Implementasi Arsitektur REST API pada Pengembangan Sistem Transfer Dompet Digital

Disusun Oleh:

Nama : Irfan Rizq Dzaky Muhammad

NIM : 18523279

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

Implementasi Arsitektur REST API pada Pengembangan Sistem Transfer Dompet Digital

TUGAS AKHIR

Disusun Oleh:

Nama : Irfan Rizq Dzaky Muhammad
NIM : 1853279

Yogyakarta, 24 Desember 2023

Pembimbing,

(Irving Vitra Paputungan, S.T., M.Sc., Ph.D)

HALAMAN PENGESAHAN DOSEN PENGUJI

Implementasi Arsitektur REST API pada Pengembangan Sistem Transfer Dompet Digital

TUGAS AKHIR

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

Yogyakarta, 13 Februari 2024

Tim Penguji

Irving Vitra Paputungan, S.T., M.Sc., Ph.D. _____

Anggota 1

Mukhammad Andri Setiawan, S.T., M.sc., Ph.D. _____

Anggota 2

Hari Setiaji, S.Kom., M.Eng. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T, M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Irfan Rizq Dzaky Muhammad

NIM : 18523279

Tugas akhir dengan judul:

Implementasi Arsitektur REST API pada Pengembangan Sistem Transfer Dompet Digital

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

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

Yogyakarta, 24 Desember 2023

(Irfan Rizq Dzaky Muhammad)

HALAMAN PERSEMBAHAN

Bismillahirahmanirahim, alhamdulillah puji syukur kepada Allah *subhanahu wa ta'ala* atas segala pemberian kesehatan, kemudahan, dan kelancaran dalam mengerjakan tugas akhir. Shalawat serta salam kepada Nabi Muhammad *sallallahu ala muhammad sallallahu alaihi wasallam.* Laporan ini saya persembahkan kepada keluarga yang telah mendoakan, menyemangati, dan mendukung dalam pengerjaan laporan. Terimakasih karena telah memberikan dukungan terbaik yang saya butuhkan.

HALAMAN MOTO

Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya (Q.S Al Baqarah ayat 286)

“Kegagalan adalah peluang belajar”

“Kejar mimpi bukan waktu”

KATA PENGANTAR

Alhamdulillah puji syukur kepada Allah *Subhana wa ta'ala* yang telah memberikan karunia dan hidayahnya sehingga membantu dan memberi arahan dalam menyelesaikan tugas akhir. Shalawat serta salam kepada Nabi Muhammad *sallallahu ala muhammad sallallahu alaihi wasallam* sebagai rahmat bagi seluruh alam.

Penulisan skripsi ini merupakan hasil karya yang disusun untuk memenuhi salah satu syarat dalam menyelesaikan Program Studi Informatika, Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penelitian ini berjudul "Implementasi REST API pada Pengembangan Sistem Transfer Dompet Digital", merupakan sebuah upaya penulis dalam mempelajari pengetahuan pada bidang *software development*.

Penyelesaian Tugas Akhir ini telah mendapatkan dukungan, motivasi, dan bimbingan dari berbagai pihak. Oleh karena itu penulis mengucapkan terimakasih banyak kepada:

1. Allah *Subhana wa ta'ala* yang telah memberikan kesehatan dan arahan dalam pengerjaan tugas akhir dan Nabi Muhammad *sallallahu ala muhammad sallallahu alaihi wasallam* sebagai teladan dalam kehidupan sehari-hari.
2. Ayah, Ibu, dan Keluarga saya yang telah memberikan doa dan semangat selama pengerjaan penelitian berlangsung.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Irving Vitra Paputungan, S.T., M.SC., Ph.D selaku dosen pembimbing yang telah sabar membimbing, mengarahkan, dan mengajari saya dalam pengerjaan tugas akhir.
5. Teman-teman yang selalu menemani dan mendukung dalam proses penulisan tugas akhir.

Penulis menyadari bahwa penulisan tugas akhir ini tidak lepas dari keterbatasan dan kekurangan. Kritik dan saran kepada penulis sangat diapresiasi agar bertambahnya ilmu dan wawasan penulis. Akhir kata penulis berharap tugas akhir ini dapat memberikan manfaat bagi ilmu pengetahuan bidang *software development,* para pembaca, dan untuk diri penulis sendiri.

Yogyakarta, 24 Desember 2023

(Irfan Rizq Dzaky Muhammad)

SARI

Dompet digital sebagai sarana transaksi atau pembayaran secara non-tunai semakin meluas dan dapat digunakan dimanapun dalam kegiatan sehari-hari. Terdapat banyak sekali dompet digital yang digunakan di Indonesia, beberapa dompet digital yang sering digunakan adalah Dana, Ovo, Gopay, Shopeepay, dan LinkAJa. Setiap dompet digital memiliki perbedaan dalam penggunaannya. Kendala penggunaan dompet digital saat ini adalah tidak terdapatnya sebuah sistem untuk mentransfer dana dompet digital yang berbeda. Dari permasalahan tersebut, maka dikembangkan *Application Programming Interface* (API) sebagai bentuk komunikasi *client* dan *server* dengan arsitektur *Representational State Transfer* (REST). API dikembangkan dengan menggunakan bahasa pemograman Typescript dan *framework* NestJS. Sistem akan menjadi pihak ketiga dalam proses pemindahan saldo dompet digital. Hasil REST API berupa format *JavaScript Object Nation* (JSON) dengan menggunakan metode GET, POST, PUT, DELETE. Hasil penelitian ini adalah pengembangan sistem transfer dompet digital menggunakan REST API agar dapat dikembangkan oleh pengembang *frontend* atau *mobile*.

Kata kunci: REST API, dompet digital, pengambangan sistem, sistem transfer.

GLOSARIUM

API	Antarmuka agar *server* dan *client* dapat berbagi data dan berinteraksi.
Client/Frontend	Sistem atau perangkat lunak yang mengakses sumber data dan informasi dari *server*.
Endpoint	Alamat yang telah ditentukan untuk mengakses suatu sistem untuk melakukan operasi tertentu.
HTTP	Protokol komunikasi yang digunakan untuk melakukan proses pertukaran data.
JSON	Format pertukaran data atau informasi pada API.
REST	Salah satu arsitektur pada API.
Server/Backend	Sistem yang menyediakan sumber data dan informasi kepada *client* melalui jaringan.
XP	Salah satu metode pengembangan sistem dari *Agile Software Development*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi saat ini memengaruhi perekonomian Indonesia termasuk pada metode pembayaran. Peranan pembayaran secara tunai mulai bergeser ke sistem pembayaran elektronik. Definisi dompet digital menurut Bank Indonesia melalui lembaran negara Bank Indonesia nomor 18/40/PBI/201 adalah layanan elektronik untuk menyimpan data instrumen pembayaran antara lain alat pembayaran dengan menggunakan kartu dan/atau uang elektronik, yang dapat juga menampung dana, untuk melakukan pembayaran. Penggunaan istilah "kartu" dalam definisi Bank Indonesia merujuk pada *e-money*, *chip* pada *e-money* berguna dalam setiap transaksi yang dilakukan. Sementara dompet digital menggunakan aplikasi yang berbasis *server* dan membutuhkan koneksi internet (Silalahi et al., 2022). Dompet digital membuat *financial technology* berkembang yang membantu aktivitas keuangan dan transaksi sehari-hari. Di Indonesia, beberapa dompet digital yang sering digunakan antara lain Gopay, Ovo, Shopeepay, Dana, dan LinkAja.

InsightAsia, perusahaan riset pemasaran, melakukan riset yang bertajuk "*Consistency That Leads:* 2023 *E-wallet Industry Outlook*" menunjukkan dompet digital merupakan metode pembayaran yang paling banyak dipilih oleh masyarakat Indonesia (Wulandari, 2023). Hasil riset menunjukkan 71% dari total 1300 responden menggunakan dompet digital untuk berbagai macam transaksi keuangan mereka, sebanyak 49% responden memilih pembayaran tunai, 24% memilih transfer bank, 21% menggunakan *Quick Response Code Indonesia Standard* (QRIS), 18% paylater, 17% kartu debit, dan *Virtual Account* transfer sebanyak 16%.

Dompet digital tersebut memiliki perbedaan dalam penggunaannya, serta kerjasama tertentu dengan suatu pihak atau perusahaan. Contohnya, Gopay dengan *e-commerce* Tokopedia, Dana dengan *e-commerce* Lazada, Ovo dengan *e-commerce* Bukalapak, Shopeepay sebagai dompet digital *e-commerce* Shopee, dan LinkAja dengan perusahaan Kereta Api Indonesia (KAI) melalui aplikasi KAI Access.

Menurut hasil survei "tren *e-commerce* 2022 SurveySenyum", perilaku pengguna *e-commerce* menunjukkan kecenderungan memiliki tingkat loyalitas yang rendah (Fachrizal, 2022). Hasil survei berdasarkan 1000 responden menunjukkan 41% pengguna *e-commerce* - sering beralih antara satu situs *e-commerce* dengan situs *e-commerce* lainnya, 31% pengguna hanya menggunakan satu layanan *e-commerce*, dan 27% pengguna beralih situs *e-commerce* hanya beberapa kategori dan metode pembayaran. Alasan perilaku perpindahan penggunaan *e-*

commerce mengatakan 81% responden beralih karena ketersedian produk dan produk yang lebih variatif, sementara 71% mengatakan bahwa alasan peralihan karena suatu *e-commerce* menawarkan harga yang lebih terjangkau atau memberikan nilai barang yang sepadan dengan uang yang dikeluarkan. Pengguna menginginkan produk dengan harga yang bersahabat tapi tetap memiliki kualitas yang tinggi. Selain itu, survei juga menunjukkan bahwa 88% pengguna *e-commerce* lebih memilih metode pembayaran menggunakan dompet digital atau *e-wallet*.

Lembaga riset pasar Kadence Internasional melakukan survei bertajuk "*Digital Payment and Finansial Services Usage and Behavior in Indonesia*" yang dirilis pada bulan Agustus 2021 (Kadence International, 2021). Survei dilakukan secara daring oleh 1000 responden yang berasal dari Jabodetabek (40%), Surabaya (20%), Makassar (10%), Palembang (10%), Bandung (10%), dan Medan (10%). Terdiri dari 45% pria dan 55% wanita. Hasil survei menunjukkan bahwa rata-rata responden mengetahui 5-6 dompet digital, pernah mencoba 3-4 dompet digital, aktif menggunakan 2-3 dompet digital dalam satu bulan terakhir, dan 44% menyatakan menggunakan dompet digital sebanyak 4 kali dalam seminggu. Responden survei menggunakan Ovo (31%), Gopay (25%), Shopeepay (20%), Dana (19%), dan LinkAja (4%). Dompet digital tersebut banyak digunakan untuk memesan makanan, transportasi, dan belanja *online*, alasan penggunaan beberapa dompet digital dikarenakan banyak pedagang yang menerima pembayaran secara online menggunakan berbagai dompet digital.

Hasil beberapa survei tersebut menunjukkan bahwa pengguna *e-commerce* perlu menyimpan saldo atau dana mereka di berbagai platform dompet digital. Proses pengisian saldo ke dompet digital bisa dilakukan melalui layanan perbankan seperti ATM atau *mobile banking*, namun proses ini memiliki biaya administrasi yang besarannya berbeda sesuai ketentuan setiap dompet digital. Sayangnya, saat ini tidak terdapat cara untuk mentransfer saldo dari satu dompet digital ke dompet digital lainnya seperti halnya perbankan. Oleh karena itu, diperlukan suatu sistem yang memungkinkan pemindahan saldo antar dompet digital guna menghemat waktu dan mengurangi biaya administrasi yang terkait.

Penelitian ini berdasarkan pengalaman magang penulis sebagai pengembangan *backend* yang berhasil mengembangkan API pada sistem *server-side* sehingga memungkinkan sistem dikembangkan oleh pengembang *frontend* dan *mobile*. API tersebut menggunakan arsitektur REST. REST merupakan salah satu arsitektur pengembangan perangkat lunak yang bersifat terdistribusi. REST menggunakan protokol *Request-Response* pada HTTP sebagai bentuk komunikasi (Feridi, 2019). Hasil sistem dapat digunakan oleh pengguna dompet digital dengan

status pengguna *basic* tanpa perlu menaikkan status pengguna menjadi premium agar dapat mengakses sistem.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah yang dapat diambil adalah bagaimana agar para pengguna dompet digital dapat mentransfer saldo dompet digital mereka melalui sistem transfer dompet digital.

1.3 Batasan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya dan agar penelitian lebih terfokus, batasan masalah pada penelitian ini adalah sebagai berikut:

a. Fokus penelitian adalah belum terdapatnya cara untuk mentransfer saldo dompet digital yang berbeda.
b. Dompet digital adalah adalah dompet digital yang terdaftar dan telah diawasi oleh Otoritas Jasa Keuangan (OJK), seperti Gopay, Ovo, Dana, Shopeepay, dan LinkAja.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah merancang API dengan arsitektur REST pada sistem transfer dompet digital sehingga memungkinkan dikembangkan oleh pengembang *frontend* atau *mobile*.

1.5 Manfaat Penelitian

Manfaat yang diharapkan peneliti dari penelitian ini sebagai berikut:

a. Fokus penelitian adalah belum terdapatnya cara untuk mentransfer saldo dompet digital yang berbeda.
b. Dompet digital adalah adalah dompet digital yang terdaftar dan telah diawasi oleh Otoritas Jasa Keuangan (OJK), seperti Gopay, Ovo, Dana, Shopeepay, dan LinkAja.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian menjelaskan keseluruhan materi yang terdapat pada setiap bab. Sistematika penulisan terdiri dari lima bab, yaitu:

A. BAB I PENDAHULUAN

Bagian ini berisi Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan Penelitian, Manfaat Penelitian, dan Sistematika Penulisan terkait penelitian Implementasi Arsitektur REST API pada Pengembangan Sistem Transfer Dompet Digital.

B. BAB II LANDASAN TEORI

Bagian ini berisi teori penjelasan REST API, *Framework*, NestJS, Metode Pengembangan, dan Penelitian Terkait.

C. BAB III METODOLOGI PENELITIAN

Bagian ini berisi metodologi pengembangan pada penelitian menggunakan metode *Extreme Programming*.

D. BAB IV HASIL DAN PEMBAHASAN

Bagian ini berisi hasil pengembangan sistem berupa fungsionalitas dan respon API.

E. BAB I KESIMPULAN DAN SARAN

Bagian ini berisi kesimpulan dan saran dari hasil keseluruhan penelitian yang dilakukan.

BAB II
LANDASAN TEORI

2.1 REST API

Secara umum pengembangan suatu sistem memiliki dua peran utama, yaitu pengembang *frontend* dan *backend. Frontend* fokus pada aspek yang berhubungan langsung dengan pengguna, seperti antarmuka tampilan (Pham, 2020). Di sisi lain, *backend* bertanggung jawab dalam mengembangkan bagian *server*, termasuk operasi penambahan, penghapusan, dan perubahan data dalam sistem. Tugas *backend* juga mencakup manajemen basis data, pengelolaan *server*, serta keamanan sistem (Yanti & Rihyanti, 2021).

Perancangan *backend* menggunakan suatu bahasa pemograman yang dijalankan pada sisi *server* (*server-side*) (Nurhayati & Agussalim, 2023). Beberapa contoh bahasa pemograman *backend* adalah Java, PHP, Ruby, Python, dan lain sebagainya. Pengembangan *backend* menghasilkan API (Putra, 2020).

Application Programming Interface (API) merupakan antarmuka pada perancangan sistem *server* untuk berbagi data dan berinteraksi antara sisi *client* dan *server*. API dapat digunakan dalam berbagai bahasa pemograman dan dapat berjalan pada semua jenis *server* seperti Apache, NGINX, Tomcat, dan lain sebagainya. API berguna untuk meringankan beban *server* pada sistem serta meningkatkan efisiensi dalam pengembangan tanpa membuat fungsi yang serupa (Kurniawan et al., 2020). Pada pengembangan suatu sistem, API akan memanggil fungsi *Create, Read, Delete, Update* (CRUD) melalui HTTP dengan metode GET, POST, PUT, dan DELETE. Hasil dari fungsi tersebut akan dikonversi dalam bentuk JSON (Yanti & Rihyanti, 2021).

