halaman rincian nilai tukar mata uang merupakan halaman yang berupa *form* yang di dalamnya terdapat rincian data nilai tukar mata uang berdasarkan kode negara yang dipilih. Sama seperti halaman daftar nilai tukar mata uang, halaman ini juga hanya bisa diakses ketika pengguna sudah berhasil melakukan proses verifikasi masuk sistem. Alamat URL dari halaman rincian nilai tukar mata uang ini yaitu “hostname/rate/page-rate/detail-rate-artajasa/:id_negara”, karena pada pengujian ini menggunakan *local environment* maka URLnya berbentuk seperti yang ada pada Gambar 6.13.



Gambar 6.13 URL rincian nilai tukar mata uang

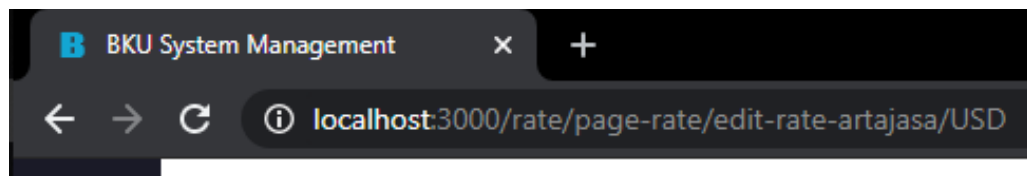
Setelah pengguna menekan tombol “Detail” pada halaman daftar nilai tukar mata uang sesuai negara yang diinginkan, maka sistem akan memuat halaman rincian ini. Karena masih dalam *library* ReactJS, maka sistem tidak akan memuat ulang seluruh halaman, melainkan hanya komponen yang diminta.



Gambar 6.14 Tampilan halaman rincian nilai tukar mata uang

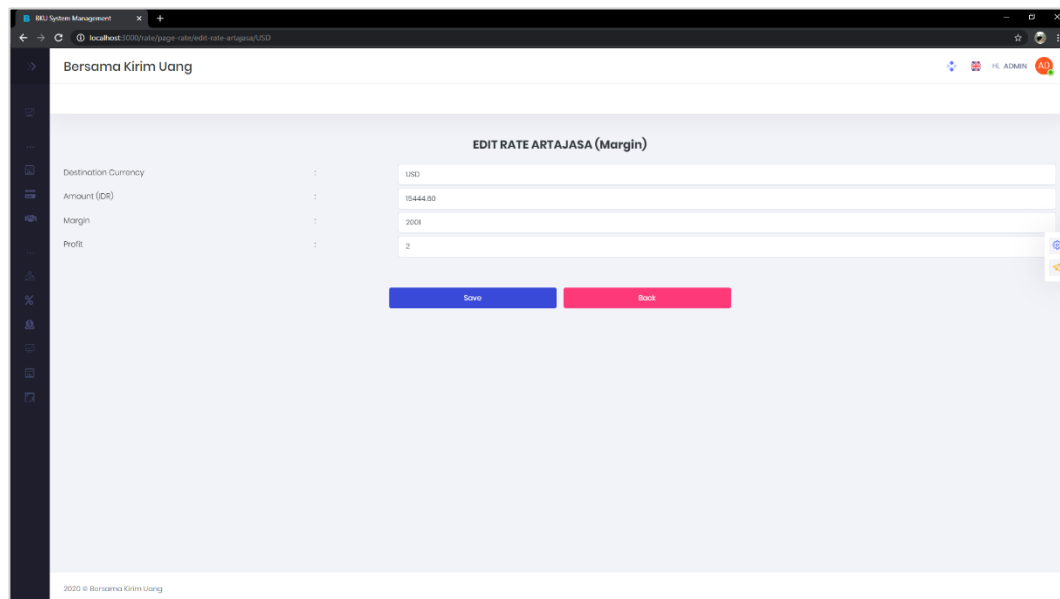
3) Pengujian Halaman *Update* Nilai Tukar Mata Uang

Halaman *update* nilai tukar mata uang secara struktur sama dengan halaman rincian nilai tukar mata uang, namun yang membedakan adalah terdapatnya kolom masukan untuk menampung data baru yang akan dikirim ke basis data guna memperbaharui data yang ada. Masih sama seperti dua halaman sebelumnya, halaman *update* nilai tukar mata uang ini juga hanya dapat diakses ketika pengguna sudah berhasil melakukan proses verifikasi masuk sistem. Alamat URL dari halaman *update* nilai tukar mata uang ini adalah “hostname/rate/page-rate/edit-rate-artajasa/:id_negara”, karena pada pengujian ini menggunakan *local environment* maka URLnya berbentuk seperti yang ada pada Gambar 6.15.



