PENGEMBANGAN FRONT-END PADA APLIKASI M-BANKING AGEN46 DENGAN TEKNOLOGI FLUTTER

Disusun Oleh:

N a m a : Farah Dila Anastasia

NIM : 18523274

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN FRONT-END PADA APLIKASI M-BANKING AGEN46 DENGAN TEKNOLOGI FLUTTER

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Farah Dila Anastasia

NIM : 18523274

Yogyakarta, 3 Juli 2022

Pembimbing,

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

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN FRONT-END PADA APLIKASI M-BANKING AGEN46 DENGAN TEKNOLOGI FLUTTER

TUGAS AKHIR JALUR MAGANG

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, 18 Juli 2022

Tim Penguji

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

Anggota 1

Rian Adam Rajagede, S.Kom., M.Cs. _____

Anggota 2

Sri Mulyati, S.Kom., M.Kom. _____

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

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Farah Dila Anastasia

NIM : 18523274

Tugas akhir dengan judul:

PENGEMBANGAN FRONT-END PADA APLIKASI M-BANKING AGEN46 DENGAN TEKNOLOGI FLUTTER

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, 3 Juli 2022

(Farah Dila Anastasia)

HALAMAN PERSEMBAHAN

Assalamu'alaikum Warahmatullahi Wabarakatuh *Alhamdulilahirobbil'alamin,* puji syukur kepada Allah SWT berkat izin, karunia serta rahmat-Nya yang telah memberikan kesehatan, kemudahan, kelancaran serta kemudahan selama proses pembuatan hingga penyelesaian Tugas Akhir ini tepat pada waktunya. Semoga keberhasilan ini menjadi satu langkah awal untuk menuju masa depanku dalam meraih cita-cita serta menjadi manusia yang dapat bermafaat bagi orang lain dan masyarakat, Allahumma Aamiin.

Dengan ini saya persembahkan karya ini untuk,

Ibu dan Bapak tercinta,

Terima kasih untuk kedua orang tua saya, Ibu Dewi Ratna Sari dan Bapak Joko Budiarto yang telah memberikan cinta dan kasih nya kepada saya, proses yang telah saya lewati tidak akan berhasil jika tanpa dukungan baik secara moral maupun materiil dari bapak dan ibu. Semoga ilmu yang saya dapatkan selama menimba ilmu di UII dapat bermanfaat bagi kehidupan saya selanjutnya.

Kakak saya yang luar biasa,

Terima kasih untuk kakak saya Mba Eka, Mas Dwi, Mas Yudhistira, Mas Gana, Mas Habib yang selalu memberikan bimbingan, nasehat, dorongan, dan semnagat yang diberikan kepada saya selaku adik terakhir selama menempuh pendidikan di Univeristas Islam Indonesia.

Dosen Pembimbing,

Terima kasih sebanyak-banyaknya untuk dosen pembimbing saya, Bapak Irving Vitra Paputungan, S.T, M.Sc., Ph.D. yang selalu sabar dan meluangkan waktunya dalam membimbing saya dalam pengerjaan Tugas Akhir ini.

Dosen Informatika UII,

Terima kasih untuk semua dosen Informatika Universitas Islam Indonesia yang telah memberikan seluruh ilmu, waktu, tenaga, pikiran, serta pengalaman sehingga saya dapat melalui semua tahapan dengan lancar. Semoga ilmu yang Bapak/Ibu Dosen berikan dapat menjadi bekal saya serta dapat saya terapkan di perjalanan saya selanjutnya.

Sahabat seperjuangan,

Untuk sahabat seperjuangan saya Fira, Dhea dan Tia terima kasih atas segala dukungan, masukan, semangat yang selalu diberikan saya selama melewati masa senang maupun susah. Terima kasih telah menemani perjuangan saya baik saat magang hingga penyelesaian Tugas Akhir ini. Akhir kata dari saya, **Wassalamu'alaikum Warahmatullahi Wabarakatu.**

HALAMAN MOTO

فَإِنَّ مَعَ الْعُسْرِ يُسْرًا إِنَّ مَعَ الْعُسْرِ يُسْرًا

“Karena sesungguhnya sesudah kesulitan itu ada kemudahan. Sesungguhnya sesudah kesulitan itu ada kemudahan”

Q.S Al Insyirah: 5-6

“Ilmu pengetahuan tanpa agama adalah pincang, agama tanpa ilmu adalah buta”

