BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Perusahaan

Toko Indah merupakan perusahaan yang bergerak di bidang retail kebutuhan pokok sehari hari yang berlokasi di Jalan Babakan Ciparay No 36, RT.01/RW.03, Sukahaji, Kecamatan Babakan Ciparay, Kota Bandung.

2.1.1 Profil Toko Indah

Toko Indah berlokasi di Jalan Babakan Ciparay No 36, RT.01/RW.03, Sukahaji, Kecamatan Babakan Ciparay, Kota Bandung, merupakan sebuah usaha ritel yang telah berdiri sejak 2013 oleh Ibu Indah Susanti dan Bapak Ujang Adi Setiawan. Toko Indah awal mulanya menjual produk kebutuhan pokok sehari-hari, seiring waktu berjalan kini Toko Indah juga menjual layanan *top up* untuk berbagai keperluan seperti pulsa, paket data, token listrik, *voucher* internet fisik dan *e-wallet* atau dompet digital.

2.1.2 Visi dan Misi

Toko Indah memiliki visi dan misi dalam menjalankan usahanya yaitu mengutamakan kenyamanan dan kepuasan pelanggan. Adapun yang visi dan misi yang dimiliki Toko Indah adalah sebagai berikut:

1. Visi

   Menjadi Toko retail pilihan utama di kawasan masyarakat sekitar dengan menyediakan produk berkualitas, layanan pelanggan terbaik dan harga yang kompetitif.

2. Misi
   a. Menjual produk yang terjangkau dengan kualitas baik dan memenuhi kebutuhan masyarakat.
   b. Memberikan pelayanan yang terbaik kepada pelanggan.

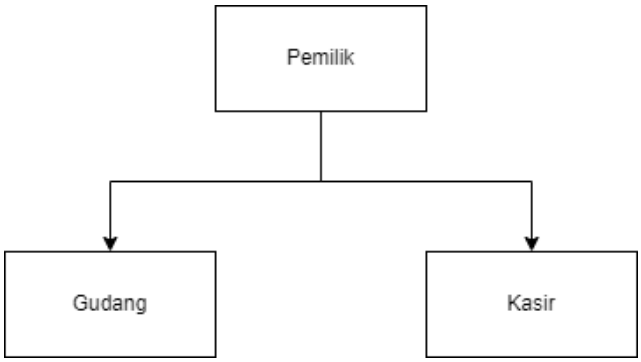
2.1.3 Logo Toko Indah

Logo merupakan suatu identitas ataupun ciri dari suatu usaha, Adapun Logo yang dimiliki oleh Toko Indah dapat dilihat pada Gambar 2.1 Toko Indah.

Gambar 2.1 Logo Toko Indah

2.1.4 Struktur Organisasi

Struktur organisasi adalah pola hubungan antara bagian-bagian organisasi, atau menggambarkan dengan jelas bagaimana kegiatan pekerjaan terpisah satu sama lain dan bagaimana hubungan antara aktivitas dan fungsi dibatasi. Toko Indah memiliki struktur organisasi sebagai berikut:



Gambar 2.2 Struktur Organisasi Toko Indah

2.1.5 Deskripsi Tugas

Deskripsi tugas digunakan untuk menentukan tugas, wewenang, dan tanggung jawab setiap jabatan. Dalam struktur organisasi di atas, tugas dan tanggung jawab masing-masing jabatan adalah sebagai berikut:

1. Pemilik Toko Indah berperan penting dalam memastikan kesuksesan dan keberlanjutan bisnis. Pemilik bertindak sebagai perencana, pengawas, penilai, menetapkan kebijakan dan tanggung jawab, dan secara teratur meninjau penjualan, pembelian, dan persediaan barang.
2. Gudang bertanggung jawab untuk melaporkan kepada pemilik toko tentang kondisi stok produk yang tersedia dan membuat daftar permintaan untuk produk yang tidak tersedia.

3. Kasir bertanggung jawab untuk melakukan transaksi pembelian. Kasir juga harus melaporkan kepada staf gudang tentang ketersediaan stok produk.

2.2 Landasan Teori

Landasan teori adalah komponen penting dalam penelitian karena membantu penelitian menjadi lebih terarah. Berikut merupakan beberapa landasan teori yang digunakan dalam pembangun aplikasi *top up* digital studi kasus (Toko Indah).

2.2.1 Aplikasi

Pengertian aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai dengan kemampuan yang dimilikinya. Aplikasi adalah perangkat komputer yang dapat digunakan oleh pengguna dan memiliki fitur yang diperlukan untuk mendukung kebutuhan pengguna [13].

2.2.2 Top up Digital

Top up digital secara umum adalah proses pengisian ulang saldo digital pada berbagai jenis layanan [14]. Dalam konteks ini, terdapat beberapa layanan yang dapat diisi ulang, seperti:

1. *Top up* Pulsa Elektrik merupakan proses pengisian ulang saldo kartu prabayar telepon seluler.
2. *Top up* Paket data adalah proses mengisi ulang kuota internet pada kartu prabayar.
3. *Top up E-wallet* adalah proses pengisian ulang saldo pada layanan dompet digital seperti GoPay, Ovo, Dana, dan ShopeePay.
4. *Top up* Masa Aktif adalah proses pengisian ulang yang bertujuan untuk memperpanjang masa aktif kartu prabayar pada layanan telepon seluler.
5. *Top up Voucher* Fisik adalah cara mengisi ulang saldo digital dengan menggunakan *voucher* atau kartu fisik. Dalam hal ini *voucher* fisik sering digunakan untuk mengisi ulang saldo pada berbagai layanan paket data internet.
6. *Top up* Token listrik adalah proses mengisi ulang nilai listrik dengan menggunakan token listrik, yang biasanya terdiri serangkaian angka atau kode unik yang dimasukan ke dalam meter prabayar.

2.2.3 Supplier

Pemasok atau *supplier* adalah perusahaan atau individu yang menyediakan barang atau jasa untuk memenuhi permintaan perusahaan lain untuk memenuhi kebutuhan dan aktivitas operasionalnya. Perusahaan membutuhkan pemasok untuk menjalankan bisnisnya, karena perusahaan tidak bisa beroperasi secara mandiri. Sebuah bisnis *top up* digital tentu akan membutuhkan pemasok layanan untuk menjalankan bisnisnya agar lancar [15].

2.2.4 Multi Supplier

Multi dapat diartikan banyak, lebih dari satu, lebih dari dua dan berlipat ganda [16]. Sedangkan Multi *supplier* merujuk pada konsep sebuah entitas bisnis atau organisasi menggunakan lebih dari satu pemasok untuk memenuhi layanan yang diperlukan dalam operasinya [17]. Hal ini dilakukan untuk memastikan bahwa layanan yang diperlukan dalam operasi bisnis dapat terpenuhi dengan baik.

2.2.5 Harga Kompetitif

Harga adalah sejumlah uang yang dibayarkan oleh konsumen untuk memperoleh, memiliki, dan memanfaatkan suatu produk atau jasa. Ini mencakup semua nilai yang ditukar oleh konsumen untuk mendapatkan manfaat dari barang atau jasa tersebut. Sedangkan harga kompetitif adalah harga yang ditetapkan oleh perusahaan dengan tujuan agar mampu bersaing dengan produsen atau distributor lain. Harga ini dianggap kompetitif ketika konsumen merasa bahwa biaya yang mereka keluarkan sepadan dengan produsen/distributor/*supplier* lain. Harga kompetitif mencakup beberapa indicator, seperti harga yang sesuai dengan kualitas, harga yang mampu bersaing dengan produk sejenis, dan harga yang terjangkau [18].

Dalam penelitian ini, harga kompetitif didefinisikan sebagai harga *supplier* baru yang lebih murah atau setara dibandingkan harga yang ditawarkan oleh *supplier* sebelumnya. Untuk merumuskan konsep ini secara matematis, dapat menggunakan beberapa variable sebagai berikut:

1. Harga Lama (H_L) : Harga dari *supplier* lama yang digunakan oleh Toko Indah.
2. Harga Baru (H_B) : Harga baru yang ditawarkan *supplier* baru.
3. Persentase Kompetitif (P): Persentase harga yang kompetitif.

4. Jumlah Produk Kompetitif: Total produk yang memiliki harga baru yang lebih murah dibandingkan harga lama.
5. Total Produk: Jumlah total produk yang diobservasi.