JavaScript Object Nation (JSON) merupakan format untuk melakukan penyimpanan dan pertukaran informasi data secara terstruktur. Terdapat dua elemen pada JSON, yaitu *Key,* tipe string yang diapit dengan tanda kutip, dan *Value,* objek atau informasi yang mengisi *key* seperti string, boolean, angka, dan lain sebagainya (Yanti & Rihyanti, 2021).

Jenis-jenis API terbagi berdasarkan hak akses yang diberikan (Bondel et al., 2021).

a. *Public* API atau juga disebut *Open* API merupakan API yang dapat diakses oleh semua orang pada berbagai platform. Beberapa contoh Open API seperti Google Maps API, Facebook API, dan lain sebagainya (Saputra & Tjahyanti, 2022).
b. *Partner* API merupakan API yang dikhususkan kepada pengguna atau perusahaan tertentu melalui perjanjian atau kesepakatan. *Partner* API biasanya digunakan dalam

pertukaran data dari suatu perusahaan ke perusahaan lain (Saputra & Tjahyanti, 2022).

c. *Private* API merupakan API yang digunakan untuk keperluan pribadi atau internal suatu perusahaan.

API memiliki arsitektur dalam mengembangkan sistem (Vijayakumar, 2018).

a. *Representational State Transfer* (REST)

   Representational State Transfer (REST) merupakan arsitektur API yang paling banyak digunakan. Menggunakan metode GET, POST, PUT, dan DELETE dengan hasil dalam format JSON (Saputra & Tjahyanti, 2022).

b. *Remote Procedure Call* (RPC)

   Remote Procedure Call (RPC) merupakan arsitektur API yang terdistribusi. RPC dapat digunakan oleh pengembang untuk memanggil sebuah prosedur atau fungsi dari jarak jauh atau *server* external tanpa melibatkan *server* lokal atau internal (Uma, 2023).

c. *Simple Object Access Protocol* (SOAP)

   Simple Object Access Protocol (SOAP) merupakan arsitektur API yang hanya menggunakan format *Extensible Markup Language* (XML) sehingga setiap pendefinisian parameter pada arsitektur SOAP didefinisikan kepada suatu tipe tertentu seperti Integer, String, dan lain sebagainya (Uma, 2023).

Tanaem & Iriani (2021) berdasarkan penelitiannya yang berjudul Perbandingan *Web Service* berbasis SOAP dan RESTful menyimpulkan bahwa arsitektur REST sangat ringan dan mudah dalam distribusi data (*Request* dan *Response*) dibandingkan dengan SOAP. Perbedaan ini disebabkan oleh format data yang berbeda antara REST dengan JSON dan SOAP dengan XML dalam hal kapasitas, membangun komunikasi antara *client* dan *server*, penggunaan *web cache* pada REST. REST mampu menyelesaikan permasalahan kemampuan sebuah sistem.

Susrama *et al*. (2021) berdasarkan penelitian yang berjudul "*Comparative Analysis of Rest and GrapQL Technology on Nodejs-Based API Development*". Penelitian tersebut membandingkan arsitektur REST dan GraphQL dalam pengembangan API berbasis NodeJS dengan menggunakan *framework* Express. Hasil penelitian mengatakan REST memiliki rata-rata kecepatan respon data lebih baik dalam menangani *request* API dibandingkan dengan GrapQL. GrapQL lebih unggul dalam menyajikan data kepada klien dalam mendefinisikan atribut tertentu tanpa adanya data yang tidak diperlukan. Arsitektur REST digunakan untuk aplikasi yang sederhana dan umum dalam penggunaannya (Pratama et al., 2022).

REST merupakan salah satu arsitektur dalam perancangan perangkat lunak yang bersifat terdistribusi. Pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000. REST menggunakan protokol *Request-Response* pada *Hypertext Transfer Protocol* (HTTP) sebagai bentuk komunikasi. Data/*Resource* disediakan oleh REST *server* dengan mengirim HTTP *Response* dalam format JSON atau XML kepada *client* sebagai bentuk balasan dari HTTP *Request client* (Feridi, 2019).

Beberapa komponen pada HTTP *Response* (Feridi, 2019), yaitu:

a. HTTP *method* , HTTP *method* yang digunakan dalam komunikasi data, yaitu:
b. Identifikasi lokasi resource server melalui Uniform Resource Identifier (URL).
c. Versi dari HTTP yang digunakan.
d. *Request Header,* berisi informasi dari HTTP *Request* seperti format *web client*, format body, dan lain sebagainya.
e. *Request Body,* konten data atau *resource* yang diminta *client* kepada *server.*

Sama seperti HTTP *Response*, HTTP *Request* juga memiliki beberapa komponen, yaitu:

a. *Response Code,* hasil *response* berupa kode untuk mengidentifikasi status *server* terhadap data yang diminta. Misal: 404, hasil *response* berupa data tidak ditemukan, dan 200, hasil *response* berhasil atau OK.
b. Versi dari HTTP yang digunakan.
c. *Response Header,* berisi informasi dari HTTP *Response* seperti tipe konten, panjang konten, tanggal *response*,dan lain sebagainya.
d. *Response Body,* konten data atau resource yang diberikan kepada *web client* oleh *web server.*

2.2 Framework

Framework merupakan kerangka kerja berisi kumpulan fungsi dan kelas yang siap digunakan untuk tujuan tertentu. *Framework* dapat mempermudah dan mempercepat pengerjaan proyek bagi seorang pengembang (Mualim & Putra, 2017). Manfaat penggunaan *framework* bagi seorang pengembang adalah membantu perancangan, pembuatan, pengujian dan pemeliharaan (Haniefardy et al., 2019). *Framework* menjadikan pengerjaan proyek lebih tersusun dan terstruktur dengan memanfaatkan fungsi, prosedur, dan kelas yang sudah disediakan (Pangestika & Dirgahayu, 2020). *Framework* Prado 1 merupakan *framework* yang pertama kali diperkenalkan pada tahun 2004, setahun kemudian muncul beberapa *framework* terkenal seperti Prado 2, Symfony 1, dan CakePHP 1 (Prasena & Sama, 2020).

Haryadi *et al*. (2023) dalam penelitiannya yang berjudul Perbandingan REST API Menggunakan NodeJS dan PHP pada Aplikasi Pemilihan Umum, penelitian tersebut membandingkan performa NodeJS dan PHP dalam mengembangkan REST API. Aspek yang dibandingkan adalah kecepatan respon, kemampuan sistem atau skalabilitasnya, dan kemudahan pengembangan yang diuji menggunakan perangkat lunak Postman. Hasil penelitian menyimpulkan PHP memiliki dokumentasi lebih lengkap, struktur jelas dan konsisten, pengaturan awal proyek yang lebih mudah, dan waktu respon yang meningkat dengan semakin meningkatnya ukuran data. Hasil penelitian NodeJS menyimpulkan ekosistem dan modul NodeJS lebih beragam, lebih fleksibel dikarenakan kemampuan merancang arsitektur secara bebas, kecepatan respon yang lebih stabil dan konsisten daripada PHP.

NodeJS merupakan *runtime environment* yang berbasis bahasa pemograman JavaScript dan bersifat *open-source* (Shinta, 2021). Bahasa pemograman JavaScript biasa dikenal sebagai bahasa pemograman yang berjalan disisi *client*, sementara NodeJS ada sebagai pelengkap JavaScript disisi *server* (Kurniawan et al., 2020).

NodeJS memiliki beberapa *framework* seperti ExpressJS, NestJS, KoaJS, dan lain sebagainya. Berdasarkan survei dari situs stackoverflow tahun 2023 ExpressJS dan NestJS merupakan *framework* NodeJS yang paling banyak digunakan pengembang. ExpressJS merupakan *framework* NodeJS yang minimalis dan ringan (ExpressJS Documentation, 2023). ExpressJS dalam pengembangan REST API mengimplementasikan protokol *request-response*, *routing*, dan *middleware.* Fleksibilitas ExpressJS sangat menguntungkan bagi pengembang yang berpengalaman dan dalam skenario yang rumit, fleksibilitas tersebut dapat menyebabkan *error* yang lebih tinggi dan kesalahan dalam pembuatan struktur (Ozen, 2022). NestJS hadir sebagai pengembangan ExpressJS yang lebih terstruktur. Beberapa perbandingan ExpressJS dan NestJS:

a. Struktur

NestJS memiliki struktur pola arsitektur berorientasi objek (OOP), memungkinkan memisahkan logika bisnis, *middleware*, dan *routing* dengan jelas (NetsJS Documentation, 2023). ExpressJS memberikan kebebasan dalam mengatur struktur sehingga perlu membuat struktur sendiri (ExpressJS Documentation, 2023).

b. *Middleware* dan *Pipe*

NestJS memberikan pengelolaan *middleware* dan *pipe* dengan lebih baik, sehingga memudahkan dalam penggunaan protokol *request-response* pada HTTP (NetsJS

Documentation, 2023). ExpressJS dalam pengelolaan *middleware* dan *pipe* dapat lebih manual daripada NestJS (Ozen, 2022).

c. *Dependency Injection*

NestJS memiliki *Depedency Injection* bawaan dengan menambahkan *Decorator Injectable* pada modul NestJS (NetsJS Documentation, 2023). Fungsi ini dapat lebih mudah melakukan testing dan memelihara sistem yang besar (Ozen, 2022).

d. Dokumentasi API

NestJS mendukung dokumentasi API melalui modul Swagger yang otomatis membangun *endpoint*, sehingga mempermudah dalam membangun REST API (Ozen, 2022).

Pemilihan *framework* NestJS dirasa cocok dengan Implementasi Arsitektur REST API pada Sistem Transfer Dompet Digital karena lebih terstruktur, dan lebih mudah mendokumentasikan REST API dengan modul Swagger.

2.3 NestJS

NestJS atau Nest dikembangkan oleh Kamil Myśliwiec untuk pengembangan *backend* NodeJS yang efektif dan terstruktur. NestJS mendukung Javascript dan juga Typescript dengan mengombinasikan elemen *Object Oriented Programming* (OOP), *Functional Programming* (FP), dan *Functional Reactive Programming* (FRP) (NetsJS Documentation, 2023).

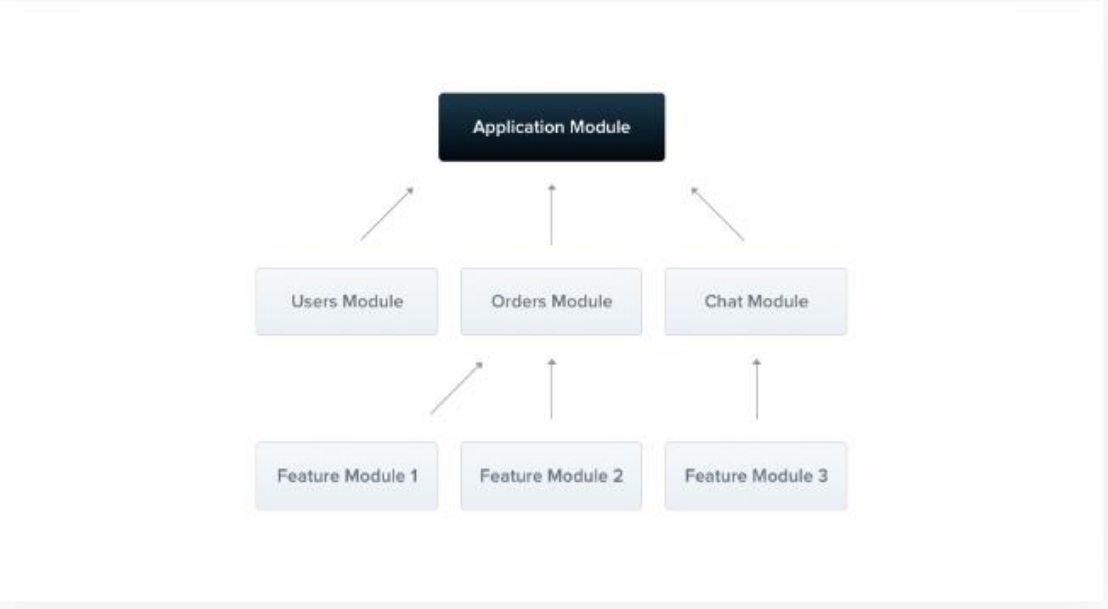
2.3.1 Struktur NestJS

Berikut merupakan beberapa struktur utama dalam arsitektur NestJS:

a. NestCLI: NestCLI (*Nest Command Line Interface*) merupakan alat yang membantu dalam pengembangan dan pemeliharaan aplikasi berbasis Nest. Membantu dalam meningkatkan proyek, pemeliharaan dalam mode pengembangan, dan membangun aplikasi pada tahap distribusi produksi (NetsJS Documentation, 2023).

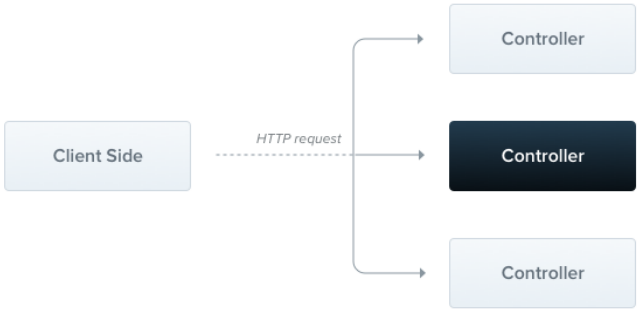
b. *Module*: *Module* (Gambar 2.1) merupakan sebuah kelas yang ditandai dengan sebuah dekorator @Module. *Module* berguna sebagai mengatur struktur aplikasi seperti *Controller*, *Service*, *Middleware*, dan lain sebagainya. *Module* pada Nest bisa memiliki lebih dari satu *module* (NetsJS Documentation, 2023).

c. *Controller*: *Controller* (Gambar 2.2) berguna mengatur HTTP *request* dan *response* yang akan dikirim ke *client*. *Controller* juga mengatur *endpoint* yang akan digunakan dan dikirim ke *server* (NetsJS Documentation, 2023).



Gambar 2.1. Nest *Module*

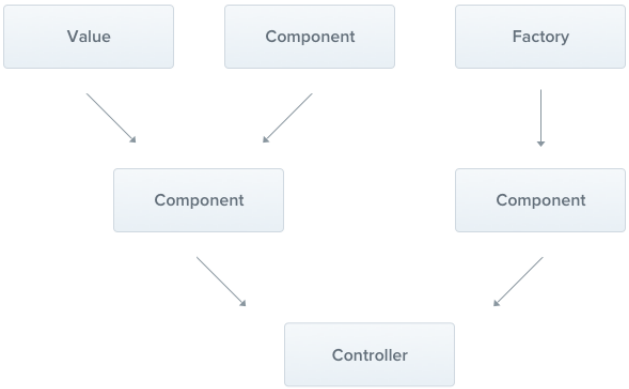
Sumber: Nest *Documentation* (2023)



Gambar 2.2. Nest *Controller*

Sumber: Nest *Documentation* (2023)

d. *Provider*/*Service*: *Provider*/*Service* (Gambar 2.3) berguna melakukan proses diluar HTTP *request*, seperti koneksi basis data, dan melakukan *request* terhadap *microservice* (NetsJS Documentation, 2023).



Gambar 2.3. Nest *Service*

Sumber: Nest *Documentation* (2023)

e. *Middleware*: *Middleware* (Gambar 2.4) merupakan sebuah fungsi yang dijalankan sebelum *route handler* (*controller*). *Middleware* mempunyai akses terhadap *request* dan *response* sehingga *middleware* dapat mengatur apakah *request* diteruskan ke *router handle* atau tidak dengan suatu pesan (NetsJS Documentation, 2023).

