open access
Published online on: https://journal.uii.ac.id/jurnalsnati/
Check for updates
DOI: 10.20885/snati.v3.i2.35
JURNAL SNATI Sains, Nalar, dan Aplikasi Teknologi Informasi
Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi
Vol. 3 No. 2 (2024) 79 - 87 ISSN Media Electronic: 2807-5935

# Pengembangan Backend Server Berbasis Arsitektur REST API pada Sistem Transfer Dompet Digital

*Development of Backend Server Based on REST API Architecture in E-Wallet Transfer System*

Irfan Rizq Dzaky Muhammad[1], Irving V. Paputungan[2]
[1,2]Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta, Indonesia
[1]18523279@students.uii.ac.id, [2]irving@uii.ac.id

***Abstract***

*Cash payments are starting to shift towards an electronic or cashless payment system. While the process of transferring balances between ewallet at this time is considered less effective because it requires withdrawal from ewallet and replenishment using banks. This paper presents how to develop a Backend Server based on REST API Architecture using the NestJS framework on a ewallet transfer system. The development uses NestJS framework tools, MySQL database, API testing with Postman, Extreme Programming (XP) development method, and BlackBox Testing method. The result of this research is a server side system for transferring funds between ewallet in the form of a REST API that can be used by frontend and mobile developers.*

*Keywords: Backend; Transfer; REST API; E-Wallet*

**Abstrak**

Pembayaran secara tunai mulai bergeser ke arah sistem pembayaran elektronik atau *cashless*. Sedangkan proses pemindahan saldo antar dompet digital pada saat ini dirasa kurang efektif karena membutuhkan penarikan dari dompet digital dan pengisian menggunakan bank. Makalah ini menyajikan bagaimana sebuah Pengembangan *Backend Server* Berbasis Arsitektur REST API pada sistem transfer dompet digital. Pengembangan menggunakan *tools* framework NestJS, basis data MySQL, Pengujian API dengan Postman, metode pengembangan *Extreme Programming* (XP), dan metode pengujian *BlackBox Testing*. Hasil dari penelitian ini adalah sistem *server side* transfer dana antar dompet digital berupa REST API yang dapat digunakan oleh pengembang *frontend* dan *mobile*.

Kata kunci: *Backend*; Pembayaran; REST API; Dompet Digital

## 1. Pendahuluan

Pengembangan teknologi telah berdampak pada ekonomi negara dengan mengubah cara pembayaran dari menggunakan uang tunai menjadi metode pembayaran elektronik. Uang elektronik (*e-money*) dan uang virtual (*virtual money*) merupakan metode pembayaran elektronik [1]. Definisi dompet digital menurut Bank Indonesia, yang disampaikan melalui lembaran negara Bank Indonesia nomor 18/40/PBI/201, merujuk pada layanan elektronik yang memfasilitasi penyimpanan data instrumen pembayaran seperti kartu dan/atau uang elektronik. Ini memungkinkan untuk melakukan transaksi pembayaran. Istilah "kartu" dalam definisi Bank Indonesia mengacu pada *e-money*, di mana *chip* dalam *e-money* berperan penting dalam setiap transaksi yang dilakukan. Sebaliknya, dompet digital menggunakan aplikasi berbasis *server* dan memerlukan koneksi internet [2]. 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 dari "Tren *e-commerce* 2022 SurverSenyum" [3], perilaku pengguna *e-commerce* menunjukkan kecenderungan memiliki tingkat loyalitas yang rendah. Dari 1000 responden yang diambil, 41% dari mereka sering beralih antara situs *e-commerce* yang satu dengan yang lain, 31% menggunakan hanya satu layanan *e-commerce*, sementara 27% beralih hanya pada beberapa kategori produk atau metode pembayaran. Alasan utama di

Received: 19-12-2023 | Revised: 22-01-2024 | Accepted: 23-01-2024 | Published: 31-01-2024

balik perilaku perpindahan *e-commerce* ini adalah ketersediaan barang atau produk yang lebih luas, disebutkan oleh 81% responden, sementara 71% menyatakan bahwa alasan peralihan tersebut adalah karena suatu *e-commerce* menawarkan harga yang lebih terjangkau atau memberikan nilai yang sepadan dengan uang yang dikeluarkan. Pengguna menginginkan produk dengan harga yang bersahabat namun tetap memiliki kualitas yang tinggi. Selain itu, survei juga menunjukkan bahwa 88% pengguna *e-commerce* lebih memilih metode pembayaran menggunakan *e-wallet*.