Definisi harga kompetitif dan tidak kompetitif dapat dirumuskan sebagai berikut:

Harga Kompetitif : $(H_B) \leq (H_L)$
Harga Tidak Kompetitif : $(H_B) > (H_L)$ (1)

Persentase harga kompetitif pada hari pertama $(P1)$ dapat dihitung dengan rumus:

$$(P1) = (\frac{Jumlah\ Produk\ Kompetitif}{Total\ Produk}) * 100\% \quad (2)$$

Untuk mendapatkan nilai hasil persentase akhir dari pengamatan selama tiga hari, dapat menggunakan rumus sebagai berikut:

$$(P) = (\frac{P1 + P2 + P3}{Total\ Hari}) \quad (3)$$

Dimana Adapun penjelasan dari rumus diatas adalah sebagai berikut:

1. P = Nilai hasil persentase akhir yang dicari.
2. P1 = Persentase harga kompetitif pada hari pertama.
3. P2 = Persentase harga kompetitif pada hari kedua.
4. P3 = Persentase harga kompetitif pada hari ketiga.
5. Total Hari = Jumlah hari yang digunakan untuk pengamatan (dalam hal ini, 3 hari).

Rumus di atas memberikan persentase rata-rata harga kompetitif yang diobservasi selama tiga hari, yang memungkinkan analisis lebih komprehensif terhadap tren harga yang ditawarkan oleh *supplier* baru dibandingkan dengan *supplier* sebelumnya.

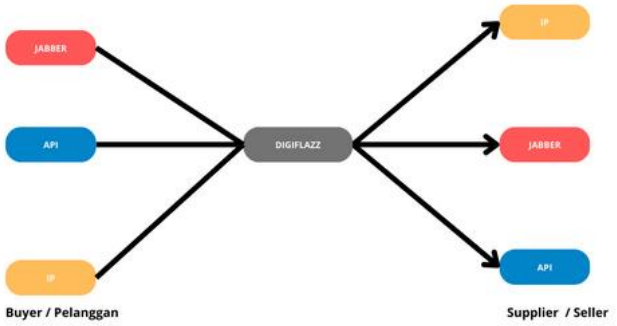
2.2.6 Digiflazz

Gambar 2.3 Logo Digiflazz

Sumber gambar: https://digiflazz.com/

Digiflazz adalah pasar pertama untuk produk digital dan pulsa di Indonesia, Digiflazz menawarkan berbagai layanan, seperti paket data, token listrik, voucher game, dan pulsa untuk semua operator. Digiflazz telah berkerjasama dengan banyak perusahaan digital, *switching* dan *aggregator* pulsa terkemuka di Indonesia. Melalui kolaborasi ini Digiflazz menjadi solusi praktis bagi konsumen yang ingin membeli produk digital karena memungkinkan pengguna terintegrasi dengan berbagai penjual yang tergabung dalam Digiflazz. Dengan Digiflazz pengguna tidak perlu mencari *supplier* pulsa atau produk digital secara terpisah. Beberapa kelebihan menggunakan Digiflazz:

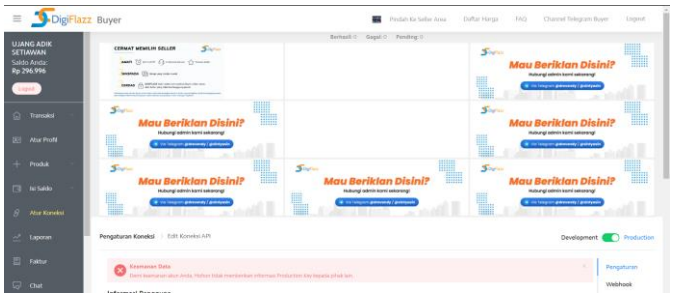
1. Mendapatkan beragam *supplier* dan memilih *supplier* sesuai dengan kebutuhan.
2. Cukup *deposit* di satu tempat dan bisa melakukan transaksi untuk banyak *supplier*.
3. Satu format transaksi dan jawaban untuk transaksi kebanyak *supplier*.
4. *Cross connection* untuk transaksi kebanyak *supplier* dengan jalur yang berbeda-beda, untuk lebih jelasnya bisa dilihat pada Gambar 2.4 Cross Connection Digiflazz.



Gambar 2.4 Cross Connection Digiflazz

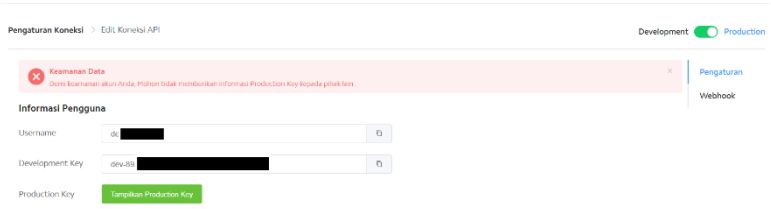
Dalam penelitian ini untuk koneksi menggunakan API, adapun penggunaannya ada beberapa persiapan yang mesti disiapkan sebelum menjalakan proses integrasi. Langkah-langkah persiapannya adalah sebagai berikut:

1. Kunjungi halaman Pengaturan Koneksi API, seperti pada Gambar 2.5 Halaman Pengaturan Koneksi Api Digiflazz.



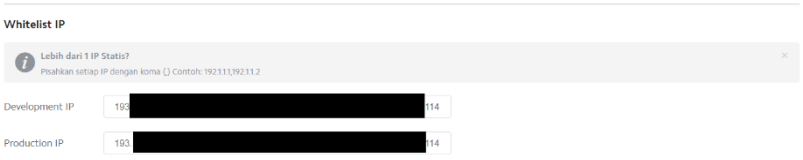
Gambar 2.5 Halaman Pengaturan Koneksi Api Digiflazz

2. Geser ke bawah untuk mendapat username serta api key, seperti pada Gambar 2.6 Mendapatkan Username dan API Key Digiflazz.



Gambar 2.6 Mendapatkan Username dan API Key Digiflazz

3. Selanjutnya untuk menambahkan *whitelist* API yaitu untuk menentukan IP yang diizinkan mengakses mode pengembangan (*development*) dan produksi (*production*), seperti pada Gambar 2.7 Setting Whitelist IP Address Digiflazz.



Gambar 2.7 Setting Whitelist IP Address Digiflazz

4. Setelah pengaturan koneksi API selesai, penting untuk memperhatikan saat melakukan permintaan, bagian *header Content-Type* harus berupa

application/json, dan seluruh data harus dikirim menggunakan metode *POST*. Respons akan diterima dalam bentuk JSON.

Berikut adalah integrasi yang tersedia pada Digiflazz. Integrasi ini dimaksudkan agar aplikasi yang akan dibangun dapat terhubung dengan berbagai layanan yang disediakan oleh Digiflazz. Adapun daftar integrasi tersebut akan dijelaskan sebagai berikut.

1. Cek Saldo

Cek saldo adalah integrasi yang digunakan untuk mendapatkan informasi saldo akun Digiflazz. Integrasi ini sangat penting dalam memastikan jumlah saldo terakhir yang tersedia. Detail integrasi untuk cek saldo dapat dilihat pada Tabel 2.1 – Tabel 2.3.

Tabel 2.1 Endpoint Cek Saldo Digiflazz

Metode HTTP	Endpoint
POST	https://api.digiflazz.com/v1/cek-saldo

Berikut adalah data permintaan yang perlu dikirim untuk melakukan cek saldo dapat dilihat pada Tabel 2.2 Data Permintaan Cek Saldo Digiflazz.

Tabel 2.2 Data Permintaan Cek Saldo Digiflazz

No	Parameter	Deskripsi	Tipe Data	Wajib
1	cmd	Value:”deposit”	String	Ya
2	username	Username yang telah diatur di pengaturan koneksi api	String	Ya
3	sign	Signature dengan formula md5(username+apiKey+”depo”)	String	Ya

Berikut adalah data respons yang diterima dari permintaan pengecekan saldo, yang dibungkus dalam *object* data. Data ini dapat dilihat pada Tabel 2.3 Data Respons Cek Saldo Digiflazz.

Tabel 2.3 Data Respons Cek Saldo Digiflazz

No	Parameter	Deskripsi	Tipe Data	Wajib
1	deposit	Sisa Deposit Akun Digiflazz	Float	Ya

2. Daftar Produk

Daftar produk atau daftar layanan dalam dokumentasi Digiflazz adalah integrasi yang digunakan untuk mendapatkan daftar produk yang telah diatur. Detail integrasi untuk Daftar Produk dapat dilihat pada Tabel 2.4 – Tabel 2.6.

Tabel 2.4 Endpoint Daftar Produk Digiflazz

Metode HTTP	Endpoint
POST	https://api.digiflazz.com/v1/price-list

Berikut adalah data yang perlu dikirim untuk melakukan permintaan daftar layanan atau produk dapat dilihat pada Tabel 2.5 Data Permintaan Daftar Produk.