Gambar 2.4. Nest *Middleware*

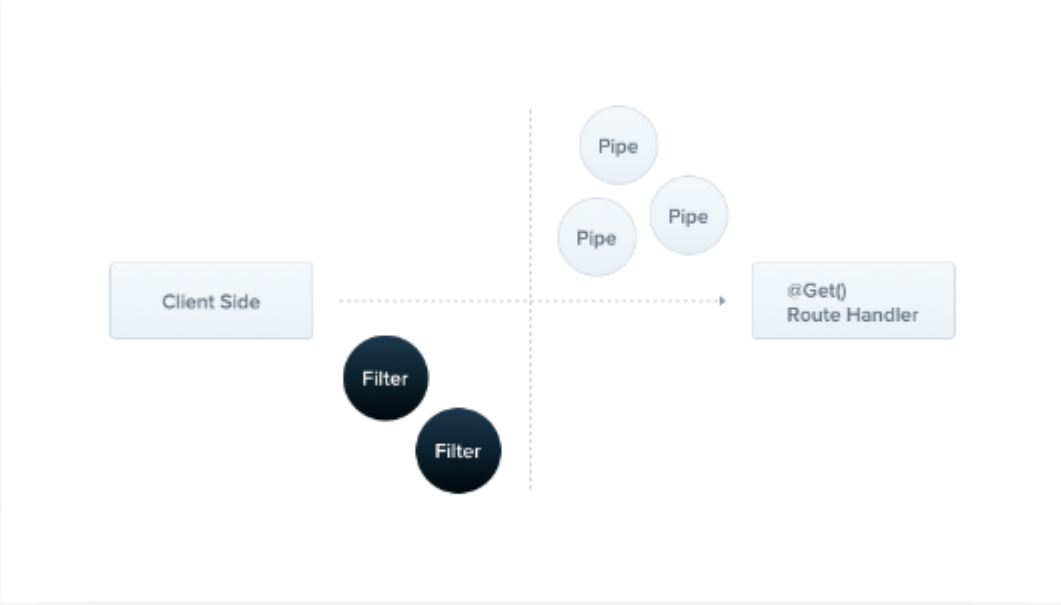
Sumber: Nest *Documentation* (2023)

2.3.2 Fitur NestJS

a. *Exception Filter*: *Exception Filter* (Gambar 2.5) merupakan fitur NestJS dalam mengelola kesalahan atau *error* dan menghasilkan suatu *response error* (NetsJS Documentation, 2023).

b. *Pipes*: *Pipes* (Gambar 2.5) merupakan fitur NestJS berupa kelas yang diberi decorator @Injectable. *Pipe* memiliki dua fungsi utama yaitu *transform*, fungsi mengubah sebuah format ke format yang diinginkan seperti mengonversikan data string ke integer, dan *validation,* fungsi memvalidasi data yang dimasukkan, jika

data benar maka *response* akan diteruskan dan jika salah maka akan melempar fungsi *exception* (NetsJS Documentation, 2023).



Gambar 2.5. Nest *Exception Filter* dan *Pipes*

Sumber: Nest *Documentation* (2023)

c. *Guard*: *Guard* (Gambar 2.6) merupakan fitur NestJs berupa kelas yang berguna dalam menentukan apakah suatu *request* yang diberikan *guard* dapat dieksekusi atau tidak oleh *route handler*, seperti fitur yang hanya bisa digunakan oleh orang tertentu, jika guard menyetujui maka *request* akan diproses, sebaliknya jika tidak maka akan ditolak (NetsJS Documentation, 2023).

Gambar 2.6. Nest *Guard*

Sumber: Nest *Documentation* (2023)

2.4 Metode Pengembangan

Pengembangan sebuah perangkat lunak membutuhkan tahapan atau metodologi agar pengembangan tersebut terstruktur dan terencana (Melinda et al., 2018). Metode pengembangan perangkat lunak berguna sebagai kerangka kerja dalam pengembangan

sehingga menghasilkan perangkat lunak sesuai yang dibutuhkan (Fatoni & Dwi, 2016). Metode pengembangan perangkat lunak pada dunia industri dikenal sebagai *Software Development Life Cycle* (SDLC), SDLC menjadi dasar metode pengembangan perangkat lunak seperti *Agile model, Prototype model, Iterative model*, dan lain sebagainya (Binuko Paksi et al., 2023).

Binuko Paksi *et al*. (2023) berdasarkan penelitian yang berjudul Perbandingan Model Pengembangan Perangkat Lunak untuk Proyek Tugas Akhir Program Vokasi. Penelitian tersebut membandingkan tiga model pengembangan perangkat lunak, yaitu *Prototype, Iterative*, dan *Agile* yang ketiga model tersebut dianggap memiliki kesamaan dalam hal fleksibilitas. Metode *Prototype* cocok digunakan pada pengembangan perangkat lunak yang dapat disesuaikan pada proses pengembangannya, metode tersebut mengharuskan memiliki gambaran awal proyek sebelum masuk ke fase selanjutnya. Metode *Iterative* cocok digunakan pada pengembangan perangkat lunak yang dapat disesuaikan pada proses pengembangannya dan berkelanjutan, metode tersebut berguna menyelesaikan proyek suatu industri tertentu yang mempunyai kemungkinan akan dikembangkan lebih lagi dalam jangka panjang berupa perangkat lunak *versioning.* Metode *Agile* cocok digunakan pada pengembangan perangkat lunak yang dapat disesuaikan pada proses pengembangannya, berkelanjutan, dan butuh waktu pengembangan yang singkat, metode tersebut berguna menyelesaikan masalah umum atau publik atau industri yang pengumpulan datanya terbatas dan butuh pengembangan lanjutan dalam waktu singkat.

Agile Software Development merupakan metode pengembangan perangkat lunak yang iteratif dan cepat, memerlukan komunikasi yang terorganisir antar tim (Mahendra & Eby Yanto, 2018). Terdapat beberapa metode pendekatan pada *Agile Software Development* seperti *Scrum* dan *Extreme Programming*. *Scrum* merupakan metode pengembangan perangkat lunak yang *iterative* dan *incremental* yang dapat diterapkan bukan hanya pada proyek tetapi juga bisa pada tingkat manajemen, *scrum* memecah komponen besar menjadi komponen-komponen kecil dalam metodenya (Abdullahi, 2022). *Extreme Programming* merupakan metode pengembangan perangkat lunak yang berfokus pada pengembangan kode pemograman atau *coding* dan dapat melakukan iterasi berulang sesuai kebutuhan pengembangan (Gumelar et al., 2018). *Extreme Programming* sangat bersifat responsif terhadap perubahan kebutuhan pengembangan, sehingga cocok digunakan pada tim berskala kecil atau menengah (Syafaat, 2018). *Extreme programming* menekankan metode pengembangan yang singkat (Trisnadoli, 2021). *Scrum* cocok digunakan oleh organisasi besar dengan metode pengembangan dengan jangka panjang, sementara *Extreme Programming* cocok digunakan oleh tim kecil atau

menengah yang membutuhkan pengembangan yang berulang dalam waktu singkat (Abdullahi, 2022). Pemilihan metode *Extreme Programming* dirasa cocok dengan penelitian karena fokus pada pengkodean atau *coding*, dan pengerjaan dengan tim kecil dalam jangka waktu yang relatif singkat.

Extreme programming merupakan salah satu metode pengembangan dalam *Agile Software Development* yang berfokus pada pengkodean atau *coding* (Gumelar et al., 2018). Metode pengembangan pada *Extreme programming* dapat dengan cepat beradaptasi pada setiap perubahan saat pengembangannya sedang berlangsung (Suryantara, 2017). Tahapan metode pengembangan *Extreme programming* (Suryantara, 2017) adalah sebagai berikut:

a. Perencanaan (*Planning*) merupakan tahap yang meliputi pemahaman fungsi dan fitur aplikasi (Suryantara, 2017).
b. Perancangan (*Design*) merupakan tahap yang meliputi proses pendefinisian komponen-komponen perangkat lunak.
c. Pengkodean (*Coding*) merupakan tahap menerjemah proses perencanaan dan perancangan kedalam bahasa yang dikenali komputer (Melinda et al., 2018).
d. Pengujian (*Testing*) merupakan tahap yang meliputi proses pengujian fungsionalitas dan respon API.

Metode pengujian dalam pengembangan perangkat lunak terbagi menjadi dua metode, yaitu *blackbox testing* dan *whitebox testing. Blackbox testing* merupakan pengujian perangkat lunak yang mengamati hasil serta fungsionalitas tanpa perlu mengetahui kode pemograman (Nurfauziah & Jamaliyah, 2022). *Whitebox testing* merupakan pengujian perangkat lunak yang mengharuskan penguji memiliki pemahaman kode pemograman dan kesesuaian terhadap spesifikasi kebutuhan perangkat lunak (Cholifah et al., 2018). Pengujian *whitebox testing* menguji logika perangkat lunak, kode pemograman, dan program secara struktural dan prosedural (Irawan, 2017).

Nurfauziah & Jamaliyah (2022) berdasarkan penelitiannya yang berjudul Perbandingan Metode Testing Antara *Blackbox* dengan *Whitebox* pada Sebuah Sistem Informasi. Penelitian tersebut membandingkan metode *blackbox testing* dan *whitebox testing*, hasil penelitian tersebut menyatakan *blackbox Testing* digunakan dalam pengujian terhadap fungsionalitas dan keluaran sistem tanpa perlu mengetahui kode pemograman, sementara *whitebox testing* digunakan dalam pengujian yang lebih kompleks, dilakukan oleh penguji yang paham *Quality Assurance* (QA) perangkat lunak, mengharuskan pemahaman struktur perangkat lunak dan kode pemograman perangkat lunak. Pemilihan metode pengujian *blackbox testing* dirasa cocok

karena cukup mengamati hasil fungsionalitas dan keluaran perangkat lunak serta keterbatasan penulis dalam ilmu QA.

2.5 Penelitian Terkait

Mulyana & Wijaya (2018) berdasarkan penelitian terkait yang berjudul Perancangan *E-Payment System* pada *E-Wallet* Menggunakan Kode QR Berbasis Android. Penelitian tersebut mengangkat permasalahan dimana pengguna dompet digital hanya bisa memindahkan saldo ke dompet digital yang sama. *E-wallet* sebagai media pembayaran dan transaksi seharusnya dapat memindahkan saldo ke dompet digital yang berbeda. Penelitian tersebut hanya dapat memindahkan saldo dompet digital dari pembeli barang/jasa ke penjual melalui teknologi *Quick Response* (QR) berbasis android, sehingga dari penelitian tersebut belum terdapatnya sistem untuk memindahkan saldo pengguna ke dompet digital yang berbeda. Perbandingan penelitian Perancangan *E-Payment System* pada *E-Wallet* Menggunakan Kode QR Berbasis Android dan penelitian ini dapat dilihat pada Tabel 2.1.

Tabel 2.1. Perbandingan Penelitian Terkait

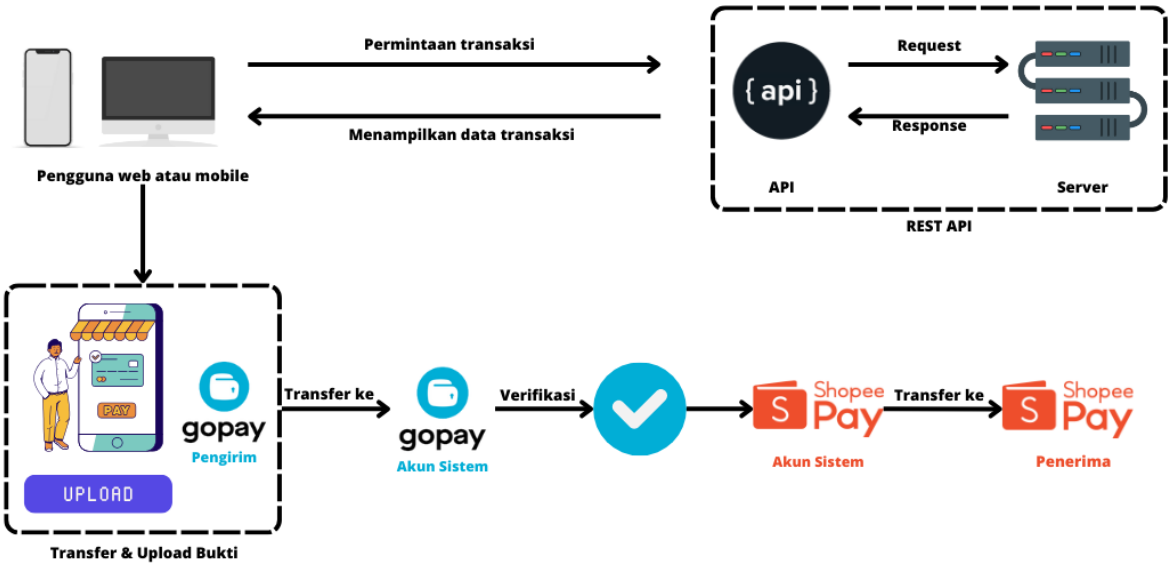
Perbandingan	***E-Payment System* kode QR**	**Sistem Transfer Dompet Digital**
Cara kerja	Menggunakan kode QR berbasis android	Memindahkan saldo dompet digital A ke B dengan menggunakan dompet digital yang sama .
Fungsi	Memindahkan saldo pembeli dan penjual, tarik dana, *top up* saldo, pembelian pulsa, dan membuat kartu RFID.	Memindahkan saldo ke dompet digital yang berbeda.
Jenis Dompet Digital	T-cash Telkomsel, BBM Money Permata Bank, DOKU.	Gopay, Dana, Shopeepay, Ovo, LinkAja.

Menurut survei Populix yang bertajuk "*Consumer Preference Toward Banking and E-Wallet Apps*" (R. Ramli & Ika, 2022). Survei mengatakan saat ini terdapat 5 dompet digital yang paling banyak digunakan oleh masyarakat Indonesia, yaitu Gopay, Dana, Shopeepay, Ovo, dan LinkAja. Dari 1000 responden dengan rentang usia 18 – 55 tahun menyatakan 88% menggunakan Gopay sebagai dompet digital yang paling diminati oleh masyarakat Indonesia, disusul 83% menggunakan Dana, Ovo 79%, Shopeepay 76%, dan LinkAja 30%. Pemilihan jenis dompet digital pada Tabel 2.1 didasarkan karena dompet digital tersebut adalah yang paling banyak diminati oleh masyarakat Indonesia saat ini.

BAB III
METODOLOGI PENELITIAN

Metode yang digunakan dalam penelitian Implementasi Arsitektur REST API pada Pengembangan Sistem Transfer Dompet Digital adalah metode *Extreme Programming*. Tahapan pada metode *Extreme Programming* adalah perencanaan (*planning*), perancangan (design), pengkodean (*coding*), pengujian (*testing*).

Ilustrasi serta gambaran REST API sistem dapat dilihat pada Gambar 3.1. Pengguna mengakses sistem melalui *website* atau *mobile* untuk meminta melakukan proses transaksi. API bekerja dengan meminta data atau informasi melalui protokol *request* kepada *server*, *server* akan memberikan *response* berupa data kepada *client* dalam format *JavaScript Object Nation* (JSON). Setelah memberikan *response* pengguna perlu mentransfer saldo ke dompet digital sistem dan akan diverifikasi oleh admin. Saldo yang telah diterima akan dikirimkan melalui dompet digital yang sama dengan dompet digital tujuan/penerima.



Gambar 3.1. Ilustrasi REST API Sistem

3.1 Perencanaan (*Planning*)

Tahapan perencanaan merupakan tahapan awal dalam metode *Extreme Programming*. Tahapan perencanaan dengan merancang kebutuhan fungsionalitas sistem yang akan dijelaskan pada Tabel 3.1. Kolom *ID* berisi identifikasi berdasarkan setiap kebutuhan sistem. Setiap kebutuhan sistem akan dibagi menjadi dalam beberapa grup kebutuhan. Kolom kebutuhan berisi aktivitas sistem dengan didahului metode *endpoint.* Kolom terakhir berisi deskripsi dari setiap kebutuhan sistem.