Hasil survei tersebut menunjukkan bahwa pengguna *e-commerce* perlu menyimpan dana di berbagai platform dompet digital. Proses pengisian saldo ke dompet digital bisa dilakukan melalui layanan perbankan seperti ATM atau *mobile banking*, namun proses ini biasanya melibatkan biaya administrasi. Namun, sayangnya, saat ini tidak ada cara untuk mentransfer saldo dari satu dompet digital ke yang lain, meskipun dengan biaya administrasi yang terkait. Oleh karena itu, diperlukan suatu sistem yang memungkinkan pemindahan saldo antar dompet digital guna menghemat waktu dan mengurangi biaya administrasi yang terkait dengan pengisian saldo.

Penelitian terkait Perancangan *E-Payment System* Pada *E-wallet* Menggunakan Kode QR Berbasis Android [2] mengangkat masalah dimana pengguna hanya bisa mentransfer dana ke dompet digital yang sama. *E-wallet* sebagai media pembayaran seharusnya dapat memindahkan saldo dompet digital ke dompet digital lainnya. Penelitian tersebut hanya memindahkan saldo pembeli ke penjual barang atau jasa melalui teknologi QR berbasis android, sehingga dari *literatur review* yang dilakukan belum terdapatnya sistem untuk memindahkan saldo pengguna dompet digital yang berbeda.

Artikel ini menyajikan hasil dari pengerjaan proyek sebagai pengembang *backend* yang berhasil mengembangkan *Application Programming Interface* (API) pada sistem *server-side* sehingga memungkinkan penggunaan transfer dana antar dompet digital oleh pengembang *frontend* dan *mobile*. API adalah antarmuka yang dirancang pada sistem *server-side* (*backend*) untuk bertukar data dan memungkinkan interaksi antara sisi *client* (*frontend*) dan sisi *server* (*backend*). API dapat digunakan dalam beragam bahasa pemrograman serta dapat beroperasi di berbagai jenis *server* seperti Apache, NGINX, Tomcat, dan sebagainya [4]. API sistem menggunakan arsitektur *Representational State Transfer* (REST). *Representational State Transfer* (REST) merupakan salah satu arsitektur API yang bersifat terdistribusi yang paling banyak digunakan. REST menggunakan protokol *request* dan *response* dengan metode GET, POST, PUT, dan DELETE pada *Hypertext Transfer Protokol* (HTTP) serta keluaran dalam format JSON [5].

## 2. Metodologi Penelitian

*Agile Software Development* merupakan metode pengembang perangkat lunak yang iteratif dan cepat, memerlukan komunikasi yang terorganisir antar tim [6]. Beberapa model pengembangan perangkat lunak *Agile Software Development*, yaitu *Extreme Programming* (XP), *Adaptive Software Development, Dynamic Systems Development method, Scrum,* dan *Agile Modelling* [6].

Metode yang digunakan dalam penelitian adalah metode *Extreme Programming* (XP). *Extreme programming* (XP) merupakan metode pengembangan yang berfokus pada pengembangan kode atau *coding* [7]. *Extreme programming* (XP) bersifat iteratif dan dapat terjadi perulangan disetiap fasenya [8], berbeda dengan metode waterfall yang bersifat sistematik dan sekuensial, metode waterfall mengharuskan penyelesain suatu fase sebelum masuk fase selanjutnya [9].

Tahapan metode pengembangan *Extreme Programming* (XP) [10] adalah sebagai berikut:

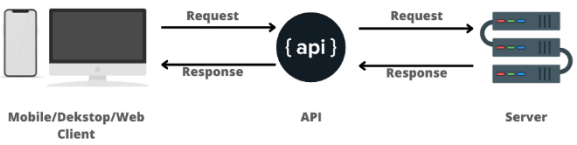
1. Perencanaan (*Planning*) merupakan tahap yang meliputi pemahaman fungsi dan fitur aplikasi [10].
2. Perancangan (*Design*) merupakan tahap yang meliputi proses pendefinisian komponen-komponen perangkat lunak.
3. Pengkodean (*Coding*) merupakan tahap yang penerjemahan proses perencanaan ke dalam bahasa pemograman yang dikenali komputer [11].
4. Pengujian (*Testing*) merupakan tahap yang meliputi proses pengujian fungsionalitas dan respon API.

Selain itu, pengembangan API ini menggunakan *framework* NestJS dengan bahasa pemograman Typescript. Basis data menggunakan MySQL. Kode editor menggunakan Visual Studio Code. API akan diuji menggunakan *tools* Postman.

## 3. Hasil dan Pembahasan

Arsitektur REST API sistem memiliki cara kerja yang akan digambarkan pada Gambar 1. API bekerja ketika *client* membutuhkan sebuah data atau informasi dan melakukan protokol *request* kepada *server*, *server* akan memberikan *response* berupa data atau informasi dalam format *JavaScript Object Nation* (JSON) tanpa mengubah data asli pada *server*. *Request* dapat berupa metode GET, POST, PUT, dan DELETE.

*JavaScript Object Nation* (JSON) merupakan format untuk melakukan penyimpanan dan pertukaran informasi data secara terstruktur. Terdapat dua elemen pada *JavaScript Object Nation* (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 [4].