Albert Einstein

“Resiko yang paling besar adalah tidak mengambil resiko. Dalam dunia yang berubah dengan cepat, strategi yang pasti akan gagal adalah tidak mengambil resiko”

Mark Zuckerberg

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah dihaturkan kepada Allah Swt., yang telah melimpahkan rahmat dan taufik serta hidayah-Nya hingga dapat menyelesaikan Tugas Akhir yang berjudul "PENGEMBANGAN FRONT-END PADA APLIKASI M-BANKING AGEN46 DENGAN TEKNOLOGI FLUTTER". Serta shalawat dan salam diucapkan kepada Nabi Muhammad Sallallahu Alaihi Wasallam, yang telah membawa umat Islam dari zaman kebodohan menuju zaman yang terang benderang dengan Islam dan ilmu pengetahuan. Puji syukur yang sebesar-besarnya selalu dihaturkan kepada Allah terutama atas berkah dan izin-Nya, proses kuliah hingga Tugas Akhir dapat terlaksana dengan baik sebagai media pembelajaran terhadap ilmu dan ciptaan-Nya.

Adapun dari berlangsungnya kegiatan magang di PT. Bank Negara Indonesia Tbk, kemudian laporan Tugas Akhir ini disusun untuk memenuhi persyaratan tugas akhir jalur magang di Fakultas Teknologi Industri Jurusan Informatika Universitas Islam Indonesia. Dalam penyusunan tugas akhir ini, tidak lepas dari arahan dan bimbingan berbagai pihak. Tidak lupa mengucapkan rasa hormat dan terima kasih kepada semua pihak yang telah membantu. Pihak-pihak yang terkait diantaranya sebagai berikut:

1. Allah Subhanahu Wata'ala yang telah memberikan kehidupan, keselamatan, kesehatan, jasmani dan rohani serta kesempatan untuk dapat melaksanakan magang dengan baik.
2. Kedua orang tua tercinta yang selalu mendukung dan memberikan doa, bantuan positif baik secara moral maupun material dalam kegiatan magang ini.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D. selaku Dosen Pembimbing yang telah bersedia membimbing dan mengarahkan penyusunan laporan ini.
5. Bapak Taufikurrahman, VP Customer Banking Channel, Divisi Pengembangan Digital (DGL) PT. Bank Negara Indonesia Tbk selaku pembimbing yang telah menerima dengan baik.
6. Ibu Dian Amalia Pongdatu dan tim pengembang di kelompok Branchless Banking Channel (BBC), Divisi Pengembangan Digital (DGL) PT. Bank Negara Indonesia Tbk.

7. Segenap karyawan dan karyawati PT. Bank Negara Indonesia Tbk yang telah tulus menerima, memberi pengarahan dan membantu selama melaksanakan magang di perusahaan tersebut.

Laporan Tugas Akhir ini telah dibuat dengan usaha terbaik, tetapi masih jauh dari kata sempurna. Sehingga diperlukan saran dan kritik yang membangun dari pembaca untuk penyempurnaan laporan Tugas Akhir ini. Akhir kata, dengan disusunnya Tugas Akhir ini diharapkan dapat memberikan manfaat bagi semua pihak

Wassalamu'alaikum Wr. Wb.

Purwokerto, 2 Juli 2022

Farah Dila Anastasia

SARI

PT. Bank Negara Indonesia Tbk (BNI) merupakan perusahaan yang bergerak di bidang layanan finansial yang saat ini sedang mengembangkan aplikasi Agen46 untuk layanan *m-banking* berbasis *multiplatform*. Salah satu fasilitas yang dikembangkan adalah layanan pembayaran non tunai berupa *top up* uang elektronik (*e-money*). *Framework* yang digunakan adalah *Flutter*. *Framework* ini dapat membuat aplikasi *mobile* Android maupun iOS hanya dengan satu basis *coding* (*codebase*) dalam sekali waktu. Bahasa pemrograman yang digunakan adalah Dart. Pola desain yang diterapkan adalah *BLoC Pattern*. *BLoC Pattern* memudahkan *developer* untuk fokus dalam melakukan konversi *event* menjadi *state* sehingga jika implementasi diubah, maka *developer* tidak perlu lagi melakukan banyak perubahan pada *code* lainnya. *Code* yang dihasilkan menjadi lebih mudah dimengerti dan *reusable* pada pengembangan selanjutnya sehingga dapat meminimalisir *resource*. Metode pengembangan yang digunakan *adalah Agile Scrum* dengan beberapa tahapan dari mulai pendefinisan, inisialisasi, perencanaan, pelaksanaan, pengendalian & evaluasi hingga penutupan proyek. Berdasarkan hasil yang didapat, diketahui bahwa implementasi *BLoC Pattern* pada pengembangan *frontend* dapat mempermudah *developer* mengembangkan aplikasi *mobile banking* berbasis *multiplatform*. Terbukti dengan pengembangan dan pengujian yang telah dilakukan menghasilkan fitur yang dapat berjalan dalam dua sistem operasi yaitu Android maupun iOS serta fungsionalitas dari fitur tersebut dapat berjalan dengan baik.