Tabel 3.1. Kebutuhan Fungsionalitas Sistem

ID	**Grup Kebutuhan**	**Kebutuhan**	**Deskripsi**
REQ-01.01	Autentikasi	*Post Register*	Proses untuk mendaftarkan data *user*
REQ-01.02		*Post Login*	Proses untuk mendapatkan akses sistem yang diproteksi
REQ-01.03		*Get* Konfirmasi Email	Proses untuk konfirmasi email melalui *link* yang dikirim ke email
REQ-02.01	*Users*	*Get User*	Proses untuk melihat data *user* dengan atau tidak dengan parameter *id*
REQ-02.02		*Get User* Email	Proses untuk melihat data *user* berdasarkan email
REQ-02.03		*Delete User*	Proses untuk menghapus data *user* tertentu dengan parameter *id*
REQ-02.04		*Put Update* Akun	Proses untuk mengubah data *user* dengan parameter *id*
REQ-03.01	*Password*	*Post* Ganti *Password*	Proses untuk mengubah *password* setelah *login*
REQ-03.02		*Post* Lupa *Password*	Proses mengirimkan email yang berisi *link* untuk melakukan reset *password*
REQ-03.03		*Post* Reset *Password*	Proses untuk mengubah *password* melalui cara reset
REQ-04.01	Transaksi	*Get* Transaksi	Proses untuk melihat transaksi dengan atau tidak dengan parameter *id*
REQ-04.02		*Get* Riwayat Transaksi	Proses untuk melihat riwayat transaksi yang telah dilakukan dengan atau tidak dengan parameter *id*
REQ-04.03		*Post* Transaksi	Proses untuk melakukan transaksi
REQ-04.04		*Put Upload* Bukti	Proses untuk mengirimkan bukti transaksi dengan parameter *id* transaksi
REQ-04.05		*Put* Status Transaksi	Proses untuk konfirmasi/tolak transaksi dengan parameter *id* transaksi

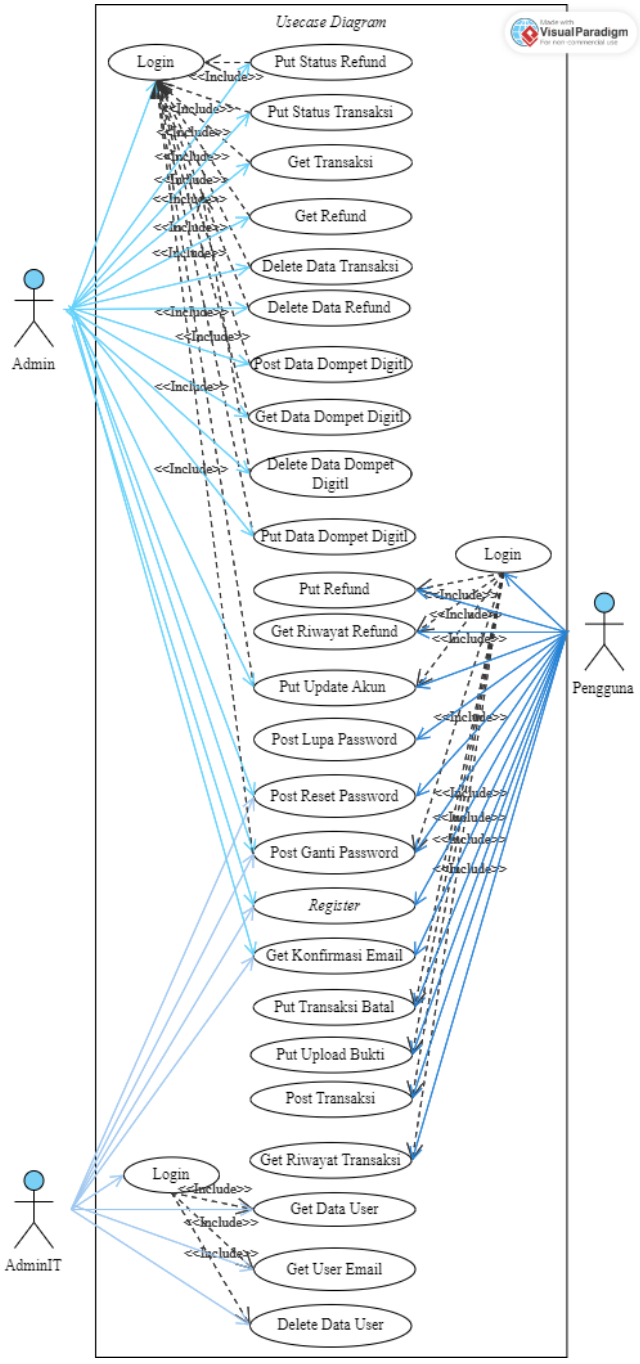
REQ-04.06		*Put* Batalkan Transaksi	Proses untuk membatalkan transaksi yang belum selesai dengan parameter *id* transaksi
REQ-04.07		*Delete* transaksi	Proses untuk menghapus data transaksi
REQ-05.01	*Refund*	*Get Refund*	Proses untuk melihat pengembalian dana dengan atau tidak dengan parameter *id*
REQ-05.02		*Get* Riwayat *Refund*	Proses untuk melihat riwayat pengembalian dana berdasarkan *user id*
REQ-05.03		*Put Refund*	Proses meminta pengembalian dana jika terjadi kesalahan pada transaksi
REQ-05.04		*Put* Status *Refund*	Proses untuk konfirmasi/tolak pengembalian dana dengan parameter *id* transaksi
REQ-05.05		*Delete Refund*	Proses untuk menghapus data pengembalian dana dengan parameter *id*
REQ-06.01	Dompet Digital	*Post* Dompet Digital	Proses untuk menambah data baru dompet digital sistem
REQ-06.02		*Get* Dompet Digital	Proses untuk melihat data dompet digital dengan atau tidak dengan parameter *id*
REQ-06.03		*Delete* Dompet Digital	Proses untuk menghapus data dompet digital dengan parameter *id*
REQ-06.04		*Put* Dompet Digital	Proses untuk mengubah data dompet digital dengan parameter *id*

3.2 Perancangan (*Design*)

Pada tahap ini dilakukan perancangan pemodelan sistem dengan menggunakan *Unified modelling language* (UML). UML merupakan teknik memodelkan sistem (Muhamad Saepuloh & Ginting, 2022). Pemodelan sistem dengan memvisualisasikan, merancang, dan mendokumentasikan sistem. Beberapa contoh dari diagram UML seperti *usecase diagram, activity diagram, sequence diagram,* dan *class diagram.* UML yang digunakan pada penelitian

ini adalah *usecase diagram*, *activity diagram*, dan rancangan basis data menggunakan *Entity Relationship Diagram* (ERD).

3.2.1 Pemodelan Sistem



Gambar 3.2. *Usecase Diagram*

Pemodelan sistem dijelaskan dengan menggunakan *usecase diagram*. *Usecase diagram* merupakan pendeskripsian suatu bentuk aktivitas pada sistem dengan menghubungkan interaksi antara aktor dan sistem (Heriyanto, 2018). Gambar 3.2 akan menjelaskan *usecase diagram* sistem.

Usecase Diagram tersebut memiliki 3 aktor yaitu pengguna, admin, dan adminIT. Ketiga aktor tersebut dapat melakukan *login*, konfirmasi email, lupa dan reset *password*. Pengguna dan admin dapat melakukan *update* akun. Pengguna secara khusus dapat melakukan aktivitas *register*, transaksi, pembatalan transaksi, *upload* bukti, proses pengembalian dana dan melihat riwayat transaksi atau pengembalian dana. Admin secara khusus dapat melakukan aktivitas tambah akun admin, lihat transaksi, konfirmasi transaksi, tolak transaksi, tambah, lihat, hapus, dan ubah data dompet digital. AdminIT secara khusus dapat melakukan lihat data dan hapus pengguna.

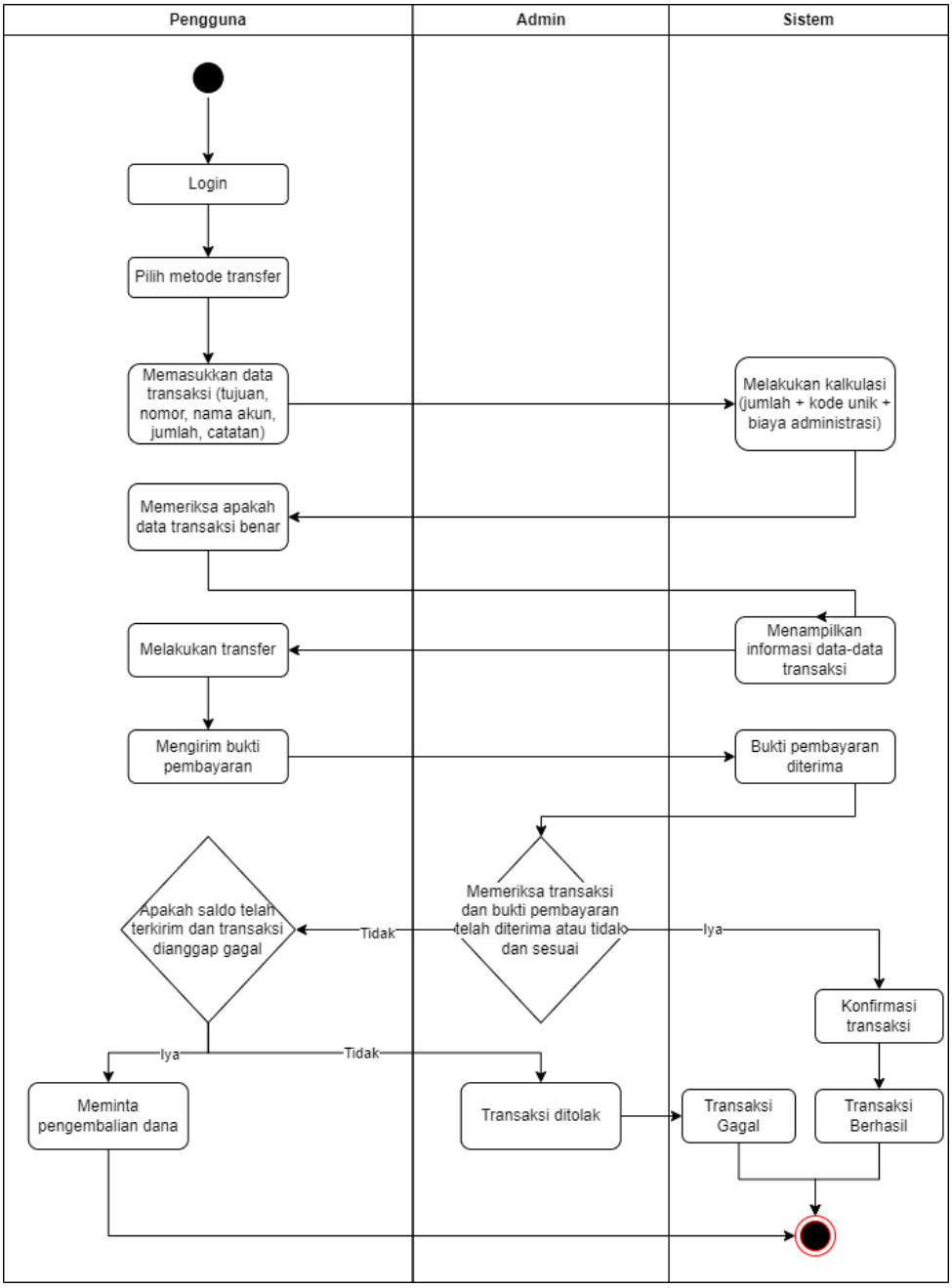
3.2.2 Alur Logika Sistem

Alur logika sistem digambarkan menggunakan *Activity Diagram* dan diambil berdasarkan *Usecase Diagram* yang telah dibuat. *Activity Diagram* merupakan salah satu pemodelan dalam UML yang memvisualisasikan aliran fungsionalitas suatu sistem. *Activity Diagram* akan mendefinisikan kapan mulai dan berhentinya suatu *workflow*, urutan serta aktivitas apa saja yang terjadi pada *workflow* tersebut (Dewi et al., 2012).

Pengguna yang akan mengakses sistem perlu melakukan proses *register* dan *login*. Informasi pengguna yang telah terdaftar dapat diganti kapan saja setelah melakukan proses *login*. Informasi yang dapat diganti adalah informasi nama, nomor hp, email dan *password*. Email yang diganti haruslah email yang belum terdaftar pada sistem. Mengganti *password* membutuhkan data atau informasi *password* baru dan konfirmasi *password* harus sama. Pengguna yang lupa dengan *password* dapat melakukan proses lupa dan reset *password.* Proses lupa *password* akan mengirimkan *link* untuk melakukan proses reset *password.* Reset *password* membutuhkan data *password* baru dan konfirmasi *password* harus sama. Admin dapat mengelola informasi dompet digital yang akan digunakan sebagai data pembayaran sistem. Admin dapat menambah, mengubah, dan menghapus data dompet digital yang sudah terdaftar.

a. *Activity Diagram* Transaksi

   Pengguna yang akan melakukan transfer antar dompet digital dapat melakukan aktivitas transaksi yang dapat dilihat pada Gambar 3.3. Pengguna memasukkan data transaksi yang berisi dompet digital tujuan, nomor tujuan, nama akun tujuan, jumlah, dan catatan transaksi. Pengguna dapat memilih pembayaran melalui dompet digital Gopay, Ovo, Shopeepay, LinkAja, dan Dana. Admin mengecek apakah transaksi berhasil diterima atau gagal, jika transaksi berhasil maka akan mengirim status transaksi berhasil, dan jika transaksi gagal maka akan mengirim status transaksi gagal.