Tabel 2.5 Data Permintaan Daftar Produk

No	Parameter	Deskripsi	Tipe Data	Wajib
1	cmd	Value: ”*prepaid*”	String	Ya
2	username	Username yang telah diatur di pengaturan koneksi *API.*	String	Ya
3	sign	Signature dengan formula md5(username+apiKey+”pricelist”).	String	Ya

Berikut adalah data respons yang diterima dari permintaan daftar produk, yang dibungkus dalam *object* data. Data ini dapat dilihat pada Tabel 2.6 Data Respons Daftar Produk.

Tabel 2.6 Data Respons Daftar Produk

No	Parameter	Deskripsi	Tipe Data	Wajib
1	product_name	Nama produk.	String	Ya
2	category	Nama kategori.	String	Ya

No	Parameter	Deskripsi	Tipe Data	Wajib
3	brand	Nama brand.	String	Ya
4	type	Nama tipe.	String	Ya
5	seller_name	Nama *supplier* atau *seller*.	String	Ya
6	price	Harga produk yang ditentukan oleh *supplier*.	Int	Ya
7	buyer_sku_code	Kode produk yang telah diatur oleh *buyer*.	String	Ya
8	buyer_product_status	Status produk sebagai *buyer.*	Boolean	Ya
9	seller_product_status	Status produk dari *supplier* atau *seller*.	Boolean	Ya
10	unlimited_stock	Penentu apakah stok terbatas atau tidak.	Boolean	Ya
11	stock	Sisa stok *seller* (dapat diabaikan jika unlimited_stock bernilai *true*).	String	Ya
12	multi	Transaksi dapat dilakukan lebih dari satu kali ke denom atau nomor tujuan yang sama di dalam satu hari.	Boolean	Ya
13	start_cut_off	Jam mulai *cut off* (format: hh:mm)	String	Ya
14	end_cut_off	Jam berakhir *cut off* (format: hh:mm).	String	Ya
15	desc	Deskripsi produk.	String	Ya

3. Top up

Top up atau integrasi *top up* adalah proses untuk melakukan permintaan *top up* baru yang akan diteruskan kepada *supplier*. Proses ini juga dapat digunakan untuk memeriksa detail *voucher*, detail pelanggan *e-wallet*, dan status transaksi,

dengan menyesuaikan nilai *buyer_sku_kode.* Untuk keperluan memeriksa status transaksi mesti diperhatikan pada parameter ref_id dimana parameter tersebut mesti dikirim sesuai dengan transaksi yang pernah dilakukan sebelumnya. Detail integrasi untuk *top up* dapat dilihat pada Tabel 2.7 – Tabel 2.9.

Tabel 2.7 Endpoint Top up Digiflazz

Metode HTTP	EndPoint
POST	https://api.digiflazz.com/v1/transaction

Berikut adalah data yang perlu dikirim untuk melakukan permintaan *top up* dapat dilihat pada Tabel 2.8 Permintaan Data Top up Digiflazz.

Tabel 2.8 Data Permintaan Top up Digiflazz

No	Parameter	Deskripsi	Tipe Data	Wajib
1	username	*Username* yang telah diatur di pengaturan koneksi api.	String	Ya
2	buyer_sku_code	Kode produk disesuaikan dengan keperluan untuk *top up*, memeriksa detail voucher, detail pelanggan *e-wallet*.	String	Ya
3	customer_no	Nomor pelanggan, nomor seri atau nomor tujuan.	String	Ya
4	ref_id	Ref_id unik jika ingin melakukan cek status gunakan ref_id yang pernah digunakan.	String	Ya
5	sign	Signature dengan formula md5(username+apiKey+ref_id)	String	Ya
6	testing	*Value true* apabila ingin melakukan *development.*	Boolean	Tidak
7	max_price	*Limit* Harga *Max*	Int	Tidak
8	cb_url	*Callback* URL.	String	Tidak

No	Parameter	Deskripsi	Tipe Data	Wajib
9	allow_dot	*Value true* apabila ingin Parameter customer_no berisi titik, optimal digunakan ke *Seller* dengan koneksi selain Jabber.	Boolean	Tidak

Berikut adalah data respons yang diterima dari permintaan *top up*, yang dibungkus dalam *object* data. Data ini dapat dilihat pada Tabel 2.9 Data Respons Top up Digiflazz.

Tabel 2.9 Data Respons Top up Digiflazz

No	Parameter	Deskripsi	Tipe Data	Wajib
1	ref_id	Re ID Unik yang dikirim.	String	Ya
2	customer_no	Nomor Pelanggan.	String	Ya
3	buyer_sku_code	Kode produk.	String	Ya
4	message	Deskripsi status transaksi.	String	Ya
5	status	Status Transaksi Sukses, Pending, Gagal.	String	Ya
6	rc	Respons Kode yang didapatkan.	String	Ya
7	sn	Serial Number transaksi atau detail *voucher*, detail pelanggan *e-wallet.*	String	Tidak
8	buyer_last_saldo	Saldo Terakhir setelah transaksi terjadi.	Float	Tidak
9	price	Harga produk saat transaksi terjadi.	Int	Ya
10	tele	Telegram *Supplier*.	String	Tidak
11	wa	Whatsapp *Supplier*.	String	Tidak

4. Inquiry PLN

Inquiry PLN adalah integrasi untuk mendapatkan detail pelanggan PLN. Integrasi ini sangat penting dalam transaksi token PLN dan memastikan bahwa pelanggan yang dituju telah sesuai dengan yang diharapkan. Detail integrasi untuk *Inquiry* PLN dapat dilihat pada Tabel 2.10 Endpoint Inquiry PLN.

Tabel 2.10 Endpoint Inquiry PLN

Metode HTTP	EndPoint
POST	https://api.digiflazz.com/v1/transaction

Berikut adalah data yang perlu dikirim untuk melakukan permintaan *inquiry* PLN dapat dilihat pada Tabel 2.11 Data Permintaan Inquiry PLN.

Tabel 2.11 Data Permintaan Inquiry PLN

No	Parameter	Deskripsi	Tipe Data	Wajib
1	commands	*Value*:"pln-subscribe"	String	Ya
2	customer_no	Id Pelanggan atau nomor meter	String	Ya

Berikut adalah data respons yang diterima dari hasil permintaan *inquiry* PLN, yang dibungkus dalam *object* data. Data ini dapat dilihat pada Tabel 2.12 Data Respons Inquiry PLN.

Tabel 2.12 Data Respons Inquiry PLN

No	Parameter	Deskripsi	Tipe Data	Wajib
1	customer_no	Id Pelanggan / Nomor Meter.	String	Ya
2	meter_no	Nomor meter.	String	Ya
3	subscriber_id	Id pelanggan.	String	Ya
4	name	Nama Pelanggan PLN.	String	Ya
5	segment_power	Segment power atau daya pelanggan PLN.	String	Ya

Setelah detail integrasi di atas telah dibahas, perlu diperhatikan bahwa dalam integrasi *top up* terdapat *response code* atau RC dimana setiap RC memiliki keterangan. Berikut adalah detail dari *response code* yang dapat dilihat pada Tabel 2.13 Respons Kode Digiflazz.

Tabel 2.13 Respons Kode Digiflazz

No	RC	Pesan	Status	Terbentuk Transaksi
1	00	Transaksi Sukses.	Sukses	Ya
2	01	*Timeout.*	Gagal	Ya
3	02	Transaksi Gagal.	Gagal	Ya
4	03	Transaksi *Pending*.	Pending	Ya
5	40	*Payload Erro*r.	Gagal	Tidak
6	41	*Signature* tidak *valid*.	Gagal	Tidak
7	42	Gagal memproses API *Buyer*.	Gagal	Tidak
8	43	SKU tidak di temukan atau *Non*-Aktif.	Gagal	Tidak
9	44	Saldo tidak cukup.	Gagal	Tidak
10	45	IP Anda tidak kami kenali.	Gagal	Tidak
11	47	Transaksi sudah terjadi di buyer lain.	Gagal	Tidak
12	49	Ref ID tidak unik.	Gagal	Tidak
13	50	Transaksi Tidak Ditemukan.	Gagal	Ya
14	51	Nomor Tujuan Diblokir.	Gagal	Ya
15	52	*Prefix* Tidak Sesuai Dengan Operator.	Gagal	Ya
16	53	Produk Seller Sedang Tidak Tersedia.	Gagal	Ya
17	54	Nomor Tujuan Salah.	Gagal	Ya
18	55	Produk Sedang Gangguan.	Gagal	Ya
19	56	*Limit* saldo seller.	Gagal	Tidak
20	57	Jumlah Digit Kurang Atau Lebih.	Gagal	Ya
21	58	Sedang *Cut Off*.	Gagal	Ya
22	59	Tujuan di Luar Wilayah/Cluster.	Gagal	Ya
23	60	Tagihan belum tersedia.	Gagal	Ya
24	61	Belum pernah melakukan deposit.	Gagal	Tidak
25	62	*Seller* sedang mengalami gangguan.	Gagal	Tidak
26	63	Tidak *support* transaksi *multi*.	Gagal	Tidak