Gambar 1. Cara Kerja API

Berikut adalah hasil dan pembahasan pengunaan metode *Extreme Programming* pada penelitian ini:

### 3.1. Perencanaan *(Planning)*

Tahapan perencanaan dilakukan dengan merancang kebutuhan fungsionalitas sistem yang akan dijelaskan pada **Error! Reference source not found.**. Kolom *ID* berisi identifikasi berdasarkan setiap kebutuhan sistem. Setiap kebutuhan sistem akan dibagi menjadi beberapa grup kebutuhan. Kolom kebutuhan berisi aktivitas sistem dengan didahului metode *endpoint.* Kolom terakhir berisi deskripsi dari setiap kebutuhan sistem.

Tabel 1. Kebutuhan Fungsionalitas Sistem

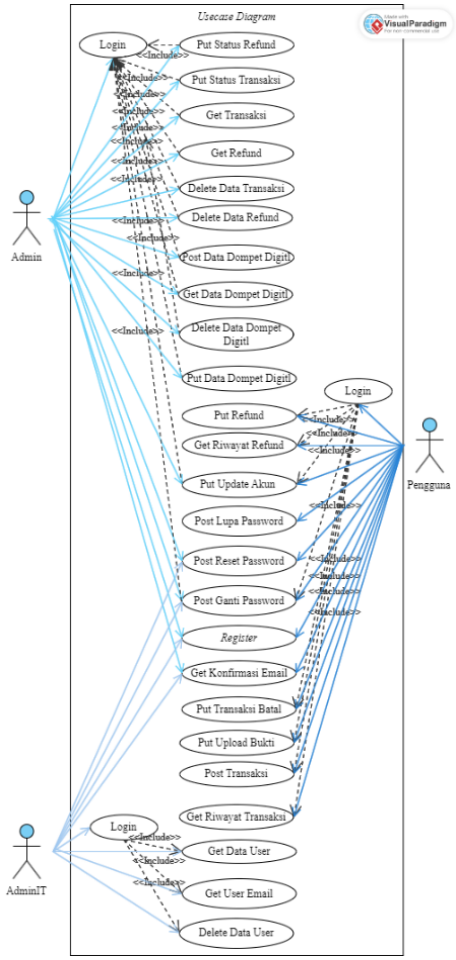
| ID | Grup Kebutuhan | Kebutuhan | Deskripsi |
|---|---|---|---|
| REQ-01.01 | Autentikasi | *Post Register* | Proses untuk mendaftarkan data *user* |
| REQ-01.02 | | *Post Login* | Proses untuk mendapatkan akses sitem yang diproteksi |
| REQ-01.03 | | *Get* Konfirmasi Email | Proses untuk konfirmasi email melalui *link* yang dikirim ke email |
| REQ-02.01 | *Users* | *Get User* | Proses untuk melihat data *user* dengan atau tidak dengan 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-03.04 | | *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 berdasarkan *user 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 dengan parameter *id* |
| 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 menggunakan *Unified Modelling Language* (UML). UML merupakan teknik memodelkan sistem [12]. Pemodelan sistem dengan memvisualisasi, merancang, dan mendokumentasikan sistem. Pemodelan sistem pada penelitian ini menggunakan *usecase diagram*, alur logika sistem menggunakan *activity diagram,* dan rancangan basis data menggunakan *Entity Relationship Diagram* (ERD).

*Usecase diagram* merupakan pendeskripsian *behavior* suatu sistem dengan menghubungkan interaksi antara aktor dan sistem [13]. *Usecase Diagram* mempunyai 3 aktor yaitu pengguna, admin, dan adminIT dan dapat dilihat pada Gambar 2. Ketiga aktor tersebut dapat melakukan *login*, konfirmasi email, lupa dan reset kata *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 *refund*. Admin secara khusus dapat melakukan aktivitas tambah akun admin, lihat transaksi, konfirmasi transaksi, tolak