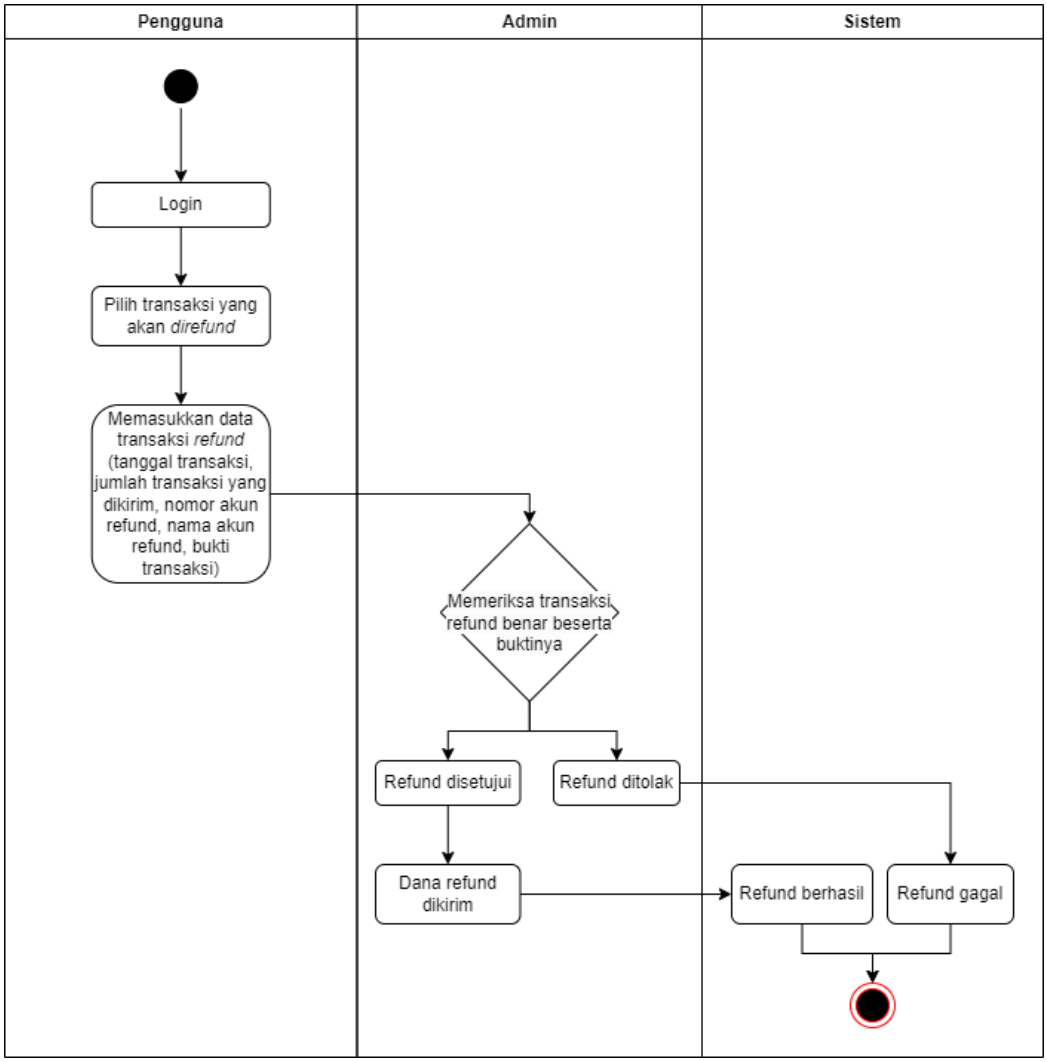


Gambar 3.3. *Activity Diagram* Transaksi

b. *Activity Diagram* Pengembalian Dana

Pengguna yang gagal melakukan transaksi tetapi telah mengirim dana/saldo atau kesalahan transaksi lainnya seperti terkirim dua kali dapat melakukan proses pengembalian dana. Alur proses pengembalian dana dapat dilihat pada Gambar 3.4. Pengguna memilih transaksi yang akan dikembalikan dananya, memasukkan data tanggal transaksi, jumlah transaksi yang telah terkirim sesuai bukti transfer, nomor akun dompet digital yang akan dikirim dana pengembalian dana, nama akun dompet digital yang akan dikirim dana pengembalian dana, alasan meminta pengembalian

dana, dan bukti transaksi. Admin akan memeriksa apakah butuh pengembalian dana atau tidak, jika disetujui maka dana akan dikembalikan sesuai bukti yang terkirim, dan jika tidak maka pengembalian dana dianggap gagal. Pengembalian dana yang dianggap gagal atau tertolak dapat mengajukan kembali proses pengembalian dana dengan menyertakan bukti baru dan informasi tambahan yang sesuai dengan keadaan transaksi.



Gambar 3.4. *Activity Diagram* Pengembalian Dana

3.2.3 Desain Basis Data

Penelitian ini menggunakan basis data MySQL. MySQL merupakan *DataBase Management System* (DBMS) yang berfungsi sebagai pengolah basis data dengan bahasa SQL (Pangestika & Dirgahayu, 2020). MySQL merupakan basis data yang kuat dan stabil sebagai media penyimpanan (Purwanto & Kom, 2012). Data pada MySQL disimpan dalam objek yang disebut tabel. Sebuah tabel terdiri dari hubungan berbagai entri dan disajikan dalam bentuk

kolom dan baris (Muhamad Saepuloh & Ginting, 2022). Basis data MySQL akan digambarkan menggunakan *Entity Relationship Diagram* (ERD).

ERD merupakan pemodelan basis data relasional. ERD berguna sebagai gambaran bagaimana basis data yang akan dibuat bekerja. ERD dibuat berdasarkan kebutuhan fungsionalitas sistem dan dapat dilihat pada Gambar 3.5. ERD dapat membantu dalam perancangan dan analisa sistem karena dapat menunjukkan komponen data yang dibutuhkan dan hubungan data didalamnya (Amijaya et al., 2019).

Gambar 3.5. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) terdiri dari tabel beserta rinciannya. Tabel transaksi (Tabel 3.2), *refund* (Tabel 3.3), *refreshToken* (Tabel 3.4), *Ewallet* (Tabel 3.5), pengguna (Tabel 3.6), admin (Tabel 3.7), adminIT (Tabel 3.8), *user* (Tabel 3.9). Tabel pengguna, admin, dan adminIT dipisah dengan tabel user berguna sebagai *Role-Based Access Control* (RBAC).

RBAC merupakan mekanisme kontrol akses berdasarkan hak dan peran pengguna (NetsJS Documentation, 2023).

Tabel 3.2. Transaksi

Atribut	***Constraint***	**Keterangan**
Id	*Primary Key*	UUID
Tujuan		Tujuan transaksi dompet digital
Nomor_tujuan		Nomor transaksi yang akan ditransfer
Nama_akun		Nama atau username akun yang akan ditransfer
Jumlah		Jumlah yang akan ditransfer
Catatan		Catatan keperluan transfer
Metode_bayar		Memilih metode bayar
Kode_unik		Kode unik pembayaran
Total		Total Jumlah + Kode Unik
Bukti		Bukti berupa gambar
Status		Status: Berhasil/Gagal/Menunggu/Batal
Create_at		Tanggal transaksi dibuat
User_id	*Foreign Key*	

Tabel 3.3. *Refund*

Atribut	***Constraint***	**Keterangan**
Id	*Primary Key*	UUID
Tanggal_transaksi		Tanggal transaksi
Jumlah_transaksi		Jumlah transaksi sesuai bukti transfer
Jenis_wallet		Jenis wallet yang akan dikirim pengembalian dana (Dana/Ovo/Shopeepay/LinkAja/Gopay)
Alasan_refund		Alasan meminta proses pengembalian dana
Nomor_wallet		Nomor dompet digital yang akan dikirim pengembalian dana
Nama_wallet		Nama dompet digital yang akan dikirim pengembalian dana
Bukti_refund		Bukti transaksi
Status_refund		Status: Diajukan/Disetujui/Ditolak/Dana dikirim
Create_at		Kode unik pembayaran
Transaksi_id	*Foreign Key*	
User_id	*Foreign Key*	

Tabel 3.4. *refreshToken*

Atribut	*Constraint*	Keterangan
Id	*Primary Key*	UUID
isRevoked		Boolean (*revoked* atau *not revoked*)
expiredAt		Tanggal token kadaluarsa
User_id	*Foreign Key*	

Tabel 3.5. *Ewallet*

Atribut	*Constraint*	Keterangan
Id	*Primary Key*	UUID
Jenis		Jenis: Dana/Ovo/Shopeepay/Gopay/LinkAJa
Nama		Nama akun dompet digital yang dipakai untuk pembayaran
Nomor		Nomor akun dompet digital yang dipakai untuk pembayaran
Create_at		Tanggal dibuat
Update_at		Tanggal data diubah
User_id	*Foreign Key*	

Tabel 3.6. Pengguna

Atribut	*Constraint*	Keterangan
Id	*Primary Key*	UUID
Nomor_hp		Nomor hp sebagai data tambahan
Created_at		Tanggal dibuat
Update_at		Tanggal data diubah
User_id	*Foreign Key*	

Tabel 3.7. Admin

Atribut	*Constraint*	Keterangan
Id	*Primary Key*	UUID
Posisi		Posisi atau jabatan dari admin
Created_at		Tanggal dibuat
Update_at		Tanggal data diubah
User_id	*Foreign Key*	

Tabel 3.8. AdminIT

Atribut	*Constraint*	Keterangan
Id	*Primary Key*	UUID
Created_at		Tanggal dibuat
Update_at		Tanggal data diubah
User_id	*Foreign Key*	

Tabel 3.9. *User*

Atribut	*Constraint*	Keterangan
Id	*Primary Key*	UUID
nama_lengkap		Nama lengkap pengguna akun
email		Email pengguna
password		*Password* pengguna
salt		Enkripsi *password*
role		Pengguna/admin/adminIT
emailVerified		*Verified* atau tidak
created_at		Tanggal dibuat
Update_at		Tanggal data diubah

3.3 Pengkodean (*Coding*)

3.3.1 Konfigurasi

Tahap pengkodean (*coding)* dimulai dengan mengkonfigurasi proyek menggunakan NestCLI. Prasyarat menggunakan *framework* NestJS adalah minimal versi 16 NodeJS. Versi NodeJS yang digunakan adalah versi 18.17.1 dan NPM 9.6.7 dengan menggunakan kode editor Visual Studio Code. Mulai membuat proyek dengan memasukkan perintah pada Gambar 3.6 pada terminal sistem operasi. Perintah *project-name* pada Gambar 3.6 diganti sesuai dengan nama proyek yang akan dibuat.

```
$ npm i -g @nestjs/cli
$ nest new project-name
```

Gambar 3.6. Konfigurasi Projek

Tahap dilanjutkan dengan menginstal beberapa fitur. Fitur-fitur yang diinstal pada proyek adalah sebagai berikut:

a. TypeORM

ORM merupakan teknik memetakan basisdata kesebuah objek, pemetaan dilakukan kepada tabel-tabel basisdata dengan suatu kelas pada bahasa pemograman yang berorientasi objek (Wijayanto, 2011). Salah satu ORM yang disediakan oleh TypeScript adalah TypeORM. TypeORM berguna untuk integrasi basis data baik basis data relasional, seperti PostgreSQL, Oracle, Microsoft SQL *Server*, dan SQLite, atau basis data NoSQL seperti MongoDB (NetsJS Documentation, 2023). Masukkan perintah pada Gambar 3.7 untuk menginstal TypeORM. Setelah penginstalan perlu untuk mengkonfigurasikan basis data pada modul utama sistem dan dapat dilihat pada Gambar 3.8.

```
$ npm install --save @nestjs/typeorm typeorm mysql2
```

Gambar 3.7. *Instal* TypeORM

```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'root',
      password: 'root',
      database: 'test',
      entities: [],
      synchronize: true,
    }),
  ],
})
export class AppModule {}
```

Gambar 3.8. Konfigurasi Basis Data

b. *ValidationPipe*

ValidationPipe merupakan *library* yang berguna untuk memvalidasi data yang dimasukkan, jika data yang dimasukkan benar maka akan mengirimkan respon dan jika tidak maka akan ditolak dan melempar fungsi *exception*. Aturan-aturan spesifik yang ditentukan *ValidationPipe* dideklarasi secara sederhana dalam kelas *Data*

Transfer Object (DTO) (NetsJS Documentation, 2023). Masukkan perintah pada Gambar 3.9 untuk menginstal *ValidationPipe.*

```
$ npm i --save class-validator class-transformer
```

Gambar 3.9. *Instal ValidationPipe*

c. *File Upload*

Pengembangan sistem membutuhkan untuk pengguna meng-*upload* bukti pembayaran dan proses pengembalian dana, maka perlu menginstal fitur *file upload* dari NestJS yaitu multer. Multer menangani *file upload* melalui metode POST pada HTTP. Masukkan perintah pada Gambar 3.10 untuk menginstal multer atau *file upload. Library* multer dapat menangani *multi file upload.*

```
$ npm i -D @types/multer
```

Gambar 3.10. *Instal* Multer

d. Passport (Autentikasi)

Autentikasi merupakan proses verifikasi identitas user dalam mengakses sumber daya suatu sistem (Arief, 2010). Passport merupakan library autentikasi NodeJS paling populer. Masukkan perintah pada Gambar 3.11 untuk menginstal Passport. Autentikasi proyek menggunakan *JSON Web Token* (JWT). JWT merupakan sebuah token dalam bentuk string yang terdiri dari tiga bagian, yaitu: *header*, *payload*, dan *signature* yang berguna sebagai bentuk autentikasi dan pertukaran informasi. Pengguna yang berhasil melakukan *login* akan menerima *token* JWT dan akan disimpan pada *local storage* atau *cookies browser* (Gunawan & Rahmatulloh, 2019). Masukkan perintah pada Gambar 3.12 untuk menginstal JWT. Gambar 3.13 merupakan contoh JWT.

```
$ npm install --save @nestjs/passport passport passport-local
```

Gambar 3.11. *Instal* Passport

```
$ npm install --save @nestjs/jwt passport-jwt
```

Gambar 3.12. *Instal* JSON Web Token (JWT)


```
"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJzdWIiOiJiZmI1OTY1Yi0xMTMwLTQ4OGEtYTRjNC00YjdjYWJmYTQyM2QiLCJpYXQiOjE3MDYzMjk1MzksImV4cCI6MTcwOTkyOTUzOX0.
    zZ6uR99GALWbeDoLMeTg-MIj1JRdrfcmYokSCs-5ZMo",
```


Gambar 3.13. Contoh JSON Web Token (JWT)

e. *Universally Unique Identifier* (UUID)

Universally Unique Identifier (UUID) atau kadang juga disebut *Globally Unique Identifier* (GUID) merupakan sejenis data dalam bentuk integer dengan panjang 128-bit berguna sebagai identifikasi unik suatu data atau objek pada sistem (Gasparik, 2023). Sistem ini menggunakan UUID versi 4 yang bersifat acak sehingga memperkecil kemungkinan nomor identifikasi yang sama. Masukkan perintah pada Gambar 3.14 untuk menginstal UUID.

```
npm install uuid
```

Gambar 3.14. *Instal* UUID

f. *Hashing*

Hashing berguna sebagai pengaman informasi data seperti *password*. Fungsi *hash* menerima tipe data string dengan panjang acak dan mengkonversinya sebagai string dengan panjang keluaran yang tetap (Sakti et al., 2016). Masukkan perintah pada Gambar 3.15 untuk menginstal fungsi hash pada *library* bcrypt.

```
$ npm i bcrypt
$ npm i -D @types/bcrypt
```

Gambar 3.15. *Instal* Bcrypt

g. *Crypto*

Library crypto pada pengembangan digunakan sebagai perhitungan untuk menghasil kode unik yang akan otomatis dibuat berbeda setiap transaksi dan setiap hari. Masukkan perintah pada Gambar 3.16 untuk menginstal *library crypto.*

```
npm install crypto dayjs
```

Gambar 3.16. Instal Crypto

h. Nodemailer

Nodemailer merupakan *library* untuk mengirim email pada NestJS. Nodemailer dapat mengirim email melalui protokol *Simple Mail Transfer Protocol* (SMTP). SMTP merupakan protokol yang berfungsi untuk mentransfer surat elektronik atau email (Arifin, 2017). Masukkan perintah pada Gambar 3.17 untuk menginstal Nodemailer.

```
npm install nodemailer
```

Gambar 3.17. Instal Nodemailer

i. OpenAPI

Pendokumentasian REST API pada sistem akan menggunakan *tools* Swagger. Masukkan perintah pada Gambar 3.18 untuk menginstal Swagger. Setelah penginstalan perlu untuk melakukan konfigurasi Swagger pada file main.ts. Konfigurasi Swagger dapat dilihat pada Gambar 3.19.

```
$ npm install --save @nestjs/swagger
```

Gambar 3.18. Instal Swagger

```
async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  const config = new DocumentBuilder()
    .setTitle('Cats example')
    .setDescription('The cats API description')
    .setVersion('1.0')
    .addTag('cats')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api', app, document);

  await app.listen(3000);
}
bootstrap();
```

Gambar 3.19. Konfigurasi Swagger

j. Menjalankan Aplikasi

Setelah proses pengkodean (*coding*) selesai maka perlu menjalankan aplikasi. Masukkan perintah pada Gambar 3.20 untuk menjalankan aplikasi.

```
$ npm run start
```

Gambar 3.20. Menjalankan Aplikasi

3.3.2 Modul

Tahapan dilanjutkan dengan membagi proyek menjadi beberapa modul, yaitu *user* modul, *authentication* modul, transaksi modul, *refund* modul, *wallet* modul, dan modul lainnya. Setiap modul memiliki guard *Role Guard* (RBAC) dan *JSON Web Token* (JWT).

a. *User* modul

   User modul membagi beberapa fitur menampilkan data pengguna, register, *update* akun, mengganti email, menghapus akun, konfirmasi email, mengganti *password*, lupa dan reset *password*.

b. *Authentication* modul

   Authentication modul merupakan modul fitur sistem *login*, *logout*, dan *request* akses token baru.

c. Transaksi modul

   Transaksi modul membagi beberapa fitur sistem transaksi seperti menampilkan data transaksi, melakukan transaksi, *upload* bukti, menghapus transaksi, mengubah status transaksi.

d. *Refund* modul

   Refund modul membagi beberapa fitur sistem pengembalian dana seperti menampilkan data *refund*, mengajukan *refund*, mengganti status *refund*, dan menghapus *refund.*

e. *Wallet* modul

   Wallet modul membagi beberapa fitur seperti membuat, membaca, mengganti, dan menghapus data dompet digital.

f. Modul lainnya

   Modul ini berisi tambahan fitur sistem seperti konfigurasi JWT, email *service*, *universally unique identifier* (UUID). Email *service* berisi konfigurasi email, pada tahap pengembangan sistem menguji apakah email berhasil atau tidak menggunakan pihak ketiga email *testing* MailTrap. Email *service* juga dapat digunakan untuk pengiriman dalam bentuk gmail.