No	RC	Pesan	Status	Terbentuk Transaksi
27	64	Tarik tiket gagal, coba nominal lain atau hubungi admin.	Gagal	Tidak
28	65	Limit transaksi multi.	Gagal	Tidak
29	66	*Cut Off* (Perbaikan Sistem *Seller*).	Gagal	Tidak
30	67	*Seller* belum terverfikasi.	Gagal	Tidak
31	68	Stok habis.	Gagal	Tidak
32	69	Harga *seller* lebih besar dari ketentuan harga *Buyer*.	Gagal	Tidak
33	70	*Timeout* Dari *Biller*.	Gagal	Ya
34	71	Produk Sedang Tidak Stabil.	Gagal	Ya
35	72	Lakukan *Unreg* Paket Dahulu.	Gagal	Ya
36	73	Kwh Melebihi Batas.	Gagal	Ya
37	74	Transaksi *Refund*.	Gagal	Ya
38	80	Akun Anda telah diblokir oleh *Seller*.	Gagal	Tidak
39	81	Seller ini telah diblokir oleh Anda.	Gagal	Tidak
40	82	Akun Anda belum terverfikasi.	Gagal	Tidak
41	83	Anda telah mencapai limitasi pengecekan *pricelist*, silahkan coba beberapa saat lagi.	Gagal	Tidak
42	84	Nominal tidak valid.	Gagal	Ya
43	99	DF *Router Issue*.	Pending	Ya

Selain integrasi API, terdapat aspek lain yang perlu dipersiapkan, yaitu *webhook*. *Webhook* adalah metode di mana sebuah aplikasi menyediakan data *real-time* kepada aplikasi lain melalui panggilan balik HTTP yang dipicu oleh suatu peristiwa. Berikut detail *Webhook* yang dikirimkan oleh Digiflazz kepada klien yang dapat dilihat pada Tabel 2.14 dan Tabel 2.15.

Tabel 2.14 Webhook Header Digiflazz

No	Header	Deskripsi
1	X-Digiflazz-Event	Nama tipe *event* yang menyebabkan *event* di kirim (*create*/*update*).
2	X-Digiflazz-Delivery	Sebuah ID untuk mengidentifikasi pengiriman.
3	X-Hub-Signature	Sebuah *HMAC hex* yang berasal dari *response body*. *Header* ini akan dikirimkan jika *webhook* yang di set menggunakan secret. *HMAC hex* merupakan hasil dari fungsi *hash sha1* dan *HMAC key*.

Berikut adalah detail data yang diterima dari *webhook* Digiflazz. Data ini akan dibungkus oleh *object* data Detail dapat dilihat pada Tabel 2.15 Webhook Data Dikirim Digiflazz.

Tabel 2.15 Webhook Data Dikirim Digiflazz

No	Parameter	Deskripsi	Tipe Data	Wajib
1	trx_id	Transaksi ID Digiflazz.	String	Ya
2	ref_id	Ref ID Unik yang dikirim.	String	Ya
3	customer_no	Nomor Pelanggan.	String	Ya
4	buyer_sku_code	Kode produk.	String	Ya
5	message	Deskripsi status transaksi.	String	Ya
6	status	Status Transaksi Sukses, Pending, Gagal.	String	Ya
7	rc	Respon Kode yang didapatkan.	String	Ya
8	sn	Serial Number transaksi .	String	Tidak
9	buyer_last_saldo	Saldo Terakhir setelah transaksi terjadi.	Float	Tidak
10	price	Harga produk saat transaksi terjadi.	Int	Ya

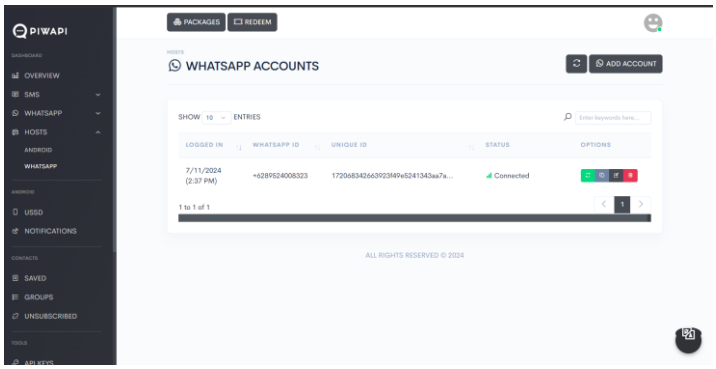
No	Parameter	Deskripsi	Tipe Data	Wajib
11	tele	Telegram *Supplier*.	String	Tidak
12	wa	Whatsapp *Supplier*.	String	Tidak

2.2.7 Piwapi

Piwapi merupakan penyedia layanan *Whatsapp Gateway* API yang memungkinkan aplikasi untuk terhubung dengan WhatsApp dan berkomunikasi melalui API yang disediakan. Dengan sistem ini, aplikasi dapat terhubung ke WhatsApp untuk bertukar data dan informasi secara otomatis, mendukung kebutuhan bisnis dalam komunikasi dan pemasaran [19].

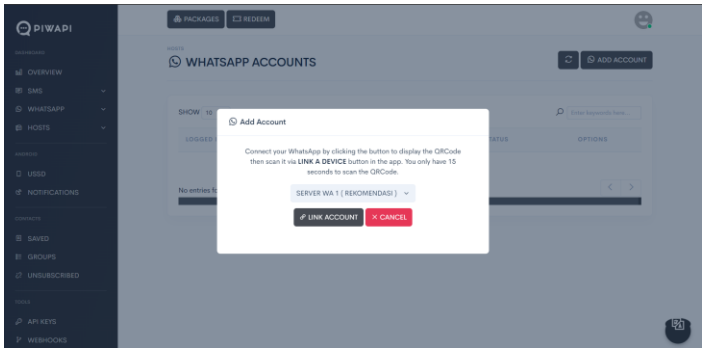
Dalam penggunaannya, terdapat beberapa persiapan yang perlu disiapkan sebelum melakukan integrasi. Berikut adalah detail persiapan.

1. Masuk ke halaman Host WhatsApp, seperti yang ditunjukkan pada Gambar 2.8 Halaman Host Whatsapp Account Piwapi.



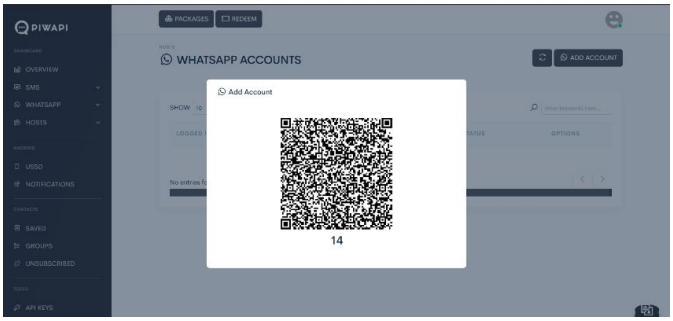
Gambar 2.8 Halaman Host Whatsapp Account Piwapi

2. Tambahkan akun WhatsApp, seperti yang ditunjukkan pada Gambar 2.9 Menambah Akun Whatsapp Piwapi.



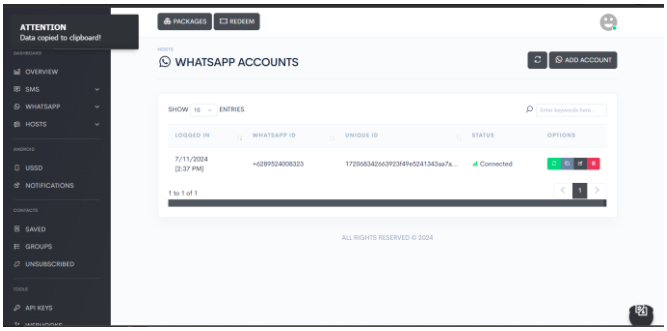
Gambar 2.9 Menambah Akun Whatsapp Piwapi

3. Pindai *QR Code* dengan masuk ke dalam WhatsApp pengguna, kemudian pindai *QR Code* yang ada di halaman Host WhatsApp, seperti yang ditunjukkan pada Gambar 2.10 Memindai QR Code Whatsapp Piwapi.