Kata kunci: *m-banking*, *top up*, *e-money*, *BLoC*, *Flutter*.

GLOSARIUM

Widget	Komponen-komponen yang membentuk tampilan pada *Flutter* seperti *button, text,* gambar dan sebagainya.
Flutter	*Platform* buatan Google yang digunakan oleh pengembang untuk membuat aplikasi berbasis *multiplatform* hanya dengan satu basis *coding*.
Property	Komponen pengatur *widget* pada *Flutter.*
Agile Scrum	Metode pengembangan perangkat lunak.
E-money	Alat pembayaran berbentuk elektronik atau secara digital.
BLoC	*Business Logic Component* yang digunakan sebagai desain *pattern* pengembangan aplikasi pada *Flutter*.
SDK	*Software Deelopment Kit* atau sekumpulan alat yang digunakan untuk membantu membuat aplikasi.
Compile	Program yang digunakan untuk menerjemahkan *code* ke dalam bahasa pemrograman lain.
Hit	Proses pengambilan data pada API.
Developer	Tim pembuat aplikasi.
Request	Proses meminta data kepada *server.*
Response	Proses mengirim data dari *server.*
API	Penghubung antar aplikasi *multiplatform* maupun *cross platform*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada era globalisasi zaman ini pertumbuhan teknologi sangat pesat dengan munculnya berbagai inovasi. Adanya teknologi telah membawa dampak positif bagi masyarakat terutama dalam penggunaan internet. Penggunaan internet dinilai dapat membantu aktivitas sehari-hari sehingga menjadi kebutuhan primer bagi masyarakat. Perkembangan internet juga berpengaruh pada sektor ekonomi khususnya dalam industri perbankan. Bank menjadi salah satu sektor penting dalam menjaga kestabilan ekonomi di Indonesia. Dalam menjalankan tugasnya bank berkewajiban memberikan dan menjamin layanan terbaik kepada nasabahnya. Layanan yang disediakan oleh bank tentu harus bisa menyesuaikan dengan mobilitas konsumennya sehingga diperlukan pemanfaatan teknologi yang tepat untuk mendukung perkembangan bisnis. Penerapan teknologi yang tepat juga perlu memperhatikan minimum biaya yang dikeluarkan untuk meminimalisir operasional agar bank dapat beroperasi secara lebih optimal.

Mobile banking merupakan inovasi terbaru untuk diterapkan di industri perbankan (internet banking). Mobile banking memungkinkan nasabah bank untuk mengakses layanan perbankan dengan menggunakan ponsel mereka untuk melakukan fungsi seperti memeriksa saldo rekening mereka, mengajukan permohonan kartu debit, melakukan transaksi pembayaran, membayar tagihan elektronik, dan lain-lain. Mobile banking berguna untuk berkomunikasi dengan bank melalui perangkat mobile, seperti ponsel atau perangkat elektronik untuk menawarkan layanan mobile banking kepada nasabahnya. Layanan ini memberikan keuntungan bagi pihak nasabah maupun bank. Nasabah akan mendapatkan informasi secara efisien, efektif, dan cepat dalam bertransaksi sehingga mengurangi waktu tunggu jika dibandingkan dengan transaksi konvensional di bank secara langsung. Layanan *m-banking* dapat menjadi peluang bagi bank untuk menawarkan nilai tambah kepada nasabah. Keefektifan dan keefisienan penggunaan layanan *m-banking* tidak terlepas dari dukungan telepon seluler dan internet. Pengguna platform layanan mobile banking dapat memperoleh informasi tentang rekening mereka secara real time dan melakukan pembayaran untuk transaksi kapanpun dan dimanapun mereka pilih.

Adanya peningkatan kebutuhan transaksi masyarakat di era globalisasi saat ini tentu berdampak pada peningkatan penggunaan uang elektronik.