Gambar 6.15 URL update nilai tukar mata uang

Setelah pengguna menekan tombol “Edit” pada halaman rincian nilai tukar mata uang, maka sistem akan mengalihkan pengguna ke halaman *update* ini sesuai dengan negara pada rincian tersebut. Kolom masukan yang dapat diubah *value*-nya adalah kolom margin dan profit saja, selain 2 kolom tersebut tidak dapat diubah atau dalam kata lain menjadi *read-only*. Karena masih dalam *library* ReactJS, maka sistem tidak akan memuat ulang seluruh halaman, melainkan hanya komponen yang diminta.



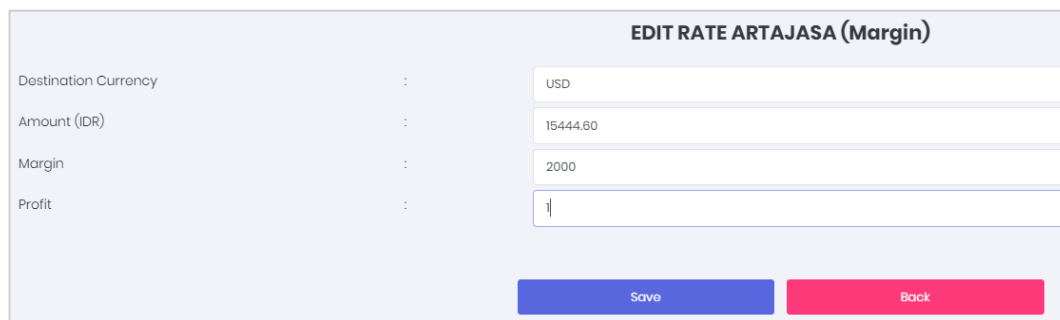
The screenshot shows a web application interface for 'Bersama Kirim Uang'. The main content area is titled 'EDIT RATE ARTAJASA (Margin)'. It contains a form with the following fields and values:

Field	Value
Destination Currency	USD
Amount (IDR)	15444.80
Margin	2001
Profit	2

At the bottom of the form, there are two buttons: 'Save' (blue) and 'Back' (pink). The footer of the application indicates '2020 © Bersama Kirim Uang'.

Gambar 6.16 Tampilan halaman update nilai tukar mata uang

Untuk menguji apakah fungsi *update* dari antarmuka ini berjalan dengan baik, maka akan kita lakukan pembaruan data pada negara Amerika dengan mata uang USD. Dapat kita lihat pada Gambar 6.16 yaitu nilai margin dan profit masing-masing adalah bernilai 2001 dan 2, akan kita ubah menjadi nilai baru yaitu 2000 dan 1.



This close-up view of the form shows the following data:

Field	Value
Destination Currency	USD
Amount (IDR)	15444.80
Margin	2000
Profit	

The 'Save' and 'Back' buttons are visible at the bottom.

Gambar 6.17 Contoh masukan nilai tukar mata uang

Ketika kita menekan tombol “Save” maka fungsi akan berjalan dan ketika sudah selesai maka sistem akan membawa pengguna kembali ke halaman daftar nilai tukar mata uang guna melihat apakah data berhasil diubah dan ditampilkan ke halaman tersebut. Dapat kita lihat pada Gambar 6.18, pada mata uang USD, data yang ditampilkan sudah berubah menjadi data terbaru dari masukan penguji.

Rate Artajasa				
Currency	Amount (IDR)	Margin Kurs	Profit	Actions
PHP	309.11	153	0.5	Detail
CNY	2228.06	500	5	Detail
USD	15444.60	2000	1	Detail
KRW	12.94	50	2	Detail
MYR	3627.57	700	2.5	Detail
				5 rows ▾

Gambar 6.18 Contoh hasil data sudah diperbaharui

BAB VII

KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, maka didapatkan kesimpulan sebagai berikut:

1. Implementasi REST API (*backend service*) yang merupakan bagian dari modul nilai tukar mata pada produk atau layanan pengiriman uang dari dan ke luar negeri telah berhasil dibangun dan digunakan oleh antarmuka sesuai rancangan dan kebutuhan modul.
2. Implementasi antarmuka pengguna yang juga merupakan bagian dari modul nilai tukar mata uang pada produk atau layanan pengiriman uang dari dan ke luar negeri telah berhasil dibangun sesuai rancangan dan kebutuhan modul.
3. Antarmuka pengguna yang menggunakan API dari hasil implementasi REST API mampu melakukan kegiatan manajemen data meliputi melihat dan memperbaharainya.
4. Secara keseluruhan, modul hasil implementasi yang telah dibuat ini telah mampu melaksanakan proses-proses yang sudah dirancang sebelumnya pada tahap analisis dan perancangan serta telah memenuhi kebutuhan sistem (produk) secara keseluruhan.

7.2 Saran

Di bawah ini penulis berikan daftar saran terkait pengembangan yang dapat penulis berikan bagi penelitian di masa mendatang yang memiliki tema serupa agar memberikan hasil yang lebih baik, yaitu:

1. Pada desain antarmuka pengguna, usahakan untuk membuat desain yang lebih interaktif sehingga pengguna tidak merasa bosan saat melihat data yang tampilannya relatif monoton.