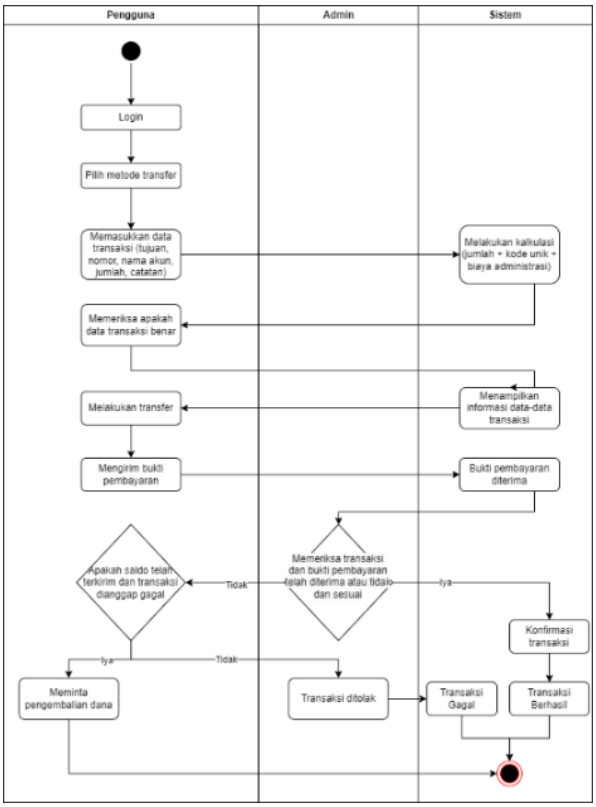
transaksi, tambah, lihat, hapus, dan ubah data dompet digital. AdminIT secara khusus dapat melakukan lihat data dan hapus pengguna.




Gambar 2. *Usecase Diagram*

Alur logika digambarkan menggunakan *Activity Diagram* dan diambil berdasarkan *Usecase Diagram* yang telah dibuat. *Activity Diagram* merupakan salah satu pemodelan dalam *Unified Modelling Language* (UML) yang memvisualisasikan aliran fungsionalitas suatu sistem. *Activity Diagram* akan mendefiniskan kapan mulai dan berhentinya suatu *workflow*, urutan serta aktifitas apa saja yang terjadi pada *workflow* tersebut [14].

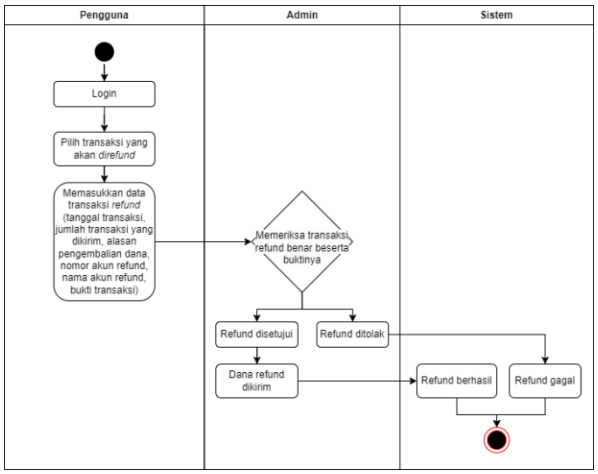
Pengguna yang akan mengakses sistem perlu melakukan proses *register* dan *login*. Informasi pengguna yang telah terdaftar dapat diganti dan diperbarui kapan saja setelah melakukan proses *login*. Informasi yang dapat diganti adalah informasi nama, nomor hp, email dan *password*. Email yang diganti haruslah email yang belum terdaftar pada sistem. Mengganti *password* membutuhkan data atau informasi *password* baru dan konfirmasi *password* harus sama.



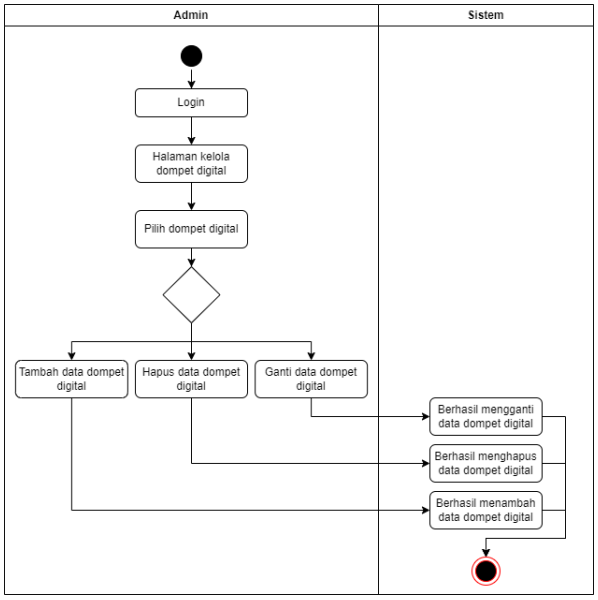

Gambar 3. *Activity Diagram* Transaksi

Melakukan proses transfer ke dompet digital memerlukan aktivitas transaksi yang dapat dilihat pada Gambar 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, 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.

Pengguna yang gagal melakukan transaksi tetapi telah mengirim dana/saldo atau kesalahan transaksi lainnya seperti terkirim dua kali bisa melakukan proses pengembalian dana. Alur proses pengembalian dana dapat dilihat pada Gambar 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, alasan meminta pengembalian dana, nama akun dompet digital yang akan dikirim dana 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.




Gambar 4. *Activity Diagram* Pengembalian Dana




Gambar 5. *Activity Diagram* kelola dompet digital

Admin dapat mengelola informasi dompet digital yang akan digunakan sebagai data pembayaran sistem. Admin dapat menambah, mengubah, dan menghapus data dompet digital terdaftar. Proses kelola dompet digital dapat dilihat pada Gambar 5.