Hal ini dilandasi sejak munculnya aturan yang dikeluarkan oleh Bank Indonesia dalam peraturan No. 26/6/PBI/2018 mengenai uang elektronik. Pada Pasal 1 ayat 3 dijelaskan bahwa uang elektronik adalah alat pembayaran yang diterbitkan atas dasar nilai uang yang disetor terlebih dahulu oleh pemegang kepada penerbit. Sistem pembayaran jumlah atau nilai uangnya diserahkan terlebih dahulu untuk melakukan pengisian nominal uang (*top up*) kepada penerbit baik melalui agen penerbit atau secara langsung. *E-money* merupakan salah satu inovasi sistem pembayaran yang muncul dari dampak perkembangan teknologi. E-money adalah alat yang digunakan untuk transaksi non tunai yang dilakukan melalui sistem elektronik.

Penggunaan uang elektronik memiliki kelebihan jika dibandingkan dengan uang tunai. Pengguna tidak perlu membawa banyak uang tunai untuk melakukan pembayaran. Transaksi yang dilakukan juga lebih cepat dan efisien. Namun *e-money* juga memiliki kekurangan yaitu masyarakat belum sepenuhnya menggunakan *e-money* sebagai sistem pembayaran. Industri perbankan dan telekomunikasi juga berlomba-lomba mengeluarkan produk *e-money*. Tindakan ini juga didukung oleh para pelaku *fintech startup* di Indonesia yang bergerak di sektor pembayaran seperti DANA, LinkAja, GoPay, ShopeePay, iSaku dan masih banyak lainnya. *E-money* dapat digunakan untuk melakukan transaksi pembayaran di internet maupun pada *merchant* yang telah bekerjasama dengan bank penerbit *e-money*.

BNI merupakan salah satu Bank BUMN (Badan Usaha Milik Negara) yang bergerak pada bidang layangan finansial. BNI menyediakan layanan penyimpanan dana maupun fasilitas pinjaman baik pada segmen koperasi, menengah maupun kecil. Beberapa produk dan layanan terbaik disesuaikan dengan kebutuhan nasabah sejak kecil, remaja, dewasa, hingga pensiun. BNI melayani layanan perbankan personal dimulai dari simpanan, pinjaman, hingga layanan kartu kredit. Adapun beberapa layanan dan produk *e-banking* milik BNI seperti BNI ATM, BNI SMS Banking, BNI Internet Banking, BNI Phone Banking, BNI Mobile Banking, Layanan Gerak, BNI Agen46, Tapcash, BNI Debit Online, BNI SMS Notifikasi, BNI IPay, BNI Smartpay, EDC BNI, Buka Rekening Tabungan Digital, QRIS, dan Internet Payment Gateway. Tidak hanya itu BNI juga melayani solusi pendanaan untuk perbankan bisnis dari tingkat UKM hingga transaksi bisnis internasional. Dalam menjalankan tugasnya BNI didukung oleh sejumlah anak perusahaan yakni BNI Syariah, BNI Multifinance, BNI Sekuritas, BNI Life Insurance dan BNI Remittance (BNI, 1992a).

BNI Agen46 adalah salah satu layanan yang tersedia di BNI. BNI Agen46 merupakan mitra BNI (perorangan atau badan hukum yang bekerjasama dengan BNI) untuk menyediakan layanan perbankan kepada masyarakat (Layanan Laku Pandai, LKD, dan *e-Payment*).

Meningkatnya kebutuhan sistem transaksi secara digital BNI membutuhkan pengembangan sistem yang baru pada aplikasi *m-banking* nya (BNI, 1992b). Salah satu proyek yang dikembangkan adalah *m-banking* Agen46 berbasis *multiplatform*. Pengembangan yang dilakukan tentu harus memperhatikan baya yang dikeluarkan untuk proses pemeliharaan sistem yang berjalan. Hal ini bertujun agar sistem dapat memenuhi kebutuhan penggunanya dan menjangku *user* yang lebih banyak lagi. Proyek Agen46 merupakan proyek yang masih berjalan pada masa pengembangan. Pengembangan dibuat dengan menerapkan desain *interface* yang baru dari aplikasi sebelumnya sehingga terdapat perubahan yang cukup besar dari segi tampilan. Pembuatan desain ulang aplikasi ini diharapkan dapat menjadi aplikasi yang lebih mudah dipahami pengguna dan menghasilkan tampilan aplikasi yang lebih menarik.