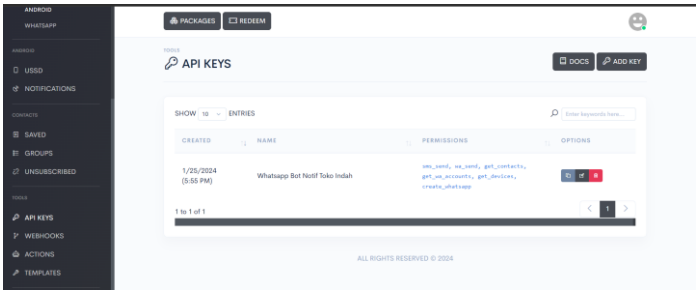
Gambar 2.10 Memindai QR Code Whatsapp Piwapi

4. Setelah berhasil ditambahkan, salin *unique id* yang nantinya akan digunakan untuk identifikasi whatsapp pengirim, seperti yang ditunjukkan pada Gambar 2.11 Salin Unique ID Whatsapp Piwapi.



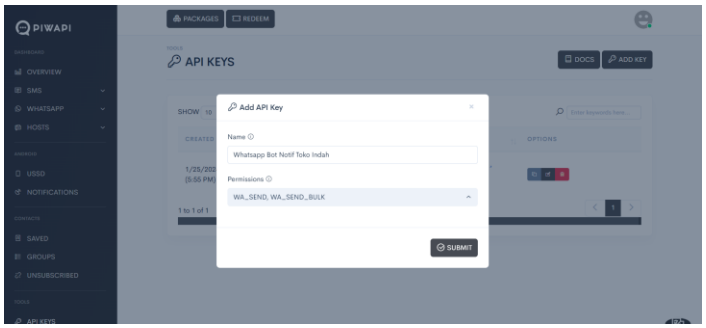
Gambar 2.11 Salin Unique ID Whatsapp Piwapi

5. Setelah itu, masuk ke halaman API Key untuk mendapatkan API, seperti yang ditunjukkan pada Gambar 2.12 Halaman Api Key Piwapi.



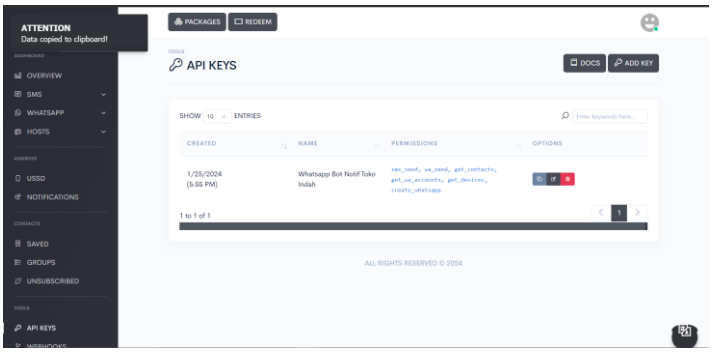
Gambar 2.12 Halaman Api Key Piwapi

6. Setelah masuk ke halaman API Key, klik Add Key dan masukkan pengaturan yang diinginkan untuk menentukan apa saja yang dapat dilakukan oleh key tersebut, seperti yang ditunjukkan pada Gambar 2.13 Setting Permission Whatsapp Piwapi.



Gambar 2.13 Setting Permission Whatsapp Piwapi

7. Untuk mendapatkan key tersebut, klik ikon *copyclip*, seperti yang ditunjukkan pada Gambar 2.14 Salin Api Key Piwapi.



Gambar 2.14 Salin Api Key Piwapi

8. Setelah persiapan API selesai, penting untuk memperhatikan pengiriman data. Data harus dikirim menggunakan metode *POST*. Respons akan diterima dalam bentuk *JSON*.

Setelah persiapan selesai, langkah selanjutnya adalah integrasi untuk melakukan pengiriman pesan ke satu tujuan. Detail untuk mengirim pesan WhatsApp dapat dilihat pada Tabel 2.16 – 2.18.

Tabel 2.16 Endpoint Kirim Pesan Whatsapp Piwapi

Metode HTTP	EndPoint
POST	https://piwapi.com/api/send/whatsapp

Berikut adalah data yang perlu dikirim untuk melakukan permintaan kirim pesan whatsapp dapat dilihat pada Tabel 2.17 Data Permintaan Kirim Pesan Whatsapp Piwapi.

Tabel 2.17 Data Permintaan Kirim Pesan Whatsapp Piwapi

No	Parameter	Deskripsi	Tipe Data	Wajib
1	secret	*API secret* yang disalin dari halaman (*Tools* -> API Keys).	String	Ya
2	account	Akun WhatsApp yang ingin digunakan untuk mengirim pesan, dapat diperoleh dari host Whatsapp.	String	Ya
3	recipient	Nomor ponsel penerima, mendukung nomor berformat E.164 dan nomor lokal menggunakan kode negara dari pengaturan profil. Contoh untuk Filipina: E.164:	String	Ya

No	Parameter	Deskripsi	Tipe Data	Wajib
		+639184661533, Lokal: 09184661533.		
4	type	Tipe pesan WhatsApp. Nilai *default*: text. Nilai yang diperbolehkan: "text", "media", "document".	String	Tidak
5	message	Pesan atau keterangan yang ingin dikirim.	String	Ya
6	priority	Jika ingin mengirim pesan sebagai prioritas, akan dikirim segera. 1 untuk ya dan 2 untuk tidak. Nilai *default*: 2.	Int	Tidak
7	media_file	Ini hanya untuk pesan tipe "media" dan "button". File media yang ingin dilampirkan dalam pesan WhatsApp, mendukung file jpg, png, gif, mp4, mp3, dan ogg.	File	Tidak
8	media_url	Ini hanya untuk pesan tipe "media" dan "button". URL file media, gunakan tautan langsung ke file media. Akan diunduh dan dilampirkan dalam pesan WhatsApp, juga mendukung file jpg, png, gif, mp4, mp3, dan ogg.	String	Tidak
9	media_type	Ini hanya untuk pesan tipe "media". Hanya perlu memasukkan parameter ini jika	String	Tidak

No	Parameter	Deskripsi	Tipe Data	Wajib
		menggunakan "media_url" alih-alih "media_file". Perlu mendeklarasikan tipe file media di URL yang disediakan. Nilai yang diperbolehkan: "image", "audio", "video"		
10	document_file	Ini hanya untuk pesan tipe "document". File dokumen yang ingin dilampirkan dalam pesan WhatsApp, mendukung file pdf, xls, xlsx, doc, dan docx.	file	Tidak
11	document_url	Ini hanya untuk pesan tipe "document". URL file dokumen, gunakan tautan langsung ke file dokumen. Akan diunduh dan dilampirkan dalam pesan WhatsApp, juga mendukung file pdf, xls, xlsx, doc, dan docx.	String	Tidak
12	document_name	Ini hanya untuk pesan tipe "document" dengan "document_url". Nama file dokumen, sertakan ekstensi file. Contoh: dokumen.pdf.	String	Tidak
13	document_type	Ini hanya untuk pesan tipe "document". Hanya perlu memasukkan parameter ini jika menggunakan "document_url" alih-alih "document_file". Perlu	String	Tidak

No	Parameter	Deskripsi	Tipe Data	Wajib
		mendeklarasikan tipe file dokumen di URL yang disediakan. Nilai yang diperbolehkan: "pdf", "xls", "xlsx", "doc", "docx"		

Berikut adalah data respons yang diterima dari permintaan kirim pesan Whatsapp. Data ini dapat dilihat pada Tabel 2.18 Data Respons Kirim Pesan Piwapi.

Tabel 2.18 Data Respons Kirim Pesan Piwapi

No	Parameter	Deskripsi	Tipe Data	Wajib
1	status	Status memiliki beberapa daftar diantaranya : 200 = *Success* 400 = *Invalid parameters* 401 = *Invalid API secret* 403 = *Access denied* 404 = *WhatsApp account doesn't* exist 500 = *Something went wrong*.	Number	Ya
2	message	Pesan yang diterima.	String	Ya
3	data	Data disini berupa array data.	Array	Ya

2.2.8 Socket.io

Socket.io adalah pustaka berbasis bahasa Javascript. Pustaka ini merupakan pengembangan dari protokol *WebSocket native* yang memiliki fitur *low-latency*, berkomunikasi dua arah (*bidirectional*), serta tambahan fitur *fallback* yang membuat koneksi lebih konsisten. Socket.IO juga dirancang lebih mudah digunakan karena didesain dengan *high-level syntax* [20].

Socket.io terdiri dari dua bagian utama yaitu bagian *server* dan bagian *client* dimana bagian *server* berjalan di node.js dan klien berjalan di browser lain atau perangkat lain. Komunikasi antara *server* dan *client* dilakukan menggunakan *protocol Websocket*. Socket IO menggunakan model *event-driven* yang memungkinkan *server* untuk saling mengirim pesan atau data dalam bentuk *event* atau peristiwa. Setiap *event* dapat membawa data yang terkait dan dapat ditangani oleh *event handler* yang telah ditentukan.