Pengelolaan data agar dapat digunakan pada sistem membutuhkan analisis yang baik. Analisis data merupakan metode yang berguna untuk pengambaran data, hubungan data, semantik data dan batasan data pada sistem informasi. Cara untuk menganalis dan memodelkan data dapat menggunakan *Entity Relationship Data* (ERD) [15].

*Entity Relationship Diagram* (ERD) merupakan pemodelan basis data relasional. *Entity Relationship Diagram* (ERD) berguna sebagai pengambaran bagaimana basis data yang akan dibuat bekerja. *Entity Relationship Diagram* (ERD) dibuat berdasarkan kebutuhan fungsionalitas sistem dan dapat dilihat pada Gambar 6. *Entity Relationship Diagram* (ERD) dapat membantu dalam perancangan dan analisa sistem karena dapat menunjukkan komponen data yang dibutuhkan dan hubungan antar data didalamnya [16].


Gambar 6. *Entity Relationship Diagram* (ERD)

### 3.3. Pengkodean *(Coding)*

Tahap pengkodean (*coding*) menggunakan *framework* NestJS. *Framework* merupakan kerangka kerja berisi kumpulan fungsi yang siap digunakan untuk tujuan tertentu [17]. Manfaat penggunaan *framework* adalah membantu perancangan, pembuatan, pengujian dan pemeliharaan sistem [18].

NestJS (Nest) dikembangkan oleh Kamil Myśliwiec untuk pengembangan *backend* Node.js yang efektif dan terstuktur. Nest mendukung Javascript dan juga Typescript dengan mengombinasikan elemen *Object Oriented Programming* (OOP), *Functional Programming* (FP), dan *Functional Reactive Programming* (FRP) [19]. NestJS memiliki struktur utama sebagai berikut:

- NestCLI (*Nest Command Line Interface*) merupakan alat yang membantu dalam pengembangan dan pemeliharan aplikasi berbasis Nest [19]. NestCLI dapat digunakan untuk penginstalan *library* atau menjalankan aplikasi.
- *Module* merupakan sebuah kelas yang ditandai dengan sebuah dekorator @Module berguna sebagai mengatur struktur aplikasi seperti *Controller*, *Service*, *Middleware*, dan lain sebagainya [19].

- *Controller* berguna mengatur HTTP *request* dan *response* yang akan dikirim ke *client*. *Controller* juga mengatur *endpoint* yang akan digunakan dan dikirim ke *server* [19].
- *Provider*/*Service* berguna sebagai bagian yang digunakan untuk melakukan proses diluar HTTP *request*, seperti koneksi basis data, dan melakukan *request* terhadap *microservice* [19].
- *Middleware* 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 [19].

Tahapan 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 versi NPM 9.6.7. Mulai membuat proyek dengan memasukkan perintah pada Gambar 7 pada terminal OS. Perintah *project-name* pada Gambar 7 diganti sesuai nama proyek yang akan dibuat.

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

Gambar 7. Konfigurasi proyek

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

- TypeORM
  TypeORM merupakan fitur untuk mengintegrasikan sistem dengan basis data. Masukkan perintah pada Gambar 8 untuk menginstal TypeORM. Setelah penginstalan perlu untuk mengkonfigurasikan basis data pada modul utama sistem.

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

Gambar 8. Instal TypeORM

- *ValidationPipe*
  *ValidationPipe* merupakan bagian dari *pipe*. *Pipe* merupakan sebuah kelas yang diberi *decorator* @Injectable. *Pipe* memiliki dua fungsi utama, yaitu *transform* dan *validation*. *Transform* merupakan sebuah fungsi yang mengubah sebuah format menjadi format yang diinginkan seperti mengonversikan tipe data string ke integer. *Validation* merupakan sebuah fungsi untuk memvalidasi data yang dimasukkan [19]. Masukkan perintah pada Gambar 9 untuk menginstal *ValidationPipe.*

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

Gambar 9. Instal *ValidationPipe*

- *File upload*
  Pengembangan sistem membutuhkan untuk pengguna meng-*upload* bukti pembayaran dan 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 10 untuk menginstal multer atau *file upload.* Penginstal multer diperlukan agar sistem dapat dengan efektif mengelola proses pengunggahan *file*. *Library* multer juga dapat menangani *multi file upload.* Pengembang dapat mengatur folder penyimpanan *file*.

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

Gambar 10. Instal Multer

- Passport (Otentikasi)
  Passport merupakan library *authentication* NodeJS paling populer. Masukkan perintah pada Gambar 11 untuk menginstal Passport. *Authentication* proyek menggunakan *JSON Web Token* (JWT). *JSON Web Token* (JWT) merupakan token dengan tipe data string acak untuk otentikasi. JWT berguna sebagai identifikasi identitas pengguna dalam bentuk token [20]. Masukkan perintah pada Gambar 12 untuk menginstal *JSON Web Token* (JWT).