Perbedaan sistem operasi setiap pengguna mendorong BNI untuk memilih *framework* yang dapat membantu pengembangan secara efisien dari segi biaya pengembangan. *Framework* yang digunakan adalah *Flutter*. Penggunaan framework ini bertujuan agar aplikasi *mobile banking* nantinya dapat berjalan pada dua sistem operasi baik iOS maupun Android untuk layanan keagenan. Layanan keagenan yang dikembangkan yaitu fitur *top up* uang elektronik (*e-money*). Pengembangan aplikasi *m-banking* Agen46 pada penelitian ini lebih fokus pada pengembangan *frontend* aplikasi dan pengintegrasi API. Salah satu fitur *top up e-money* yang dibahas pada penlitian ini adalah LinkAja. LinkAja merupakan salah satu penyedia jasa layanan uang elektronik yang bekerjasama dengan BNI.

Aplikasi *m-banking* Agen46 dikembangan dengan menggunakan *framework Flutter*. Dilihat dari sejarahnya bahwa *Flutter* merupakan *framework* yang resmi diluncurkan oleh Google pada bulan Desember 2018. Diketahui bahwa *framework Flutter* merupakan *platform* yang digunakan oleh para pengembang untuk membangun aplikasi *multiplatform* (iOS dan Android) hanya dengan satu basis *coding* saja. Bahasa yang digunakan dalam *framework Flutter* adalah Dart. Dart merupakan bahasa pemrograman yang berorientasi objek (OOP) yang dikembangkan oleh Google sejak tahun 2007. *Syntax* yang dimiliki oleh Dart memiliki kemiripan dengan C++, Java dan Javascript. Penggunaan Dart menjadikan *Flutter* sebagai *cross platform framework* tercepat yang memiliki performa seperti native. Dengan kemampuan tersebutlah yang mendukung pengembangan aplikasi *mobile* secara native untuk kedua *platform* Android dan iOS. Konsep pengembangan aplikasi yang digunakan pada *m-banking* adalah *BLoC Pattern*. BLoC (*Business Logic Component*) adalah desain *pattern* yang membantu untuk memisahkan *presentation* dengan *business logic*.

BLoC Pattern memudahkan *developer* untuk fokus dalam mengkonversi *event* menjadi *state* sehingga saat implementasi dirubah *developer* tidak perlu lagi melakukan banyak perubahan pada *code* lainnya. *Code* yang dibuat menjadi lebih mudah dimengerti dan *reusable* pada pengembangan sistem operasi yang lain sehingga *resource* dikeluarkan untuk *development* Android dan iOS menjadi lebih minim.

1.2 Ruang Lingkup Magang

Penempatan magang berada di Divisi Pengembangan Digital (DGL) Area 1 yang dipimpin oleh Bapak Taufikurrahman. Pada area 1 Divisi Pengembangan Digital (DGL) memiliki beberapa proyek yang akan dikembangkan yaitu e-Form Prakerja, Mobile Remittance, Portal EDM, IBank Remit, dan Agen46. Anggota magang Divisi Pengembangan Digital (DGL) area 1 berjumlah 11 orang dengan masing-masing pembagian sesuai dengan pengerjaan proyek yang akan dikembangkan yaitu e-Form Prakerja (1 orang), Mobile Remittance (2 orang), Portal EDM (3 orang), IBank Remit (2 orang), dan Agen46 (3 orang).

PT. Bank Negara Indonesia Tbk memiliki budaya kerja yakni prinsip yang diyakini dan menjadi landasan berbagai kebijakan pengelolaan sumber daya manusia (SDM) serta menjadi panduan perilaku untuk setiap insan PT. Bank Negara Indonesia Tbk. Budaya kerja ini dikenal dengan PRINSIP 46. Seluruh insan PT. Bank Negara Indonesia Tbk mulai dari jajaran komisaris, direksi, hingga pegawai termasuk pegawai rekanan di PT. Bank Negara Indonesia Tbk wajib mematuhi perilaku dan tata nilai budaya kerja ini.

PRINSIP 46 terdiri dari empat nilai utama dan enam perilaku utama insan BNI yang diharapkan dapat dipahami sebagai pokok dasar berpikir dan bersikap setiap insan BNI dalam setiap aktivitas pekerjaannya.

a. 4 Nilai Budaya Kerja BNI :
    1. Profesionalisme.
    2. Integritas.
    3. Orientasi Pelanggan.
    4. Perbaikan Tiada Henti.
b. 6 Perilaku Utama Insan BNI :
    1. Meningkatkan kompetensi dan memberikan hasil terbaik.
    2. Jujur, tulus, dan ikhlas.
    3. Disiplin, konsisten, dan bertanggung jawab.
    4. Memberikan layanan terbaik melalui kemitraan yang sinergis.