Berikut adalah contoh penggunaan Socket.IO bagian *server* di Node.js, seperti yang terlihat dalam Tabel 2.19 Script Server Socket.IO:

Tabel 2.19 Script Server Socket.IO

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
io.on('connection', (socket) => {
  socket.on('message', (data) => {
    console.log('Message received:', data);
    io.emit('message', data);
  });
});
const PORT = process.env.PORT || 3000;
server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Berikut adalah contoh penggunaan Socket.IO bagian *client*, seperti yang terlihat dalam Tabel 2.20 Script Client Socket.IO:

Tabel 2.20 Script Client Socket.IO

```
<script src="/socket.io/socket.io.js"></script>
  <script>
    document.addEventListener('DOMContentLoaded', (event) => {
      const socket = io('http://domainwebsocket.com');

      socket.on('message', (data) => {
        console.log(data);
      });
    });
</script>
```

2.2.9 Hypertext Prepocessor (PHP)

PHP atau *Hypertext Prepocessor* adalah sebuah bahasa pemrograman *scripting* yang bersifat *open source* yang dirancang untuk pengembangan web.

Syntax PHP didasarkan pada bahasa pemrograman C, Java, dan Perl, dan dapat dengan mudah diintegrasikan ke dalam dokumen HTML. Salah satu tujuan utama bahasa PHP adalah untuk membantu pengembang web membuat situs web dinamis dengan cepat. Ini karena bahasa PHP dapat dijalankan pada *server* dan berfungsi sebagai bahasa *scripting server-side* [21].

Adapun kelebihan pengguna PHP adalah dilengkapi:

1. PHP memiliki kemampuan untuk mengelola beban kerja yang besar. Daya tampung ini menjadikan pilihan utama untuk mengelola inti dari sistem manajemen seperti konten besar, seperti Wordpress [21].
2. Kemampuan PHP untuk berfungsi di berbagai platform membuatnya sangat mudah digunakan tanpa terkendala oleh berbagai infrasturktur teknologi [21].
3. PHP memiliki kompabilitas dengan banyak *web server* yang popular, yang memungkinkan pengembang memilih dan menyesuaikan infrastruktur *server* sesuai preferensi mereka [21].
4. PHP mendukung berbagai jenis *database* tidak hanya MySQL tetapi juga *database* lainnya. Kemampuan ini memberikan kebebasan untuk pengelolaan dan penyimpanan data, memungkinkan integrasi yang fleksibel dengan berbagai sistem *database* sesuai dengan kebutuhan [21].
5. PHP adalah bahasa pemrograman gratis dengan statusnya sebagai *open source*, pengembang dapat bebas menggunakannya. Ini mendorong pertumbuhan komunitas pengguna PHP di seluruh dunia [21].

```
<!DOCTYPE html>
<html>
<body>

<?php
$nama = "Rendi Julianto";
$umur = 22;
echo "Nama: $nama";
echo "<br />";
echo "Umur: $umur tahun";

?>

</body>
</html>
```

```
Nama: Rendi Julianto
Umur: 22 tahun
```

Gambar 2.15 Contoh Sintaks PHP

2.2.10 Framework

Framework atau kerangka kerja adalah kumpulan instruksi dan instruksi yang dikumpulkan dalam *class* dan *function-function* dengan masing-masing fungsi memungkinkan pengembang untuk memanggilnya tanpa harus menulis *syntax* program yang sama berulang. Ini memungkinkan pengembang untuk menghemat waktu dan membuat program lebih cepat [22]. Ada beberapa alasan mengapa direkomendasikan menggunakan *framework*:

1. Dalam pengembangan perangkat lunak, menggunakan *framework* dapat secara signifikan menghemat waktu. Dengan menyediakan struktur dasar dan komponen yang siap dipakai, pengembang mengurangi waktu yang diperlukan untuk mengimplementasikan fitur dasar [22].
2. Pengembang dapat membuat solusi yang lebih sistematis dan mudah dipahami dengan menggunakan pedoman yang jelas dan terstruktur. Selain itu, struktur yang konsisten membantu anggota tim bekerja sama, yang membantu mereka memahami struktur kode dengan cara yang sama [22].
3. Dengan memberikan komponen dan modul yang dapat digunakan kembali, pengembang dapat mengurangi kebutuhan untuk menulis ulang kode yang serupa, meningkatkan efisiensi, dan mengurangi kemungkinan kesalahan. Penggunaan kembali kode juga meningkatkan konsistensi dan pemeliharaan aplikasi [22].
4. Pengembangan dengan menggunakan *framework* memiliki potensi untuk meningkatkan keamanan pengembangan perangkat lunak karena *framework* biasanya menyertakan fitur keamanan bawaan. Dengan adanya lapisan keamanan, pengembang dapat mengurangi risiko serangan seperti SQL *injection*, *cross-site scripting*, dan kerentanan keamanan lainnya [22].

2.2.11 Laravel

Laravel adalah kerangka kerja PHP yang dibuat oleh Taylor Otwell pada tahun 2011. Laravel, yang berasal dari kata perancis *Larave*l, yang berarti "menyediakan jalan yang lancar," adalah salah satu *framework open source* yang terbuka untuk pengembang *software* di seluruh dunia. [23].

Laravel adalah *framework* atau kerangka kerja yang dirilis dibawah lisensi MIT dan dibangun berdasarkan konsep *Model View Controller* (MVC). MVC adalah pendekatan perangkat lunak yang memisahkan aplikasi berdasarkan komponennya, seperti *controller*, manipulasi data, dan antarmuka pengguna [24]. Adapun penjelasan dari MVC sebagai berikut:

1. *Model* menunjukkan struktur data dan membantu mengelola basis data. Itu melakukan hal-hal seperti memasukkan data ke dalam basis data, melakukan pembaruan, dan tugas-tugas lain yang berkaitan dengan pengelolaan data [24].
2. *View* merupakan bagian yang mengontrol tampilan yang dilihat pengguna dalam Laravel, itu dapat berupa halaman web atau antarmuka pengguna [24].
3. *Controller* merupakan bagian yang berfungsi sebagai penghubung antara *Model* dan *View*. *Controller* bertanggung jawab untuk mengatur logika bisnis aplikasi dan mengelola alur kerja antara *Model* yang menangani data dan *View* yang menangani tampilan [24].

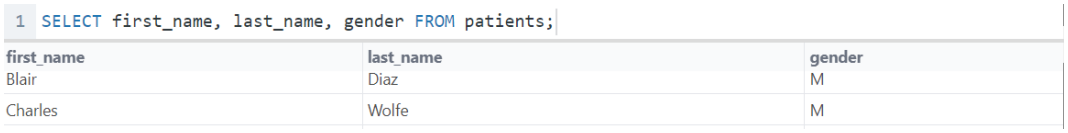
2.2.12 MySQL

MySQL adalah perangkat lunak yang tersedia secara gratis di bawah lisensi *GNU General Public License* (GPL), dan merupakan *database engine* atau *server database* yang mendukung bahasa *database* pencarian SQL. MySQL adalah perangkat lunak sistem manajemen data SQL atau DBMS yang *multi-user* dan mendukung banyak database. [25].

Berikut beberapa kelebihan dari MySQL:

1. Kemampuannya untuk beroperasi di berbagai sistem operasi, seperti Windows, Linux, FreeBSD, Mac OS X Server, Solaris, dan Amiga, memastikan bahwa MySQL dapat digunakan dalam berbagai lingkungan teknologi [25].
2. Karakteristik *open source* MySQL memungkinkan pengguna menggunakan dan mendistribusikan perangkat lunak tanpa biaya, dan komunitas penggunanya berkembang di seluruh dunia [25].

3. Beberapa pengguna dapat menggunakan MySQL secara bersamaan tanpa mengalami masalah atau konflik, yang memungkinkan mereka bekerja sama dalam pengelolaan dan manipulasi data dan mendukung penggunaan bersama dalam konteks aplikasi atau proyek yang melibatkan tim pengembang [25].
4. Kemampuan MySQL untuk memproses berbagai *query* SQL secara bersamaan memberikan kinerja yang cepat. Hal ini memungkinkan penggunaan intensif *database* dengan mengoptimalkan eksekusi *query*, memberikan *response* yang tanggap terhadap permintaan pengguna [25].
5. Jenis kolom yang tersedia oleh MySQL termasuk *integer, float, double, char, varchar, timestamp,* dan *date*. Keanekaragaman jenis data ini memungkinkan fleksibilitas dalam desain struktur *database*, memungkinkan pemodelan data yang sesuai dengan kebutuhan aplikasi [25].
6. MySQL sangat dapat diandalkan untuk aplikasi yang membutuhkan banyak data karena kapasitasnya untuk menggabungkan hingga lima puluh juta rekaman (*records*), enam puluh ribu tabel serta lima miliar baris [25].
7. MySQL menawarkan antarmuka aplikasi yang mendukung berbagai bahasa pemrograman. Dukungan untuk banyak bahasa memungkinkan pengembang mengintegrasikan dan berinteraksi dengan basis data menggunakan alat dan teknologi yang mereka pilih [25].
8. MySQL menawarkan sejumlah operator dan fungsi yang, termasuk dukungan yang luas untuk perintah *SELECT* dan *WHERE* dalam perintah *query* [25].