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

Gambar 11. Instal Passport

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

Gambar 12. Instal *JSON Web Token* (JWT)

- *Universally Unique Identifier* (UUID)
  *Universally Unique Identifier* (UUID) merupakan sejenis tipe data yang berguna sebagai *identifier* unik yang yang tidak akan sama dengan *identifier*

lainnya sebagai pengenal suatu entitas. Masukkan perintah pada Gambar 13 untuk menginstal UUID.

```
npm install uuid
```

Gambar 13. Intal UUID

- *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 [5]. Masukkan perintah pada Gambar 14 untuk menginstal fungsi *hash* pada *library* bcrypt.

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

Gambar 14. Instal bcrypt

- *Crypto*
  Library *crypto* pada pengembangan digunakan sebagai perhitungan untuk menghasil kode unik yang akan di-*generate* berbeda setiap transaksi dan setiap hari. Masukkan perintah pada Gambar 15 untuk menginstal library *crypto.*

```
npm install crypto dayjs
```

Gambar 15. Instal *crypto*

- *Nodemailer*
  Nodemailer merupakan *library* untuk mengirim email pada NestJS. Nodemailer dapat mengirim email melalui protokol *Simple Mail Transfer Protokol* (SMTP). Masukkan perintah pada Gambar 16 untuk menginstal Nodemailer

```
npm install nodemailer
```

Gambar 16. Instal nodemailer

- *OpenAPI*
  Pendokumentasian REST API pada sistem akan menggunakan *tools* Swagger. Masukkan perintah pada Gambar 17 untuk menginstal Swagger.

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

Gambar 17. Instal Swagger

- Menjalankan Aplikasi
  Setelah proses pengkodean (*coding*) selesai maka perlu menjalankan aplikasi. Masukkan perintah pada Gambar 18 untuk menjalankan aplikasi.

```
$ npm run start
```

Gambar 18. Menjalankan aplikasi

Tahapan dilanjutkan dengan membagi proyek menjadi beberapa modul, yaitu *user* modul, *authentication* modul, transaksi modul, *refund* modul, *wallet* modul, dan modul lainnya. *Role-Based Access Control* (RBAC) merupakan mekanisme kontrol akses berdasarkan hak dan peran. Peran pada sistem terdiri dari peran pengguna dan admin [19]. Setiap modul memiliki *guard* berupa *Role* Guard (RBAC) dan *JSON Web Token* (JWT).

- *User* modul membagi beberapa fitur menampilkan data pengguna, *register*, *update* akun, mengganti email, menghapus akun, dan konfirmasi email, mengganti *password*, lupa dan reset *password*.
- *Authentication* modul merupakan modul fitur sistem *login*, *logout*, dan *request* akses token baru.
- Transaksi modul membagi beberapa fitur sistem transaksi seperti menampilkan data transaksi, melakukan transaksi, *upload* bukti, menghapus transaksi, mengubah status transaksi.
- *Refund* modul membagi beberapa fitur sistem pengembalian dana seperti menampilkan data *refund*, mengajukan *refund*, dan mengganti status *refund,* menghapus *refund.*
- *Wallet* modul membagi beberapa fitur seperti membuat, membaca, merubah, dan menghapus data dompet digital.
- Modul lainnya berisi tambahan fitur sistem seperti email *sevice*, dan *universally unique identifier* (UUID).

Tahap 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 [19]. *Data Transfer Object* (DTO) merupakan suatu kelas yang berguna untuk mendefinisikan sumber data agar bisa dikirim ke *entity* atau basis data [19]. *Repository* merupakan suatu kelas yang berguna untuk merangkum atau memanipulasi sumber data berdasarkan kebutuhan dalam pengkodean (*coding*) [21].

### 3.4. Pengujian (*Testing*)

Tahap pengujian menggunakan metode pengujian *Blackbox Testing*. *Blackbox Testing* merupakan

pengujian perangkat lunak yang mengamati hasil serta fungsionalitas tanpa perlu mengetahui kode pemograman [22]. *Blackbox Testing* digunakan dalam pengujian terhadap fungsionalitas dan keluaran sistem tanpa perlu mengetahui kode pemograman [22]. REST API yang diuji dapat dilihat pada Tabel 2. Kolom *ID* pada Tabel 2 mengacu pada kolom *ID* Tabel 1 sehingga menyesuaikan dengan kebutuhan awal sistem. Metode dan fungsi pada Tabel 2 telah disesuaikan dengan kebutuhan sistem pada tahap perencanaan. *Endpoint* merupakan alamat identifikasi spesifik untuk dapat mengakses suatu layanan pada aplikasi/sistem. Semua API telah berhasil diuji dan diimplementasi dengan beberapa kondisi seperi *error handling* dan hak akses.