2. Penambahan komponen antarmuka yang berupa notifikasi status jika ada proses perubahan data, sehingga pengguna mengetahui data yang diubah berhasil atau gagal.
3. Pendokumentasian kode program yang lebih baik sehingga dapat dibaca dengan lebih mudah oleh pengembang lain.

DAFTAR PUSTAKA

Brown, Kyle, and Bobby Woolf. "Implementation patterns for microservices architectures." *Proceedings of the 23rd Conference on Pattern Languages of Programs*. 2016.

Marks, Audrey. "Third party real-time multi-payment and remittance system." (2003).

Namiot, Dmitry, and Manfred Sneps-Sneppe. "On micro-services architecture." *International Journal of Open Information Technologies* 2.9 (2014): 24-27.

Schwaber, Ken. "Scrum development process." *Business object design and implementation* (1997): 117-134.

Primawati, Anggraeni. "Remitan Sebagai Dampak Migrasi Pekerja Ke Malaysia." *Sosio Konsepsia* 16.2 (2017): 209-222.

Flanagan, David. "JavaScript: the definitive guide." (2006).

Quevedo, Waldemar. "Practical NATS: From Beginner to Pro." (2018).

Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." *Linux journal* 2014.239 (2014): 2.

Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes." *IEEE Cloud Computing* 1.3 (2014): 81-84.

Drake, Joshua D., and John C. Worsley. "Practical PostgreSQL." (2002).

X. Larrucea, I. Santamaria, R. Colomo-Palacios and C. Ebert, "Microservices," in *IEEE Software*, vol. 35, no. 3, pp. 96-100, May/June 2018.

Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

"Remitansi Dan Transfer Dana - Bank Sentral Republik Indonesia". *Bi.Go.Id*, 2020, <https://www.bi.go.id/id/edukasi-perlindungan-konsumen/edukasi/produk-dan-jasa-sp/remitansi/Pages/default.aspx>. Diakses 28 April 2020.

"What Is An API?". *Redhat.Com*, 2020, <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>. Diakses 28 April 2020.

"JS Versions: EcmaScript And Javascript". *Scotch*, 2020, <https://scotch.io/courses/10-need-to-know-javascript-concepts/js-versions-ecmascript-and-javascript>. Diakses 28 April 2020.

"ES5, ES6, ES7, ES8, ES9: What's new in each Version of JavaScript". *Greycampus.Com*, 2020, <https://www.greycampus.com/blog/programming/javascript-versions>. Diakses 28 April 2020.

"Apa Itu React.Js?". *School Of Computer Science*, 2020, <https://socs.binus.ac.id/2019/12/30/apa-itu-react-js/>. Diakses 28 April 2020.

"Getting Started With React - An Overview And Walkthrough Tutorial". *Taniarascia.Com*, 2020, <https://www.taniarascia.com/getting-started-with-react/>. Diakses 28 April 2020.

"Mengenal Konkurensi Dan Paralelisme". *Medium*, 2020, <https://medium.com/pujanggateknologi/mengenal-konkurensi-dan-paralelisme-10746e1b7ab6>. Diakses 29 April 2020.

Kincaid, Jason. "Google's Go: A New Programming Language That's Python Meets C++". *Techcrunch.Com*, 2011, <https://techcrunch.com/2009/11/10/google-go-language/>. Diakses 29 April 2020.

"Language Design FAQ". *golang.org*, 2010, https://golang.org/doc/go_faq.html. Diakses 29 April 2020.

"Why doesn't Go have 'implements' declarations?". *golang.org*, https://golang.org/doc/faq#implements_interface. Diakses 29 April 2020.

Metz, Cade (May 5, 2011). "Google Go boldly goes where no code has gone before". *The Register*, 2011, https://www.theregister.co.uk/2011/05/05/google_go/. Diakses 29 April 2020.

Lardinois, Frederic. "Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows". *TechCrunch*, 2015, <https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows>. Diakses 29 April 2020.

W, Rian. "Gitlab, Layanan Penyimpan Git Gratis Dan Open Source - Codepolitan.Com". *Codepolitan.Com*, 2020, <https://www.codepolitan.com/gitlab-layanan-penyimpan-git-gratis-dan-open-source>. Diakses 29 April 2020.

Dzarrin T, Nuwas. "RANCANG BANGUN MIKROSERVIS SEBAGAI PEMBANGKIT TANDA TANGAN DIGITAL BERBASIS CLOUD COMPUTING". Universitas Gadjah Mada, 2019.

Kurniawan, Oei. "Arsitektur Platform IOT Untuk Pemantauan Kualitas Tidur Berbasis Microservices Dan Event-Driven". Universitas Bina Nusantara, 2019.

Jufry, Fachry. "Perancangan Arsitektur Microservices Untuk Online Travel Agent Pada Pt. Xyz". Universitas Bina Nusantara, 2018.