5. Senantiasa melakukan penyempurnaan
6. Kreatif dan inovatif.

Pola kerja di PT. Bank Negara Indonesia Tbk yang membuat nyaman. Segala pekerjaan terorganisasi dan ter-*manage* dengan sangat baik, mulai dari dokumentasi proyek yang sangat lengkap dan segala *tools-tools* internal milik PT. Bank Negara Indonesia Tbk serta akses yang diberikan kepada peserta selama kegiatan magang berlangsung.

Penugasan magang berada pada salah satu divisi yaitu Divisi Pengembangan Digital dan berada di dalam tim *Branchless Banking Channel* (BBC). *Branchless Banking Channel* (BBC) merupakan salah satu kelompok yang fokus pada pengembangan *branchless channel* untuk layanan *branchless* digital PT. Bank Negara Indonesia Tbk melalui ekosistem keagenan. Tim *Branchless Banking Channel* (BBC) disupervisi langsung oleh General Manager (GM) dan Deputi General Manager (DGM) Divisi Digital Pengembangan (DGL). Struktur organisasi tim *Branchless Banking Channel* (BBC) dapat dilihat pada Gambar 1.1. *Branchless Banking Channel* (BBC) merupakan salah satu kelompok yang fokus pada pengembangan *branchless channel* untuk layanan *branchless* digital PT. Bank Negara Indonesia Tbk melalui ekosistem keagenan. Tim *Branchless Banking Channel* (BBC) disupervisi langsung oleh General Manager (GM) dan Deputi General Manager (DGM) Divisi Digital Pengembangan (DGL). Divisi Pengembangan Digital dipimpin oleh Direktur IT dan Operasi. Tim *Branchless Banking Channel* diketuai oleh Ibu Dian Amalia Pongdatu dengan jumlah anggota sebanyak 14 orang (11 karyawan dan 3 magang).

Gambar 1.1 Struktur Organisasi BBC

Salah satu proyek yang dikembangkan oleh tim *Branchless Banking Channel* adalah Agen46. Dalam magang ini bertugas sebagai *frontend engineer* untuk membantu pengembangan aplikasi *mobile banking* Agen46 dengan menggunakan *Flutter*. Pelaksanaan magang berlangsung selama 6 bulan yaitu dimulai dari tanggal 1 September 2021 hingga tanggal 28 Februari 2022. Adapun aktivitas yang dilakukan selama melakukan magang diantaranya:

a. Mengikuti orientasi serta pengarahan dari perusahaan
b. Mengikuti pelatihan satu bulan terkait Java Fundamental, Git, Database JDBC, HTTP, REST API, Spring Boot, REST Controller, Bean Stereotype, Apache Kafka, Docker, React JS, Flutter, dan Scrum Agile.
c. Mempersiapkan *software* dan teknologi perusahaan.
d. Mempelajari *frontend mobile* dengan *Flutter*.
e. Mengerjakan tampilan fitur *top up* uang digital LinkAja *mobile banking* Agen46.
f. Mengerjakan integrasi API fitur *top up* uang digital LinkAja *mobile banking* Agen46.
g. Melakukan *deployment* sandboxing fitur *top up* uang digital LinkAja pada *mobile banking* Agen46 ke DigiLabs PT. Bank Negara Indonesia Tbk.
h. Membuat dokumentasi pengembangan fitur *top up* uang digital LinkAja pada *mobile banking* Agen46.
i. Mengerjakan fitur pembayaran tiket transportasi KAI pada *mobile banking* Agen46.
j. Membuat dokumentasi pengembangan fitur pembayaran tiket transportasi KAI pada *mobile banking* Agen46.
k. Mengerjakan fitur *top up* uang digital DANA pada *mobile banking* Agen46.
l. Membuat dokumentasi pengembangan fitur *top up* uang digital DANA pada *mobile banking* Agen46.
m. Mengerjakan fitur *top up* uang digital Tapcash pada *mobile banking* Agen46.
n. Membuat dokumentasi pengembangan fitur *top up* uang digital Tapcash *mobile banking* Agen46.
o. Diskusi menggunakan *platform* Zoom dan Whatsapp dengan ketua tim proyek.
p. Diskusi menggunakan *platform* Zoom dengan mentor.
q. Menulis *logbook* aktivitas magang.