Tabel 2 Hasil Pengujian API

| ID | API | | | Hasil |
|---|---|---|---|---|
| | *Method* | 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-03.04 | 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/bayarGopay/bayarShopeepay/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/refundTolak/refundBerhasil)/:id | Berhasil |
| REQ-05.05 | DELETE | *Refund* | /refund/:id | Berhasil |
| REQ-06.01 | POST | Dompet digital | /(dana/ovo/gopay/shopeepay/linkAja)/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 |

Pengujian *Application Programming Interface* (API) dilakukan menggunakan aplikasi Postman dengan arsitektur REST API. REST API merupakan arsitektur yang menghasilkan keluaran dalam bentuk format *JavaScript Object Nation* (JSON).

Berikut pemaparan hasil *Application Programming Interface* (API) menggunakan aplikasi Postman:

- Transaksi


Gambar 19. *Response* API Transaksi

Pengguna yang berhasil melakukan transfer dompet digital dan telah dikirimkan dana oleh admin akan memberikan *response* transaksi berhasil. *Response* transaksi berhasil dapat dilihat pada Gambar 19. Kolom status transaksi akan berubah sesuai dengan kondisi transaksi apakah transaksi sedang diproses/menunggu, berhasil, dan gagal.

- *Refund*

Proses pengembalian dana baru dapat dibuat jika terdapat *id* transaksi dengan status transaksi gagal

atau berhasil tetapi memiliki kendala seperti dana kekirim dua kali. Pengembalian dana yang telah disetujui dan berhasil dikembalikan akan memberikan *response* yang dapat dilihat pada Gambar 20. Proses pengembalian dana yang berhasil juga akan menampilkan data transaksi dengan *id* transaksi yang terkait dengan pengembalian dana tersebut. Data transaksi yang terkait juga berguna agar administrasi dapat melihat apakah data permintaan *refund* dan data transaksi valid.

```
"id": "6f1d6341-9947-4a35-9322-9807808b8af6",
"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": "a19ae0da-0a07-4c0a-97d4-6fb5b1c2e93b-1701791572398.jpg",
"status_refund": "Dana terkirim",
"create_at": "2023-12-05T15:51:32.259Z",
"transaksi": {
    "id": "7c383503-5230-4cf9-a54d-c98fe0022737",
```


Gambar 20. *Response* API *Refund* Berhasil

## 4. Kesimpulan

Kesimpulan penelitian adalah arsitektur REST dapat digunakan untuk mengembangkan API agar dapat dikembangkan lebih lanjut oleh pengembang *frontend* dan *mobile*. Penelitian berhasil mengimplementasikan pemindahan saldo antar dompet digital. Semua API telah berhasil diimplementasi dan diuji secara berkala mulai kegunaan, *error handling*, dan lain sebagainya.

Saran untuk penelitian selanjutnya adalah menambahkan sistem keamanan seperti pin saat akan bertransaksi, atau identifikasi biometrik, serta menambahkan sistem pembayaran secara langsung dan instan seperti *payment gateaway* agar lebih mempermudah pengguna dalam bertransaksi.

## Reference


[1] D. Hendarsyah, "Penggunaan Uang Elektronik Dan Uang Virtual Sebagai Pengganti Uang Tunai Di Indonesia," *IQTISHADUNA J. Ilm. Ekon. Kita*, vol. 5, no. 1, pp. 1–15, 2016, doi: 10.46367/iqtishaduna.v5i1.74.

[2] A. Mulyana and H. Wijaya, "Perancangan E-Payment System pada E-Wallet Menggunakan Kode QR Berbasis Android," Komputika J. Sist. Komput., vol. 7, no. 2, pp. 63–69, 2018, doi: 10.34010/komputika.v7i2.1511.

[3] R. Fachrizal, "SurveySensum: 42 Persen Pengguna E-commerce Memiliki Loyalitas Rendah.pdf," INFOKOMPUTER.com, 2022. https://infokomputer.grid.id/read/123149401/surveysensum-42-persen-pengguna-e-commerce-memiliki-loyalitas-rendah?page=all (accessed Nov. 14, 2023).

[4] S. N. Yanti and E. Rihyanti, "Penerapan Rest API untuk Sistem Informasi Film Secara Daring," J. Inform. Univ. Pamulang, vol. 6, no. 1, p. 195, 2021, doi: 10.32493/informatika.v6i1.10033.

[5] P. S. Saputra and L. P. A. S. Tjahyanti, "Pemanfaatan Teknologi Informasi Menggunakan Web API di Masa Pandemi Covid-19," KOMTEKS, vol. 1, no. 1, 2022.