Tahapan pengkodean (*coding*) dilanjutkan dengan men-*generate* kelas *Entity*, *Data Transfer Object* (DTO), dan *Repository* untuk setiap modul. *Entity* merupakan suatu kelas yang berguna sebagai representasi sumber data ke basis data pada NestJS (NetsJS Documentation, 2023). *Data Transfer Object* (DTO) merupakan suatu kelas yang berguna untuk mendefinisikan sumber data agar bisa dikirim ke *entity* atau basis data (NetsJS Documentation,

2023). *Repository* merupakan suatu kelas yang berguna untuk merangkum atau memanipulasi sumber data berdasarkan kebutuhan dalam pengkodean (*coding*) (Benita, 2022).

Dotenv merupakan file yang berisi variabel kebutuhan berdasarkan tempat dimana kode dapat diubah sesuai kebutuhan pengembangan. Kode dalam file dotenv tidak dapat di-*publish* karena berisi informasi penting seperti informasi layanan gmail dan *password*, konfigurasi basis data, dan informasi penting lainnya.

3.4 Pengujian (*Testing*)

Tahap pengujian menggunakan metode pengujian *blackbox testing*. REST API yang diuji dapat dilihat pada Tabel 3.10. Kolom *ID* pada Tabel 3.10 mengacu pada kolom *ID* Tabel 3.1 dengan menyesuaikan kebutuhan pada perencanaan sistem. Semua API berhasil diuji dengan beberapa kondisi serta respon API sesuai keinginan.

Tabel 3.10. Tabel Pengujian API

ID	**API**			**Hasil**
	Metode	**Fungsi**	***Endpoint***	
REQ-01.01	POST	*Register*	/users/register/(pengguna/admin/adminIT)	Berhasil
REQ-01.02	POST	*Login*	/auth/login/(pengguna/admin/adminIT)	Berhasil
REQ-01.03	GET	Konfirmasi Email	/users/confirm/:id	Berhasil
REQ-02.01	GET	Data *User*	/users atau /users/:id	Berhasil
REQ-02.02	GET	*User* Email	/users/getEmail/:email	Berhasil
REQ-02.03	DELETE	*User*	/users/:id	Berhasil
REQ-02.04	PUT	*Update* Akun	/users/(updatePengguna/updateAdmin)/:id	Berhasil
REQ-03.01	PUT	Ganti *Password*	/users/:id/gantiPassword	Berhasil
REQ-03.02	POST	Lupa *Password*	/users/forgetPassword	Berhasil
REQ-03.03	POST	Reset *Password*	/users/resetPassword	Berhasil
REQ-04.01	GET	Data transaksi	/transaksi atau /transaksi/:id	Berhasil

REQ-04.02	GET	Riwayat transaksi	/transaksi/riwayat/:id	Berhasil
REQ-04.03	POST	Transaksi	/transaksi/(bayarDana/ bayarOvo/ bayarShopeepay/bayarGopay/ bayarLinkAja)	Berhasil
REQ-04.04	PUT	*Upload* bukti	/transaksi/uploadBukti/:id	Berhasil
REQ-04.05	PUT	Status transaksi	/transaksi/(statusBerhasil/statusGagal)/:id	Berhasil
REQ-04.06	PUT	Batalkan Transaksi	/transaksi/statusBatal/:id	Berhasil
REQ-04.07	DELETE	Transaksi	/transaksi/:id	Berhasil
REQ-05.01	GET	Data *refund*	/refund atau /refund/:id	Berhasil
REQ-05.02	GET	Riwayat *refund*	/refund/riwayat/:id	Berhasil
REQ-05.03	PUT	*Refund*	/transaksi/refund/:id	Berhasil
REQ-05.04	PUT	Status *refund*	/refund/(refundSetuju/refundBerhasil/refund Tolak)/:id	Berhasil
REQ-05.05	DELETE	*Refund*	/refund/:id	Berhasil
REQ-06.01	POST	Dompet digital	/(dana/ovo/linkAja/gopay/shopeepay)/tambah	Berhasil
REQ-06.02	GET	Dompet digital	/(dana/ovo/gopay/shopeepay/linkAja) atau /(dana/ovo/gopay/shopeepay/linkAja)/:id	Berhasil
REQ-06.03	PUT	Dompet digital	/(dana/ovo/gopay/shopeepay/linkAja)/update/ :id	Berhasil
REQ-06.04	DELETE	Dompet digital	/(dana/ovo/gopay/shopeepay/linkAja)/:id	Berhasil

Beberapa API diuji menggunakan kondisi tertentu. Kondisi tersebut berguna untuk menguji apakah semua fungsionalitas sistem sesuai yang diharapkan dengan menggunakan beberapa aplikasi pihak ketiga atau *tools* pengujian. Beberapa kondisi yang ditentukan dapat dilihat pada Tabel 3.11.

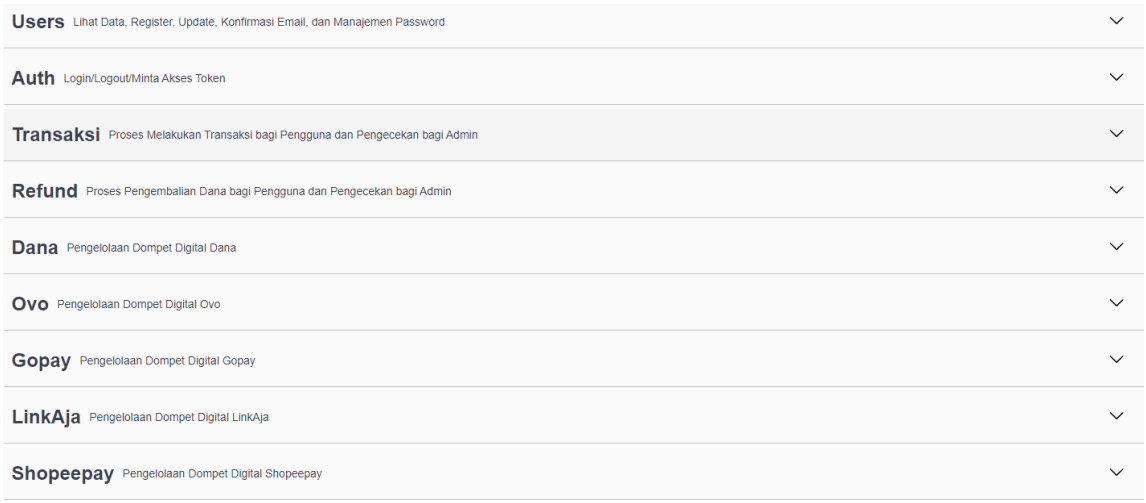
Tabel 3.11. Kondisi API

NO	Kondisi	*Tools*	Hasil
1	Apakah email berhasil dikirim beserta fungsinya seperti *link* konfirmasi dan reset *password*?	Gmail atau Mailtrap	Berhasil
2	Apakah *Role-base Access Control* (RBAC) berhasil diimplementasi dan membatasi hak akses sistem?	Postman	Berhasil
3	Apakah *error handling* berhasil menangani kesalahan *input* data?	Postman	Berhasil
4	Apakah kode unik pada transaksi berhasil di *generate* pada fungsi transaksi?	Postman	Berhasil

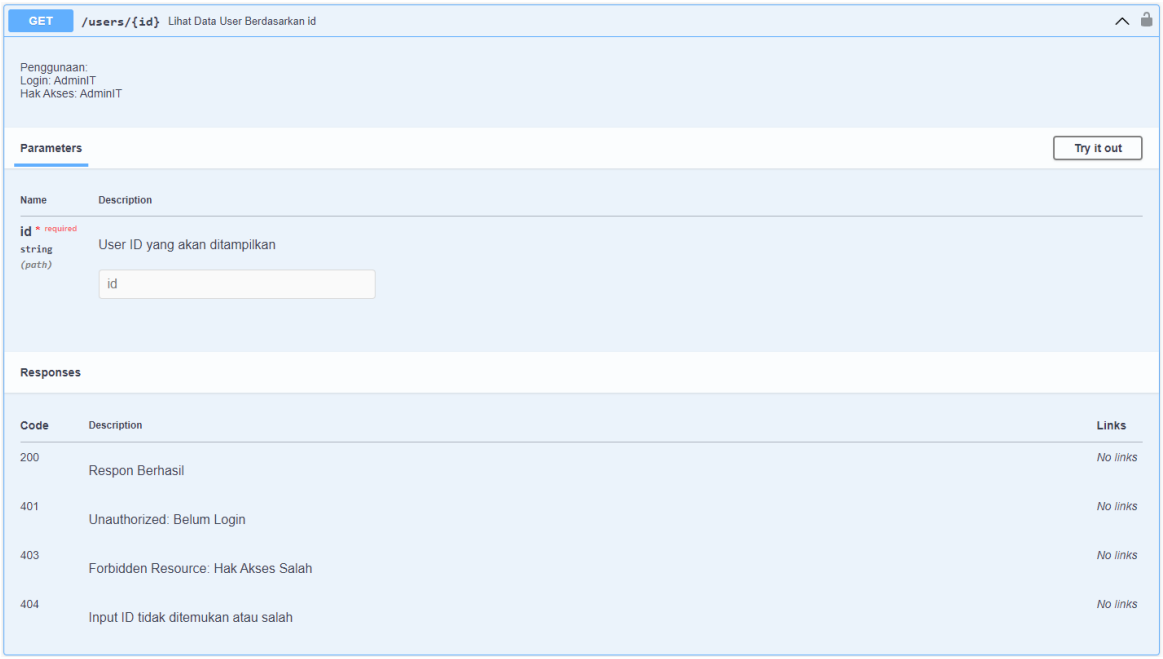
Katalog API merupakan lokasi dimana untuk mendeklarasikan bagaimana API akan bekerja, termasuk *endpoint*, tindakan pada setiap *endpoint*, respon API, dan detail API lainnya. Format *file* yang umum digunakan dalam membuat kontrak dan dokumentasi API adalah spesifikasi *Open API* atau sebelumnya dikenal sebagai Swagger, Katalog API ditujukan kepada konsumen API baik perusahaan atau pengembang yang akan menggunakan API (Hasanuddin et al., 2022).

Open API atau Swagger merupakan *tools* dalam mendokumentasi dan mengakses RESTful API berdasarkan spesifikasi dari *Open API*. Bahasa dan format *file* pada *Open API* adalah Yet Another Markup Language (YAML) (Darmawansyah & Mohd., 2018). Spesifikasi *Open API* merupakan standar yang memungkinkan manusia dan komputer untuk menemukan dan memahami kemampuan suatu layanan API tanpa perlu mengakses sumber kode (Swagger, 2023).

Implementasi *Open API* pada NestJS dapat dikonfigurasi pada file main.ts. Klasifikasi *tag* pada *Open API* untuk pendokumentasian disesuaikan dengan setiap *controller* pada seluruh modul dan dapat dilihat pada Gambar 3.21. Setiap *endpoint* berisi deskripsi penggunaan API, hak akses API, apakah diperlukan *login* atau tidak, deskripsi parameter dan *body* untuk setiap kolom, dan *list* respon beserta kodenya yang dapat dilihat pada Gambar 3.22. *List* kode beserta deskripsi kodenya akan dijelaskan pada Tabel 3.12.



Gambar 3.21. Klasifikasi *tag* API



Gambar 3.22. *Endpoint* dan deskripsinya

Tabel 3.12. Kode Respon API

NO	Kode	Deskripsi
1	200 atau 201	Respon API berhasil dan menampilkan *return*
2	400	*Bad Request*: Data yang dimasukkan salah, tidak valid, atau tidak terisi
3	401	*Unauthorized* atau belum melakukan *login*
4	403	*Forbidden Resource* atau hak akses salah
5	404	Masukkan *identifier* salah atau tidak ditemukan

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil REST API

Penelitian ini menghasilkan API dengan arsitektur REST sistem transfer dompet digital dengan format penulisan *JavaScript Object Notation* (JSON). *JavaScript Object Nation* (JSON) merupakan format untuk melakukan penyimpanan dan pertukaran informasi data secara terstruktur. Terdapat dua elemen pada JSON, yaitu *Key,* tipe string yang diapit dengan tanda kutip, dan *Value,* objek atau informasi yang mengisi *key* seperti string, boolean, angka, dan lain sebagainya (Yanti & Rihyanti, 2021). JSON menjadi ideal dalam pertukaran informasi data karena menggunakan bahasa yang umum dan mudah dipahami (Affrianto & Cahyono, 2022).

Pengguna yang akan mengakses sistem perlu aktivitas *register* dan *login.* Aktivitas *register* bagi admin baru membutuhkan hak akses admin yang telah terdaftar sebelumnya. Informasi pengguna yang telah terdaftar dapat diganti setelah melakukan aktivitas *login*. Informasi yang dapat diganti adalah nama, nomor hp, email, dan *password*. Terdapat juga aktivitas lupa dan reset *password.* Mengganti *password* membutuhkan password baru dan konfirmasi password harus sama.