1.3 Tujuan

Tujuan dari pengembangan *frontend* fitur *top up e-money* LinkAja, DANA, Tapcash, dan pembayaran tiket KAI pada aplikasi m-banking Agen46 di PT. Bank Negara Indonesia Tbk adalah sebagai berikut:

a. Membuat *frontend* fitur *top up e-money* LinkAja, DANA, TapCash, dan pembayaran tiket KAI berbasis *multiplatform* dengan menggunakan teknologi Flutter.
b. Membuat integrasi API pada pengembangan *frontend* fitur *top up e-money* LinkAJa, DANA, TapCash dan pembayaran tiket KAI pada *m-banking* Agen46.
c. Mengetahui implementasi *BLoC Pattern* pada pengembangan *frontend* aplikasi *m-banking* Agen46 BNI.

1.4 Manfaat

Manfaat dari pengembangan *frontend* fitur *top up e-money* LinkAja, DANA, Tapcash, dan pembayaran tiket KAI pada aplikasi m-banking Agen46 di PT. Bank Negara Indonesia Tbk adalah sebagai berikut:

- Fitur *top up e-money* LinkAja, DANA, TapCash, dan pembayaran tiket KAI dengan teknologi Flutter dapat digunakan untuk memudahkan developer dalam mengembangkan aplikasi berbasis *multiplatform*.
- Penggunaan implementasi *BLoC Pattern* dapat memudahkan *developer* dalam membuat integrasi API pada pengembangan *frontend m-banking* Agen46 BNI.
- Pengembangan *frontend* dengan menggunakan *BLoC Pattern* dapat meningkatkan performa aplikasi yang baru dari aplikasi sebelumnya.

1.5 Sistematika Penulisan

Sistematika Penulisan disusun untuk memberikan gambaran umum tentang laporan akhir yang dibuat. Sistematika Penulisannya sebagai berikut:

a. BAB I: PENDAHULUAN

   Bab ini berisi tentang latar belakang, ruang lingkup magang tujuan dan manfaat penelitian, dan sistematika penulisan.

b. BAB II: DASAR TEORI

   Bab ini berisi teori berupa pengertian dan definisi yang diambil dari kutipan buku dan media internet yang berkaitan dengan penyusunan laporan.

c. BAB III: PELAKSANAAN MAGANG

Bab ini berisi dokumentasi dari kegiatan dan tugas yang dikerjakan selama mengikuti proses kegiatan magang.

d. BAB IV: REFLEKSI PELAKSANAAN MAGANG

Bab ini menjelaskan hasil dari pengembangan *frontend* fitur *top up e-money* LinkAja, DANA, TapCash dan pembayaran tiket KAI pada aplikasi *m-banking* Agen46 dengan Flutter di PT. Bank Negara Indonesia Tbk.

e. BAB V: KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran yang berkaitan berdasarkan yang telah dijelaskan pada bab-bab sebelumnya.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 *Mobile Banking*

Mobile banking atau *m-banking* merupakan fasilitas yang memudahkan nasabah untuk melakukan transaksi secara online. Kegiatan transaksi yang biasanya dilakukan dengan cara mendatangi ke bank secara langsung kini dapat dilakukan hanya dengan menggunakan *handphone* saja. Hal ini memungkinkan nasabah untuk dapat melakukan transaksi dimana saja dan kapan saja. Penggunaan *m-banking* dimulai dengan melakukan pendaftaran akun terlebih dahulu kemudian nasabah dapat menggunakan untuk melakukan transaksi dengan cara memasukan *user* ID atau *password* dan PIN untuk menyelesaikan transaksi. Nasabah dapat menghemat waktu serta biaya. Selain itu pemanfaatan *handphone* yang biasanya hanya digunakan sebagai media komunikasi juga bisa dimanfaatkan untuk media berbisnis dan bertransaksi. Layanan yang tersedia pada *m-banking* memberikan kemudahan kepada nasabah untuk melakukan transaksi seperti cek saldo, transfer antar rekening, pembayaran tagihan, isi pulsa, dan lain-lain. Selain itu di dalam *m-banking* juga terdapat laporan atau ringkasan terkait aktivitas keuangan yang dilakukan oleh nasabah. Pada aplikasi *m-banking* juga terdapat layanan pembelian dan *top up* dompet digital seperti OVO, Gopay, hingga LinkAja. Layanan - layanan *m-banking* tersebut tentunya memiliki kelebihan selain efisien dan mudah diakses, nasabah juga dapat melakukan jadwal transaksi keuangan dalam satu waktu sekaligus. Namun selain banyak kelebihan m-banking juga memiliki kekurangan yaitu rentan dengan pencurian data dan membutuhkan jaringan yang bagus untuk dapat mengaksesnya (Ismail Abdelfattah, Awad, & Nasr, 2020).