[6] I. Mahendra and D. T. Eby Yanto, "Sistem Informasi Pengajuan Kredit Berbasis Web Menggunakan Agile Development Methods pada Bank Bri Unit Kolonel Sugiono," J. Teknol. dan Open Source, vol. 1, no. 2, pp. 13–24, 2018, doi: 10.36378/jtos.v1i2.20.

[7] T. Gumelar, R. Astuti, and A. T. Sunarni, "Sistem Penjualan Online Dengan Metode Extreme Programming," Telemat. Mkom, vol. 9, no. 2, pp. 87–90, 2018, doi: https://dx.doi.org/10.36080/telematikamkom.531.

[8] R. Sahrial, "Rancang Bangun Sistem Informasi Zakat Infaq Shodaqoh Menggunakan Metodologi Extreme Programming," J. Buana Inform., pp. 31–42, 2018, doi: https://doi.org/10.24002/jbi.v9i1.1666.

[9] A. Wahid Abdul, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," J. Ilmu-ilmu Inform. dan Manaj. STMIK, no. November, pp. 1–5, 2020.

[10] I. G. N. Suryantara, Merancang Aplikasi Dengan Metodologi Extreme Programmings, no. March. Jakarta: Elex Media Komputindo, 2017.

[11] M. Melinda, R. I. Borman, and E. R. Susanto, "Rancang Bangun Sistem Informasi Publik Berbasis Web (Studi Kasus : Desa Durian Kecamatan Padang Cermin Kabupaten Pesawaran)," J. Tekno Kompak, vol. 11, no. 1, p. 1, 2018, doi: 10.33365/jtk.v11i1.63.

[12] A. M. Saepuloh and S. Ginting, "PERANCANGAN SISTEM INFORMASI MANAJEMEN PROYEK DENGAN MENGGUNAKAN SOFTWARE NEST. JS BERBASIS WEB DI PT. MITRA PAJAKKU," INFOKOM (Informatika & Komputer), vol. 10, no. 1, pp. 1–9, 2022, doi: https://doi.org/10.56689/infokom.v10i1.818.

[13] Y. Heriyanto, "PERANCANGAN SISTEM INFORMASI RENTAL MOBIL BERBASIS WEB PADA PT.APM RENT CAR," J. Intra-Tech, vol. 2, no. 2, 2018, doi: 10.61220/voice.v1i1.20232.

[14] L. P. Dewi, U. Indahyanti, and Y. H. S, "Pemodelan Proses Bisnis Menggunakan Activity Diagram Uml Dan Bpmn ( Studi Kasus Frs Online )," Informatika, pp. 1–9, 2017.

[15] E. Doro and B. Stevalin, "Analisis Data dengan Menggunakan ERD dan Model Konseptual Data Warehouse," J. Inform., vol. 5, no. 1, pp. 71–85, 2012.

[16] A. Amijaya, F. Ferdinandus, and M. Bayu, "Sistem Pendukung Keputusan Pemilihan Handphone Dengan Metode Simple Additive Weighting Berbasis WEB," CAHAYAtech, vol. 8, no. 2, p. 102, 2019, doi: 10.47047/ct.v8i2.47.

[17] W. Mualim and G. U. Putra, "Implementasi Framework MVC Pada Sistem Informasi Akademik Di STMIK Yadika Bangil," J. SPIRIT, vol. 9, no. 2, pp. 35–39, 2017, doi: http://dx.doi.org/10.53567/spirit.v9i2.83.

[18] A. Haniefardy, M. B. A. Fadhillah, and S. Rochimah, "Tinjauan Literatur Sistematis: Pengaruh Penggunaan Framework Khusus dalam Proses Pengembangan Web dan Pembuatan Web," Matrix J. Manaj. Teknol. dan Inform., vol. 9, no. 2, pp. 68–73, 2019, doi: 10.31940/matrix.v9i2.1161.

[19] NetsJS Documentation, "NestJS Documentation," 2023. https://docs.nestjs.com/ (accessed Sep. 23, 2023).

[20] M. B. Jones, J. Bradley, and N. Sakimura, "JSON Web Signature (JWS)," RFC, vol. 7515, pp. 1–59, 2015, [Online]. Available: https://api.semanticscholar.org/CorpusID:40079086.

[21] R. Benita, "Implementing a Generic Repository Pattern Using NestJS.pdf," Medium.com, pp. 1–22, 2022, [Online]. Available: https://betterprogramming.pub/implementing-a-generic-repository-pattern-using-nestjs-fb4db1b61cce.

[22] H. Nurfauziah and I. Jamaliyah, "Perbandingan Metode Testing Antara Blackbox Dengan Whitebox Pada Sebuah Sistem Informasi," *J. Vis.*, vol. 8, no. 2, pp. 105–113, 2022, [Online]. Available: https://jurnas.saintekmu.ac.id/index.php/visualika/article/view/24.