4.1.1 Transaksi

Melakukan proses transfer ke dompet digital membutuhkan aktivitas transaksi. Pengguna yang akan melakukan transaksi memilih metode transfer dan mengisi data tujuan transfer, nomor tujuan, nama akun tujuan, jumlah transfer dengan minimal transfer Rp10.000, dan mengisi catatan transaksi apabila diperlukan. Sistem akan melakukan perhitungan jumlah transfer ditambah biaya administrasi dan kode unik sebagai identifikasi transaksi. Sistem akan menampilkan data transaksi untuk dikonfirmasi oleh pengguna. Data transaksi yang berhasil dibuat dapat dilihat pada Gambar 4.1. Setelah mengonfirmasi data transaksi, pengguna melakukan transfer kepada dompet digital sistem dan meng-*upload* bukti transfer ke sistem untuk dilakukan pengecekan. Bukti transfer yang berhasil di-*upload* dapat dilihat pada Gambar 4.2. Admin akan melakukan pengecekan apakah transaksi berhasil diterima atau tidak, jika saldo berhasil diterima maka akan mengirim status transaksi berhasil, dan jika saldo tidak diterima maka akan mengirim status transaksi gagal. Pengguna juga dapat melakukan pembatalan transaksi sebelum meng-*upload* bukti transaksi.

```
{
    "status": 201,
    "message": "Transaksi Berhasil dibuat",
    "user_id": "0377c315-cada-4773-a7e3-1c88b4bcb6dc",
    "nama_user": "irfan",
    "data": {
        "id": "59ac3b1a-2689-4fa4-ba81-9faab62449f7",
        "metode_bayar": "Dana",
        "tujuan": "Gopay",
        "nomor_tujuan": "+620823456621",
        "nama_akun": "Fruqon",
        "catatan": "Beli Mobil",
        "jumlah": 11000,
        "biaya_administrasi": 1000,
        "kode_unik": 350,
        "total_bayar": 12350
    }
}
```


Gambar 4.1. Respon API Transaksi


```
{
    "status": 201,
    "message": "Bukti telah diupload",
    "data": {
        "transaksi_id": "59ac3b1a-2689-4fa4-ba81-9faab62449f7",
        "bukti": "0377c315-cada-4773-a7e3-1c88b4bcb6dc-1702839535586.jpg"
    }
}
```


Gambar 4.2. Respon API *Upload* Bukti

4.1.2 Pengembalian Dana

Pengguna yang gagal melakukan transaksi tetapi telah mengirimkan dana/saldo atau kesalahan transaksi lainnya seperti dana terkirim dua kali dapat melakukan proses pengembalian dana. Pengguna yang akan melakukan proses pengembalian dana memilih transaksi mana yang perlu proses pengembalian dana. Pemilihan transaksi akan memasukkan *identifier* transaksi sebagai *request* agar proses pengembalian dana dapat dilakukan. Pengguna akan mengisi data tanggal transaksi sebagai tanggal melakukan transaksi sesuai bukti transaksi dengan format pengisian ISO 8601 (Tahun – Bulan - Hari), jumlah saldo yang akan dikembalikan sesuai bukti transaksi, nomor dengan format +62 dan nama dompet digital yang akan dikirimkan pengembalian dana, alasan mengajukan proses pengembalian dana, dan bukti yang berisi gambar agar proses pengembalian dana disetujui, bukti dapat berupa bukti transfer atau bukti lainnya sebagai pendukung proses pengembalian dana. Admin akan memeriksa apakah butuh pengembalian dana atau tidak, jika disetujui maka dana akan dikembalikan sesuai bukti yang terkirim, dan jika tidak maka pengembalian dana dianggap gagal. Proses pengembalian dana dapat dilihat pada Gambar 4.3. Data yang diperiksa oleh admin dapat dilihat pada Gambar 4.4.

```
{
    "status": 201,
    "message": "Pengembalian dana berhasil diajukan",
    "data": {
        "id": "27ed37c9-e702-411c-ab2d-fef704712ee2",
        "tanggal_transaksi": "2023-12-18",
        "jumlah_transaksi": 12350,
        "jenis_wallet": "Dana",
        "nomor_wallet": "+629823232231",
        "nama_wallet": "irfan",
        "alasan": "Dana telah terkirim tetapi dianggap gagal",
        "bukti_refund": "0377c315-cada-4773-a7e3-1c88b4bcb6dc-1702841405939.png"
    }
}
```


Gambar 4.3. Respon API *Refund*


```
{
    "id": "27ed37c9-e702-411c-ab2d-fef704712ee2",
    "tanggal_transaksi": "2023-12-18",
    "jumlah_transaksi": 12350,
    "jenis_wallet": "Dana",
    "nomor_wallet": "+629823232231",
    "nama_wallet": "irfan",
    "alasan": "Dana telah terkirim tetapi dianggap gagal",
    "bukti_refund": "0377c315-cada-4773-a7e3-1c88b4bcb6dc-1702841405939.png",
    "status_refund": "Diajukan",
    "create_at": "2023-12-17T18:54:05.088Z",
    "transaksi": {
        "id": "59ac3b1a-2689-4fa4-ba81-9faab62449f7",
        "tujuan": "Gopay",
        "nomor_tujuan": "+620823456621",
        "nama_akun": "Fruqon",
        "jumlah": 11000,
        "catatan": "Beli Mobil",
        "metode_bayar": "Dana",
        "kode_unik": 350,
        "total": 12350,
        "bukti": "0377c315-cada-4773-a7e3-1c88b4bcb6dc-1702839535586.jpg",
        "status": "Gagal",
        "create_at": "2023-12-17T18:54:05.080Z"
    }
}
```


Gambar 4.4. Respon API Data *Refund*

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini bertujuan untuk mengimplementasikan REST API pada sistem transfer dompet digital menggunakan *framework* NestJS dengan metode pengembangan *Extreme Programming*. Latar belakang penelitian mencerminkan bahwa pengguna dompet digital cenderung memiliki beberapa aplikasi dompet digital, melakukan perpindahan dompet digital untuk melakukan pembayaran belanja *online*. Sistem ini dapat digunakan oleh pengguna dompet digital dengan status basic tanpa perlu meng-*upgrade* status pada dompet digital.

Metode *Extreme Programming* digunakan sebagai pendekatan dalam mengembangkan perangkat lunak. Berisi perencanaan API yang akan dikembangkan. Perancangan menggunakan *Unified Modelling Language* (UML) *Usecase Diagram, Activity Diagram,* dan *Entity Relationship Diagram* (ERD). Pengkodean menggunakan *framework* NestJS dengan bahasa pemograman TypeScript, dilanjutkan menginstal *library* dan mengkonfigurasi sistem seperti modul, *controller* dan *service*. Pengujian API menggunakan *tools* Postman, katalog dan dokumentasi API menggunakan *Open API* Swagger.

Semua API telah berhasil diimplementasikan dan diuji mulai dari kegunaan, *error handling,* dan lain sebagainya. Arsitektur REST dapat digunakan untuk mengembangkan API agar dapat dikembangkan lebih lanjut oleh pengembang *frontend* dan *mobile*.

5.2 Saran

Berdasarkan hasil penelitian, beberapa saran yang dapat diajukan:

a. Pengembangan lanjutan

   API sistem telah diimplementasi sesuai permintaan dan telah didokumentasikan melalui *Open API* Swagger. Katalog dan dokumentasi tersebut dapat menjadi acuan agar sistem dapat dikembangkan lebih lanjut oleh pengembang *frontend* dan *mobile*. Dokumentasi juga menyertakan cara penggunaan API.

b. Pengembangan keamanan

   Keamanan sistem diharapkan dikembangkan lebih lanjut, beberapa keamanan yang diharapkan adalah pin ketika bertransaksi, dan pengamanan sistem serta data yang ada.

c. Pengoptimalan kinerja

Diharapkan kinerja sistem lebih dioptimalkan dengan menggunakan *clean code*, dan mengkombinasikan arsitektur pengembangan API, seperti kombinasi arsitektur REST dan GrapQL. REST API selalu mengembalikan seluruh set data sebagai respon API sehingga terjadinya *over-fetching*. GrapQL muncul sebagai solusi yang berbasis kueri. GrapQL dapat mengembalikan data yang diperlukan sebagai pertukaran permintaan dan respon API.

d. *Automation*

Sistem dikembangkan dan berjalan secara manual mulai dari proses transaksi, pengembalian dana, pemeriksaan transaksi, dan konfirmasi transaksi, sehingga meningkatkan kemungkinan *human error* dapat terjadi. *Automation* yang diharapkan adalah sistem *payment gateaway* agar sistem langsung mengkonfirmasi transaksi dan mempermudah pengguna dalam bertransaksi.

DAFTAR PUSTAKA

Abdullahi, A. (2022). *Extreme Programming vs Scrum: What's the difference?* CIOinsight.com.https://www.cioinsight.com/application-development/extreme programming-vs-scrum/#:~:text=Extreme programming focuses on programming,weeks to a few months.

Affrianto, M. K. N. F., & Cahyono, A. B. (2022). Implementasi REST API Pada Fitur Rencana Strategis Dalam Aplikasi *Website* E-Government (Studi Kasus CV. Atsoft Teknologi). *Jurnal AUTOMATA*.

Amijaya, A., Ferdinandus, F., & Bayu, M. (2019). Sistem Pendukung Keputusan Pemilihan Handphone Dengan Metode *Simple Additive Weighting* Berbasis WEB. *CAHAYAtech*, *8*(2), 102. https://doi.org/10.47047/ct.v8i2.47

Arief, M. R. (2010). Autentikasi, Kendali Akses, Audit Sistem Keamanan Jaringan Komputer. *Data Manajemen Dan Teknologi Informasi*, *11*(3), 73.

Arifin, S. (2017). Implementasi Monitoring Jaringan Menggunakan Raspberry Pi Dengan Memanfaatkan Protokol SMTP (Simple Mail Transfer Protocol). *Jurnal Mahasiswa Teknik Informatika*, *1*(1), 173–179. https://doi.org/https://doi.org/10.36040/jati.v1i1.1879

Benita, R. (2022). *Implementing a Generic Repository Pattern Using NestJS.pdf*. Medium.com. https://betterprogramming.pub/implementing-a-generic-repository-pattern-using-nestjs-fb4db1b61cce

Binuko Paksi, A., Hafidhoh, U., & Kariagil Bimonugroho, S. (2023). Perbandingan Model Pengembangan Perangkat Lunak Untuk Proyek Tugas Akhir Program Vokasi Program Studi D3 Teknologi Informasi, Politeknik Negeri Madiun. *JURNAL MASYARAKAT INFORMATIKA*, *14*(1), 2777–0648. https://doi.org/https://doi.org/10.14710/jmasif.14.1.52752

Bondel, G., Landgraf, A., & Matthes, F. (2021). *API Management Patterns for Public, Partner, and Group Web API Initiatives with a Focus on Collaboration*. *26th European Conference on Pattern Languages of Programs*, 1–17. https://doi.org/https://doi.org/10.1145/3489449.3490012

Cholifah, W. N., Yulianingsih, Y., & Sagita, S. M. (2018). Pengujian *Black Box Testing* Pada Aplikasi *Action* & *Strategy* Berbasis Android dengan Teknologi Phonegap. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, *3*(2), 206–210. https://doi.org/http://dx.doi.org/10.30998/string.v3i2.3048

Darmawansyah, & Mohd., S. (2018). *Restful Web Service* Untuk Pemantauan Dan Pengendalian Peternakan Ayam Broiler. *KITEKTRO: Jurnal Online Teknik Elektro*, *3*(2), 53–59.

Dewi, L. P., Indahyanti, U., & Hari, Y. (2012). Pemodelan Proses Bisnis Menggunakan *Activity Diagram* Uml dan Bpmn *(studi kasus frs online)*. Petra Christian University.

ExpressJS Documentation. (2023). *ExpressJs Documentation*.pdf. Expressjs.com. https://expressjs.com/

Fachrizal, R. (2022). SurveySensum: 42 Persen Pengguna *E-commerce* Memiliki Loyalitas Rendah.pdf. INFOKOMPUTER.Com. https://infokomputer.grid.id/read/123149401/surveysensum-42-persen-pengguna-e-commerce-memiliki-loyalitas-rendah?page=all

Fatoni, A., & Dwi, D. (2016). Rancang Bangun Sistem *Extreme Programming* Sebagai Metodologi Pengembangan Sistem. *Prosisko*, *3*(1), 1–4. http://e-jurnal.lppmunsera.org/index.php/PROSISKO/article/view/116

Feridi. (2019). Mengenal *RESTful Web Services*.pdf. https://codepolitan.com/blog/mengenal-restful-web-services

Gasparik, M. (2023). *Understanding UUID: Purpose and Benefits of a Universal Unique Identifier.pdf*. Medium.com.

Gumelar, T., Astuti, R., & Sunarni, A. T. (2018). Sistem Penjualan Online Dengan Metode *Extreme Programming*. *Telematika Mkom*, *9*(2), 87–90. https://doi.org/https://dx.doi.org/10.36080/telematikamkom.531

Gunawan, R., & Rahmatulloh, A. (2019). *JSON Web Token* (JWT) untuk *Authentication* pada Interoperabilitas Arsitektur berbasis *RESTful Web Service*. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *5*(1), 74. https://doi.org/10.26418/jp.v5i1.27232

Haniefardy, A., Fadhillah, M. B. A., & Rochimah, S. (2019). Tinjauan Literatur Sistematis: Pengaruh Penggunaan *Framework* Khusus dalam Proses Pengembangan *Web* dan Pembuatan *Web*. *Matrix : Jurnal Manajemen Teknologi Dan Informatika*, *9*(2), 68–73. https://doi.org/10.31940/matrix.v9i2.1161

Haryadi, H. L., Sujjada, A., & Simatupang, D. S. (2023). Perbandingan *REST API* Menggunakan NodeJS Dan Php Pada Aplikasi Pemilihan Umum. *Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)*, *8*(2), 460–468. https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Hasanuddin, Hari, A., & Budi, H. (2022). Rancang Bangun *REST API* Aplikasi WESHARE

Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan. *JINTEKS (Jurnal Informatika Teknologi Dan Sains)*, *4*(1), 8 – 14. https://doi.org/10.3139/9783446473157.024

Heriyanto, Y. (2018). Perancangan Sistem Informasi Rental Mobil Berbasis *Web* Pada PT. APM Rent Car. *Jurnal Intra Tech*, *2*(2), 64–77. https://doi.org/https://doi.org/10.37030/jit.v2i2.35

Irawan, Y. (2017). Pengujian Sistem Informasi Pengelolaan Pelatihan Kerja UPT BLK Kabupaten Kudus dengan Metode *Whitebox Testing*. *Sentra Penelitian Engineering Dan Edukasi*, *9*(3), 59–63.

Kadence International. (2021). *Digital Payment and Financial Services Usage and Behavior in Indonesia*. PT Kadence International. https://kadence.com/wp-content/uploads/2021/09/Kadence-Digital-Payment-and-Financial-Services-Usage-and-Behavior-in-Indonesia.pdf

Kurniawan, I., Humaira, & Rozi, F. (2020). *REST API* Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android. *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, *1*(4), 127–132. https://doi.org/10.30630/jitsi.1.4.18

Mahendra, I., & Eby Yanto, D. T. (2018). Sistem Informasi Pengajuan Kredit Berbasis *Web* Menggunakan *Agile Development Methods* pada Bank Bri Unit Kolonel Sugiono. *Jurnal Teknologi Dan Open Source*, *1*(2), 13–24. https://doi.org/10.36378/jtos.v1i2.20

Melinda, M., Borman, R. I., & Susanto, E. R. (2018). Rancang Bangun Sistem Informasi Publik Berbasis *Web* (Studi Kasus : Desa Durian Kecamatan Padang Cermin Kabupaten Pesawaran). *Jurnal Tekno Kompak, 11*(1), 1. https://doi.org/10.33365/jtk.v11i1.63

Mualim, W., & Putra, G. U. (2017). Implementasi *Framework* MVC Pada Sistem Informasi Akademik Di STMIK Yadika Bangil. *Jurnal SPIRIT*, *9*(2), 35–39. https://doi.org/http://dx.doi.org/10.53567/spirit.v9i2.83

Muhamad Saepuloh, A., & Ginting, S. (2022). Perancangan Sistem Informasi Manajemen Proyek Dengan Menggunakan *Software* Nest.Js Berbasis *Web* Di Pt. Mitra Pajakku. *INFOKOM (Informatika & Komputer)*, *10*(1), 1–9. https://doi.org/10.56689/infokom.v10i1.818

Mulyana, A., & Wijaya, H. (2018). Perancangan *E-Payment* System pada *E-Wallet* Menggunakan Kode QR Berbasis Android. *Komputika : Jurnal Sistem Komputer*, *7*(2), 63–69. https://doi.org/10.34010/komputika.v7i2.1511

NetsJS Documentation. (2023). *NestJS Documentation*.pdf https://docs.nestjs.com/

Nurfauziah, H., & Jamaliyah, I. (2022). Perbandingan Metode Testing Antara *Blackbox* Dengan *Whitebox* Pada Sebuah Sistem Informasi. *Jurnal Visualika*, *8*(2), 105–113. https://jurnas.saintekmu.ac.id/index.php/visualika/article/view/24

Nurhayati, E., & Agussalim, A. (2023). Rancang Bangun *Back-end API* pada Aplikasi Mobile AyamHub Menggunakan *Framework* Node JS Express. *Jurnal Sistem Dan Teknologi Informasi (JustIN)*, *11*(3), 524. https://doi.org/10.26418/justin.v11i3.66823

Ozen, K. (2022). *Express vs NestJS.pdf*. Medium. https://medium.com/@karahanozen/express-js-vs-nest-js-2e39fc0ce22c

Pangestika, R., & Dirgahayu, R. T. (2020). Pengembangan *Back-end* Sistem Informasi Pendataan Sekolah Desa Komunitas Pendar Foundation. *Automata*, *1*(2), 184–189.

Pham, A. D. (2020). *Developing back-end of a web application with NestJS framework Case: Integrify Oy's student management system*. 46.