Gambar 2.16 Contoh Sintaks SQL

2.2.13 Application Programming Interface (API)

API merupakan singkatan dari *Application Programming Interface* yang merupakan perangkat lunak yang mengizinkan dua atau lebih aplikasi yang berbeda platform agar dapat terhubung dan berkomunikasi satu sama lain [5]. Komponen-komponen esensial dalam API:

1. *Endpoint*

   Secara sederhana, *endpoint* berfungsi sebagai pintu masuk ke API. *Endpoint* berfungsi sebagai pintu yang membawa ke area tertentu di dalamnya. Menggunakan *endpoint* sebagai alamat untuk mengarahkan permintaan ingin melakukan sesuatu dengan aplikasi lain atau meminta data dari *server* [26]. Contoh *endpoint*:

   a. /*users*: *endpoint* ini biasanya digunakan untuk mengambil daftar pengguna yang tersedia.
   b. /*users*/{id}: *endpoint* ini biasanya untuk mengambil detail dari pengguna berdasarkan id tertentu.
   c. /*users*/*create*: *endpoint* ini biasanya digunakan untuk menambah data pengguna baru.

2. *Method* HTTP

   Metode HTTP seperti *GET, POST, PUT,* dan *DELETE* mirip dengan perintah bahasa manusia untuk berkomunikasi dengan sistem. Setiap metode memiliki fungsi khusus yang menjelaskan bagaimana cara berinteraksi dengan data di API [26].

   a. Metode *GET* biasanya digunakan untuk mengambil data dari *server*.
   b. Metode *POST* digunakan untuk mengirimkan data baru ke *server*.
   c. Metode *PUT*, *PATCH* biasanya digunakan untuk memperbarui data yang sudah ada di *server*.
   d. Metode *DELETE* biasanya digunakan untuk menghapus data dari *server*.

3. Parameter

   Parameter mengirimkan informasi tambahan kepada *server* melalui *body* permintaan atau *URL*, dua komponen utama dari parameter [26].

a. Parameter dengan *URL* /*search*?*query=keyword*, dimana *query* merupakan parameter dan *keyword* adalah nilai yang akan dikirim.
b. Parameter *request body* adalah dimana tempat kita menyusun data dengan lebih terstruktur, mengemasnya dalam format JSON atau XML.

4. *Response*

   Response adalah tanggapan yang diberikan oleh *server* setelah menerima permintaan dari klien. Ini berisi data yang diminta atau pesan kesalahan jika terjadi masalah. Kode status HTTP biasanya disertakan dalam tanggapan, yang menunjukkan apakah permintaan berhasil (misalnya 200 OK) atau mengalami masalah (misalnya 404 Tidak ditemukan atau 500 Kesalahan *Server* internal) [26].

5. Format Data

   Dua format utama dalam API yaitu JSON atau *Javascript Object Notation* dan XML (*eXtensible Markup Language*). Keduanya berfungsi sebagai penyimpanan data yang akan dikirimkan atau diterima. Karena keunggulannya yang sederhana dan dapat mudah dibaca oleh mesin maupun manusia , JSON menjadi lebih sering digunakan [7] [26].

2.2.14 Javascript Object Notation (JSON)

JSON atau *Javascript Object Notation* merupakan format pertukaran data yang ringan yang mudah diterjemahkan dan dibuat oleh komputer serta mudah dibaca oleh manusia [7] [26]. JSON merupakan format yang tidak bergantung pada gaya bahasa apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C, termasuk C++, C#, Javascript, Java, Perl, Python dan sebagainya. Karena sifat-sifatnya tersebut JSON ideal digunakan sebagai bahasa pertukaran data [27]. Adapun contoh JSON sederhana bisa dilihat pada Gambar 2.17 Contoh JSON Sederhana.

```json
{
  "key": "value",
  "key2": {
    "key3": "value3",
    "key4": "value4"
  }
}
```

Gambar 2.17 Contoh JSON Sederhana

Nilai atau value pada JSON mendukung berbagai format data, yakni sebagai berikut:

1. *Number* merupakan format data yang terdiri dari bilangan bulat dan desimal.
2. *String* merupakan format data teks yang diapit oleh tanda petik saat digunakan.
3. *Boolean* merupakan format data yang terdiri dari dua nilai yaitu benar (*true*) dan salah (*false*).
4. *Array* merupakan format data yang datanya terurut yang diapit oleh [] dan dibatasi tanda koma. Setiap elemen *object* memiliki kunci dan nilai yang terbatas.
5. *Null* merupakan format data JSON yang diisi dengan *keyword* null.

Jika digabungkan seluruh format data di atas akan menjadi seperti Gambar 2.18 Contoh Gabungan Format Data JSON dibawah ini.

```json
{
  "nomor_induk": 10,
  "nama": "Jhon Doe",
  "menikah": false,
  "hobi": [
    "membaca",
    "badminton",
    "memancing"
  ],
  "alamat": {
    "jalan": "Kopo No.13",
    "kota": "Bandung"
  },
  "istri": null
}
```

Gambar 2.18 Contoh Gabungan Format Data JSON

2.2.15 Hypertext Transfer Protocol (HTTP)

HTTP adalah protokol komunikasi yang digunakan untuk mengirim dan menerima hypertext melalui internet [28]. HTTP mendefinisikan bagaimana pesan diformat dan ditransmisikan, serta tindakan apa yang harus diambil oleh web *server* dan *browser* sebagai respons terhadap berbagai perintah.

Pesan HTTP terdiri dari dua jenis: permintaan (*request*) dari klien ke *server* dan respons (*response*) dari server ke klien. Setiap pesan terdiri dari beberapa komponen utama:

1. *Request Line* (Baris Permintaan)

   Request Line (Baris Permintaan) merupakan bagian pertama dari sebuah pesan permintaan dalam protokol HTTP. Ini terdiri dari tiga elemen utama yang menentukan sifat dan tujuan permintaan: metode (*Method*), *Request-URI (Uniform Resource Identifier)*, dan Versi Protokol (HTTP-*Version*).

   Metode HTTP adalah tindakan yang ditentukan dalam baris permintaan untuk menunjukkan operasi yang akan dilakukan pada sumber daya yang ditentukan, seperti *GET*, *POST*, *PUT*, dan *DELETE*. Contoh dari *Request Line* dapat dilihat pada Tabel 2.21 Request Line HTTP.

Tabel 2.21 Request Line HTTP

```
GET http://www.example.com/index.html HTTP/1.1
```

2. *Request Headers* (Header Permintaan)

   Request Headers (Header Permintaan) menyediakan informasi tambahan tentang permintaan itu sendiri atau tentang klien yang membuat permintaan. Contoh dari Request header dapat dilihat Tabel 2.22 Request Header HTTP.

Tabel 2.22 Request Header HTTP

```
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, seperti Gecko) Chrome/58.0.3029.110
Safari/537.36
Accept: text/html
```

3. Message Body

   Isi pesan atau *Message-Body* adalah bagian dari pesan HTTP yang digunakan untuk mengirim data terkait dengan permintaan atau respons.

Data ini dapat berupa teks, JSON, atau format lainnya. Pada permintaan HTTP, isi pesan sering digunakan dalam metode seperti *GET*, *POST*, atau *PUT* untuk mengirim informasi ke server. Berikut adalah penjelasan lebih lengkap mengenai beberapa format yang umum digunakan untuk mengirim data dalam permintaan HTTP:

a. Text: Data teks dapat dikirim langsung dalam permintaan HTTP, yang dapat dilihat pada Tabel 2.23 Request Body Text HTTP.

Tabel 2.23 Request Body Text HTTP

```
POST /submit-text HTTP/1.1
Host: www.example.com
Content-Type: text/plain
Content-Length: 11

Hello world
```

b. JSON: *JavaScript Object Notation* (JSON) digunakan untuk mentransmisikan data struktur dalam format yang mudah dibaca oleh manusia dan mudah diproses oleh mesin. yang dapat dilihat pada Tabel 2.24 Request Body JSON HTTP.

Tabel 2.24 Request Body JSON HTTP

```
POST /submit-json HTTP/1.1
Host: www.example.com
Content-Type: application/json
Content-Length: 26

{
    "field1": "value1",
    "field2": "value2"
}
```

c. XML: *Extensible Markup Language* (XML) digunakan untuk menyimpan dan mengirim data terstruktur. yang dapat dilihat pada yang dapat dilihat pada Tabel 2.25 Request Body XML HTTP.

Tabel 2.25 Request Body XML HTTP

```
POST /submit-xml HTTP/1.1
Host: www.example.com
Content-Type: application/xml
Content-Length: [panjang konten]

<?xml version="1.0" encoding="UTF-8"?>
<data>
    <field1>value1</field1>
    <field2>value2</field2>
</data>
```

d. Form URL-Encoded: Digunakan untuk mengirim data formulir dalam format *application/x-www-form-urlencoded*, dimana data dikirim dalam format *key=value* yang terpisah dengan &. Contoh: yang dapat dilihat pada Tabel 2.26 Request Body URLEncoded HTTP.

Tabel 2.26 Request Body URLEncoded HTTP

```
POST /submit-form HTTP/1.1
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: [panjang konten]

field1=value1&field2=value2
```

e. Binary Data (Multipart): Digunakan untuk mengirim file atau data biner lainnya, di mana data dibagi menjadi beberapa bagian (*part*) dengan batas yang ditentukan. Contoh: yang dapat dilihat pada Tabel 2.27 Request Body Binary Data HTTP.

Tabel 2.27 Request Body Binary Data HTTP

```
POST /upload-image HTTP/1.1
Host: www.example.com
Content-Type: multipart/form-data; boundary=--------------------------
1234567890
Content-Length: [panjang konten]

----------------------------1234567890
Content-Disposition: form-data; name="file"; filename="example.png"
Content-Type: image/png

[Binary image data]
----------------------------1234567890--
```

2.2.16 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa yang digunakan untuk memvisualisasikan, menentukan, dan mendokumentasikan sistem pengembangan perangkat lunak berbasis objek. Dalam siklus pengembangan perangkat lunak, tahap desain merupakan langkah penting untuk memastikan bahwa perangkat lunak yang dibuat sesuai dengan kebutuhan pengguna. Dalam proses desain, kebutuhan fungsional dan non fungsional perlu diubah menjadi model yang dapat dimengerti oleh pihak lain. Model adalah representasi sederhana dari sistem yang sebenarnya, memungkinkan desain dapat dimengerti oleh orang lain, untuk membuat model diperlukan bahasa pemodelan yang bisa berupa kode, gambar, diagram atau deskripsi. Dalam hal ini UML berperan sebagai salah satu bahasa pemodelan [29]. Adapun diagram UML terdiri dari beberapa pemodelan berikut:

1. *Use Case Diagram*

   Use case diagram menggambarkan hubungan interaksi antara sistem dan aktor. *Use case* dapat mendeskripsikan tipe interaksi antara pengguna dengan sistem. Terdapat beberapa hal yang menjadi komponen utama dalam *Use case* yaitu:

   a. Sistem

      Sebuah sistem digambarkan ke dalam bentuk persegi.fungsinya untuk membatasi *use case* dengan interaksi dari luar sistem. Sistem pada umumnya diberikan label dari luar sistem.

   b. Aktor adalah orang yang menjelaskan siapa yang berinteraksi dengan sistem, aktor akan memberikan informasi kepada sistem, Ketika aktor memberikan informasi kepada sistem dan ketika informasi diterima oleh sistem, keduanya dapat terjadi diwaktu yang sama, meskipun terkadang aktor belum tentu orang atau manusia.

   c. *Use case* merupakan sebuah komponen yang menggambarkan fungsional dalam sebuah sistem. Yang dimana bertujuan untuk membuat lebih mudah dimengerti mengenai alur sistem yang akan dibuat.

2. *Use case* deskripsi

   Use case deskripsi adalah sebuah deskripsi atau skenario yang menjelaskan bagaimana suatu sistem digunakan. *Use case* deskripsi dibuat untuk memahami dan menggambarkan interaksi antara pengguna dan sistem.

3. *Activity Diagram*

   Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. *Activity diagram* digunakan untuk menggambarkan langkah langkah atau aktivitas pada suatu sistem.

4. *Class Diagram*

   Class diagram atau diagram kelas, dalam UML adalah jenis diagram struktur yang menggambarkan secara jelas struktur, atribut, metode dan hubungan dari setiap objek dalam suatu sistem. Ini adalah representasi statis dari sistem, yang berarti bahwa diagram kelas fokus pada hubungan yang ada di antara kelas-kelas.

5. *Sequence Diagram*

   Sequence Diagram atau diagram urutan, adalah jenis diagram dalam UML yang digunakan untuk menjelaskan dan menampilkan interaksi antara objek objek dalam sebuah sistem secara terperinci. Diagram *sequence* terdiri dari dua bagian, yaitu bagian *vertical* untuk menunjukan waktu dan urutan kejadian. Setiap objek, termasuk aktor, memiliki kolom *vertical* yang disebut dengan *lifeline* yang menunjukan waktu aktif objek tersebut. Sedangkan bagian *horizontal* menunjukan objek objek yang terlibat dalam interaksi. Pesan atau perintah antar objek objek digambarkan sebagai garis panah dari suatu lifeline ke *lifeline* lainya.

2.2.17 Object Oriented Programming (OOP)

OOP adalah singkatan dari *Object Oriented Programming*, merupakan hasil pengembangan dari keterbatasan paradigma pemrograman sebelumnya. Pada dasarnya, OOP adalah suatu metode untuk menghubungkan unit dan data yang dijalankan oleh fungsi-fungsi yang ada, di mana unit tersebut disebut objek. Metode

ini memungkinkan penyusunan kode program menjadi lebih mudah karena setiap objek dapat dibuat menjadi kelas yang berbeda tanpa perlu menulis kode lagi dan kemudian dihubungkan oleh setiap fungsi untuk ditampilkan menjadi suatu *interface* [30]. Berikut merupakan karakteristik dari OOP:

1. Objek

   Struktur penyusunan kode OOP berbeda dari penyusunan prosedural, dimana masalah diselesaikan dengan sebuah fungsi tetapi dibagi menjadi objek-objek yang masing-masing memiliki definisi [30].

2. Kelas

   Dalam OOP, objek adalah anggota kelas-kelas. Sebagai contoh, sebuah rumah dibangun berdasarkan *blueprint* kelas yang telah direncanakan dan *blueprint* tersebut merupakan sebuah kelas [30].

3. Pewarisan atau *Inheritance*

   Sebuah kelas OOP adalah induk dari beberapa sub-kelas yang memiliki beberapa persamaan dan dapat mewarisi sebuah objek atau semua objek pada dirinya sendiri atau objek yang lain [30].

4. *Reusability*

   Kelas yang telah dibuat dan memiliki definisi dapat digunakan untuk program lain. Namun, jika diperlukan modifikasi pada kelas tersebut, dapat dibuat kelas baru yang mewarisi definisi kelas induknya. Ini berkaitan dengan konsep pewarisan [30].

5. *Polimorfisme* dan *overriding*

   Polimorfisme adalah fitur penting dari OOP yang menyederhanakan baris penulisan kode program dimana fungsi yang berbeda dapat berada pada baris yang sama [30].

2.2.18 Black Box Testing

Pengujian *black box* adalah metode pengujian perangkat lunak yang bertujuan untuk mengidentifikasi masalah yang mungkin muncul saat perangkat lunak digunakan oleh pengguna [31]. Melalui pengujian ini, masalah seperti kesalahan fungsi, antarmuka pengguna, serta struktur data atau akses basis data dapat diidentifikasi. Metode ini digunakan pada penelitian ini karena memungkinkan

evaluasi dari perspektif pengguna akhir, memberikan gambaran yang jelas tentang bagaimana sistem berfungsi dalam kondisi nyata. [30].

2.2.19 Skala Likert

Skala Likert adalah alat ukur yang digunakan dalam penelitian untuk mengukur sikap, pendapat dan persepsi individual tau kelompok terdahap suatu kejadian atau fenomena [32]. Dalam penelitian ini, Skala Likert digunakan untuk menguji tanggapan pelanggan terhadap aplikasi *top up* digital. Skala ini memungkinkan peneliti untuk mengumpulkan informasi tentang persepsi pelanggan. Umumnya, skala likert menggunakan 5 tingkatan seperti pada Tabel 2.28 Tingkatan Skala Likert [33].

Tabel 2.28 Tingkatan Skala Likert

Skala	Skor
Sangat Setuju	5
Setuju	4
Ragu-Ragu	3
Tidak Setuju	2
Sangat Tidak Setuju	1

Untuk menghitung persentase dari kuisioner dapat menggunakan rumus [34] :

Persentase % = $\frac{Total\ skor\ responden}{Skor\ ideal}\ x\ 100\%$ (1)

Total skor responden = skor x frekuensi jawaban. (2)

Skor ideal = Skor maksimal x frekuensi responden. (3)