Prasena, R. R., & Sama, H. (2020). Studi Komparasi Pengembangan *Website* dengan *Framework* Codeigniter dan Laravel. *Conference on Business, Social Sciences and Innovation Technology*, *1*(1), 614–621. https://journal.uib.ac.id/index.php/cbssit/article/download/1469/969/

Pratama, I. G. A. E., Satwika, I. P., & Anggara Wijaya, I. N. Y. (2022). Analisis Perbandingan Performa API Metode REST dan GRAPHQL dengan PHP dan GO. *Jurnal Teknologi Informasi Dan Komputer*, *8*(4 SE-Articles). https://jurnal.undhirabali.ac.id/index.php/jutik/article/view/2088

Purwanto, E., & Kom, S. (2012). Perbandingan Strategi Replikasi Pada Sistem Basis Data Terdistribusi. *Jurnal Informatika*, 1–8.

Putra, S. R. P. (2020). Panduan *Back End Development* Bagi Pemula.pdf. Medium.com. https://glovorytech.medium.com/panduan-back-end-development-bagi-pemula-dbee0619ca93

R. Ramli, R., & Ika, A. (2022). *Survey* Penggunaan Dompet Digital di Indonesia. Kompas.com. https://money.kompas.com/read/2022/07/21/203000626/ini-5-dompet-digital-yang-paling-banyak-dipakai-warga-ri-siapa-juaranya-?page=all#google_vignette

Sakti, D. V. S. Y., Agani, N., & Hardjianto, M. (2016). Pengamanan Sistem Menggunakan *One Time Password* dengan Pembangkit *Password Hash* SHA-256 dan *Pseudo Random Number Generator* (PRNG) *Linear Congruential Generator* (LCG) di Perangkat Berbasis Android. *Bit*, *13*(1), 1–10. https://doi.org/https://dx.doi.org/10.36080/bit.v13i1.442

Saputra, P. S., & Tjahyanti, L. P. A. S. (2022). Pemanfaatan Teknologi Informasi

Menggunakan *Web API* di Masa Pandemi Covid-19. *KOMTEKS*, *1*(1).

Shinta, A. (2021). Apa itu Node.js*?* Pengertian, Kelebihan, dan Contoh Penggunaannya.pdf. https://www.dewaweb.com/blog/mengenal-node-js/#:~:text=Bagi pemula di bidang programming,-source dan cross-platform.

Silalahi, P. R., Safira, R., Hubara, Z. A., & Sari, E. P. (2022). Pengaruh Dompet Digital Terhadap Budaya Belanja Individu di Kota Medan. *EKOMBIS REVIEW: Jurnal Ilmiah Ekonomi Dan Bisnis*, *10*(2), 869–878. https://doi.org/10.37676/ekombis.v10i2.2673

Suryantara, I. G. N. (2017). Merancang Aplikasi dengan Metodologi *Extreme Programmings* (Issue March). Elex Media Komputindo.

Susrama, G., Diyasa, M., Budiwitjaksono, G. S., Amarul, H., & Ade, I. (2021). *Comparative Analysis of Rest and GraphQL Technology on Nodejs-Based Api Development*. *2021*, 1–9. https://doi.org/10.11594/nstp.2021.0908

Swagger. (2023). *Swagger Documentation*. Swagger.com. https://swagger.io/specification/

Syafaat, N. (2018). Pemrograman Aplikasi *Mobile Smartphone* dan Tablet PC Berbasis Android. *Jurnal Teknik Informatika*, *11*(1), 1–18.

Tanaem, P. F., & Iriani, A. (2021). Perbandingan *Web Service* Berbasis *SOAP* dan *RESTFUL*. *Seminar Nasional & Konferensi Ilmiah Sistem Informasi, Informatika & Komunikasi, SE-Articles*, 87–91. https://publikasi.uyelindo.ac.id/index.php/semmau/article/view/65

Trisnadoli, A. (2021). Implementasi *Extreme Programming* (XP) *Agile Software Development* pada Pengembangan Sistem Informasi KELUARGAKU. *Jurnal Informatika Universitas Pamulang*, *6*(2), 305–311. https://doi.org/10.32493/informatika.v6i2.10088

Uma, B. (2023). API: Pengertian, Jenis, Cara Kerja, Arsitektur, dan Contohnya. 1–9. https://bamai.uma.ac.id/2023/03/17/api-pengertian-jenis-cara-kerja-arsitektur-dan-contohnya/

Vijayakumar, T. (2018). *Practical API Architecture and Development with Azure and AWS*. https://doi.org/10.1007/978-1-4842-3555-3

Wijayanto, B. (2011). Implementasi *Object Relational Mapping* (Orm) Menggunakan *Hibernate* (Studi Kasus: Aplikasi Peminjaman Inventaris Program Studi Informatika Unsoed). *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*.

Wulandari, C. S. (2023). Dompet digital naik daun, membetot minat kala pandemi. Bi.Go.Id. https://www.bi.go.id/id/bi-institute/BI-Epsilon/Pages/Dompet-Digital--Naik-Daun,-Membetot-Minat-Kala-Pandemi.aspx

Yanti, S. N., & Rihyanti, E. (2021). Penerapan REST API untuk Sistem Informasi Film Secara

Daring. *Jurnal Informatika Universitas Pamulang*, *6*(1), 195. https://doi.org/10.32493/informatika.v6i1.10033

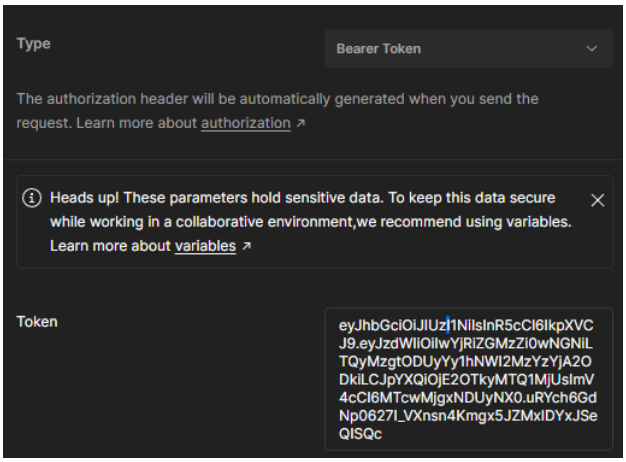
LAMPIRAN

F. *Register* dan *Login*

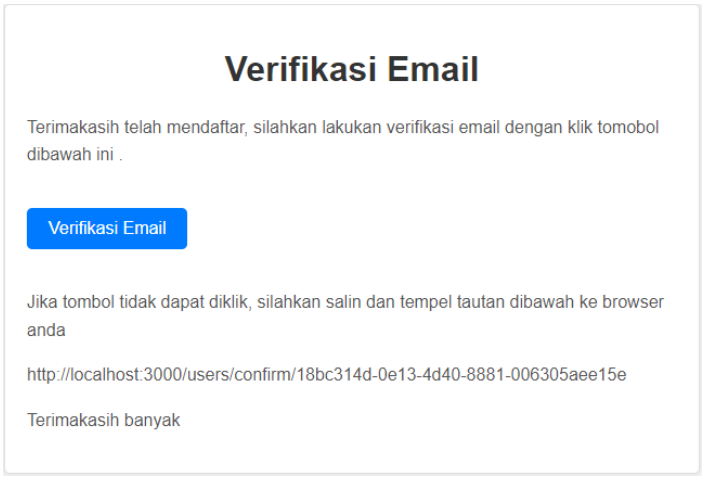

```
{
    "status": 201,
    "message": "Berhasil mendaftarkan akun",
    "data": {
        "id": "6c214932-50c7-4649-ba35-8d7598ec0ea0",
        "nama_lengkap": "user13",
        "email": "user13@gmail.com",
        "role": "Pengguna"
    }
}
```



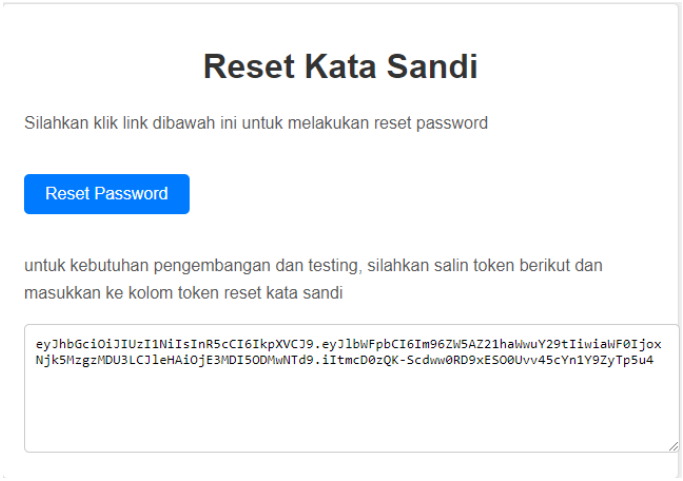
```
{
    "status": 201,
    "message": "Berhasil Login Sebagai Pengguna",
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJzdWIiOiIwYjRiZGMzZi0wNGNiLTQyMzgtODUyYy1hNWI2MzYzYjA2ODkiLCJpYXQiOjE2OTkyNjc2NzgsImV4cCI6MTcwMjg2NzY3OH0.
        EZ7nM-pMY2-jmzHxxVzzUB6rPMVjX8GNSqEItD7LMdg",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJqaWQiOiIzMzFmYTc4YS1jNWZkLTRlNGUtODIzNC02MTgwMDI1NzYxZDEiLCJpYXQiOjE2OTkyNjc2NzgsImV4cCI6NDI5MTI2NzY3OH0.
        QPTvMeYkB2y2YfF3zhvX37p0HZ3Pr_4zM_ElblqJGZQ"
}
```

G. Konfirmasi Email dan *Reset Password*



```
{"status":201,"message":"Konfirmasi email berhasil"}
```


```
{
  "statuscode": 201,
  "message": "Cek Email Anda Untuk Langkah Selanjutnya"
}
```


H. Menampilkan Data dan Riwayat Transaksi atau *Refund*


```
{
    "id": "a90ed778-843d-45ff-bfd5-1cf241e1f8d0",
    "tanggal_transaksi": "2023-12-13",
    "jumlah_transaksi": 11020,
    "jenis_wallet": "Gopay",
    "nomor_wallet": "+629823232231",
    "nama_wallet": "irfan",
    "alasan": "Dana telah terkirim tetapi dianggap gagal",
    "bukti_refund": "c4712c29-16e6-4241-be40-c8342dcee982-1699440561104.jpg",
    "status_refund": "Disetujui",
    "create_at": "2023-11-06T13:10:08.658Z"
}
```



```
{
    "id": "2fa5a4ce-3bd1-418d-ba90-881722058659",
    "nama_lengkap": "Administrasi Konfirmasi",
    "email": "admin2@gmail.com",
    "password": "$2b$10$BcnousP1gaIoFvCoshvYiu/Dbeix0jqrJ0d.J2a8/7aJUhMhi2l.S",
    "salt": "$2b$10$BcnousP1gaIoFvCoshvYiu",
    "role": "Administrasi Keuangan",
    "emailVerified": true,
    "create_at": "2023-10-17T15:59:51.212Z",
    "update_at": "2023-10-17T16:00:01.000Z",
    "pengguna": null,
    "admin": {
        "id": "a04e234d-1b31-41e5-90d8-ea49496d49bc",
        "posisi": "AdminPemeriksaanTransaksi",
        "created_at": "2023-10-17T15:59:51.221Z",
        "updated_at": "2023-10-17T16:00:52.000Z"
    },
    "adminIT": null,
    "refreshToken": [
        {
            "id": "2a170b98-760e-4f02-82cd-e2c60783aa0a",
            "isRevoked": false,
            "expiredAt": "2023-11-16T16:00:29.000Z"
        },
```

```
{
    "transaksi_id": "59ac3b1a-2689-4fa4-ba81-9faab62449f7",
    "metode_bayar": "Dana",
    "tujuan": "Gopay",
    "nomor_tujuan": "+620823456621",
    "nama_akun": "Fruqon",
    "catatan": "Beli Mobil",
    "jumlah": 11000,
    "kode_unik": 350,
    "total": 12350,
    "bukti": "0377c315-cada-4773-a7e3-1c88b4bcb6dc-1702839535586.jpg",
    "status": "Gagal",
    "create_at": "2023-12-17T18:54:05.080Z",
    "data_user": {
        "user_id": "0377c315-cada-4773-a7e3-1c88b4bcb6dc",
        "user_name": "irfan"
    }
}
```

```
{
    "id": "0c94b542-8556-440c-bdc8-f671e6840656",
    "tujuan": "LinkAja",
    "nomor_tujuan": "+6282145523004",
    "nama_akun": "Irfan",
    "jumlah": 10000,
    "catatan": "Beli Mobil",
    "metode_bayar": "Shopeepay",
    "kode_unik": 480,
    "total": 11480,
    "bukti": null,
    "status": "Menunggu",
    "create_at": "2023-11-07T21:12:50.894Z"
},
{
    "id": "207872c7-0097-4864-8a78-50be14f764ea",
    "tujuan": "Gopay",
    "nomor_tujuan": "+620823456621",
    "nama_akun": "Fruqon",
    "jumlah": 11000,
    "catatan": "Beli Mobil",
    "metode_bayar": "Dana",
    "kode_unik": 216
```


I. Mengubah Status Transaksi dan *Refund*


```
{
    "message": "Transaksi dianggap berhasil"
}
```

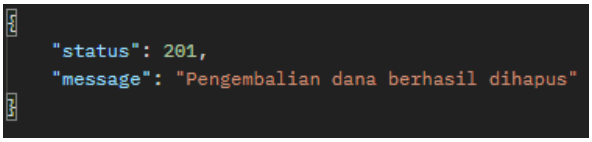
```
{
    "message": "Transaksi dianggap gagal"
}
```

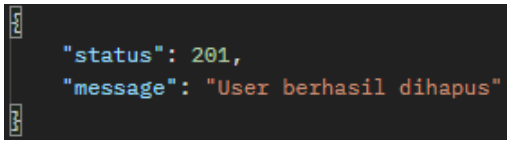
```
{
    "message": "Pengembalian dana disetujui"
}
```

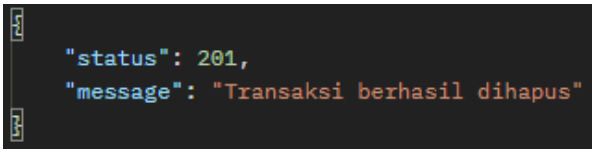
```
{
    "message": "Pengembalian dana ditolak"
}
```

```
{
    "message": "Pengembilian dana berhasil dan dana telah dikirim"
}
```

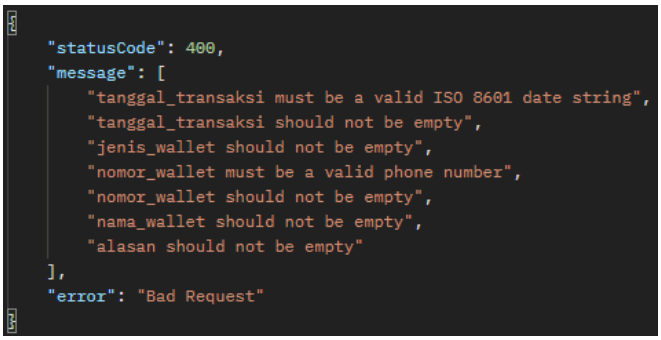
J. Menghapus Data

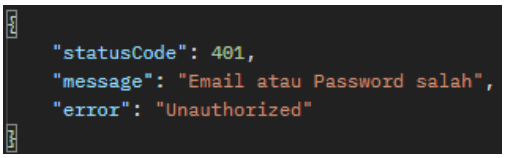


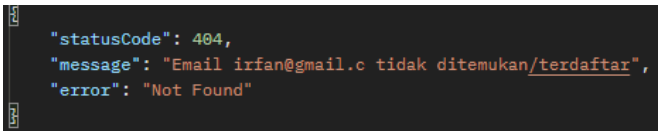




K. *Error Handling*







L. *Open API* atau Swagger