Pada aplikasi *m-banking* Agen46 tersedia beberapa layanan yaitu layanan laku pandai, layanan LKD, layanan *e-payment*. Pada layanan laku pandai terdiri dari fitur buka rekening tabungan BNI Pandai, setor tunai, dan tarik tunai. Layanan keuangan digital (LKD) terdiri dari fitur pendaftaran (*register*), setor tunai (*cash in*), dan tarik tunai (*cash out*) uang elektronik. Layanan *e-payment* terdiri dari fitur transfer (antar BNI dan online antar bank), pembelian dan pembayaran. Pada fitur pembelian terbagi menjadi beberapa fitur lagi yaitu *top up* pulsa & paket data, dan token listrik. Pada fitur pembayaran terdiri dari fitur tagihan listrik, PDAM, kartu kredit, telepon, *multifinance*, BPJS, asuransi, pembayaran tiket dan lainnya. Fitur yang dikembangankan yaitu fitur *top up* uang elektronik dan fitur pembayaran tiket kereta.

2.1.1 *Top up e-money*

Top up adalah sebuah kegiatan untuk melakukan pengisian atau penambahan sejumlah dana ke rekening agar dapat melakukan transaksi seperti pembayaran maupun pembelian. Terdapat juga arti lain dari top up yaitu sebagai pemberian pinjaman dari bank dengan tujuan untuk meningkatkan kredit dari jumlah peminjam. Fasilitas ini diberikan oleh bank agar dapat berhubungan langsung dengan riwayat transaksi nasabah. Tentunya setiap bank memiliki kriteria dan syarat tertentu dalam pemberian fasilitas *top up*.Kegiatan pengisian ulang saldo bisa dilakukan dengan berbagai metode seperti melalui minimarket, ATM Bank, *m-banking* atau gerai yang melayani pengisian ulang (Octabriyantiningtyas, Suryani, & Jatmiko, 2019).

Uang elektronik merupakan sebuah alat pembayaran dalam bentuk elektronik dimana nilai uangnya disimpan dalam media elektronik tertentu. Pengguna harus menyetorkan uang terlebih dahulu kepada penerbit dan disimpan dalam media elektronik sebelum digunakan untuk keperluan bertransaksi. Nilai uang yang tersimpan di dalam uang elektronik juga akan berkurang sesuai dengan besar nilai transaksi yang dilakukan dan dapat diisi kembali (*top up*). Media yang digunakan untuk menyimpan *e-money* berupa *chip* atau *server* (Puspitawati & Gurning, 2019). Penggunaan *e-money* ini dinilai sebagai alat pembayaran yang inovatif serta praktis dan diharapkan dapat membantu kelancaran pembayaran kegiatan ekonomi yang bersifat massal, cepat dan mikro. Perkembangan *e-money* juga diharapkan sebagai alternatif pembayaran non tunai sehingga dapat menjangkau masyarakat yang selama ini belum mempunyai akses kepada sistem perbankan. Adapun beberapa resiko penggunaan dari *e-money* yaitu dapat hilang atau digunakan oleh orang lain. Hal ini dikarenakan prinsip yang terdapat pada *e-money* sama dengan uang tunai yang apabila hilang maka tidak dapat diklaim kepada penerbit. Selain itu pengguna yang tidak memahami betul cara pemakaian *e-money* juga bisa berakibat penempelan dua kali pada *reader* sehingga besar transaksi yang dilakukan bisa lebih besar dari nilai aslinya.

Berdasarkan jenis nya *e-money* terbagi menjadi dua jenis yaitu *e-money registered* dan *e-money unregistered*. *E-money registered* yaitu uang elektronik yang data identitas pemegangnya tercatat/terdaftar pada penerbit. Dalam hal ini penerbit harus menerapkan prinsip mengenal nasabah dalam menerbitkan uang elektronik *registered* dan batas maksimum nilai uang elektronik ini sebesar Rp5.000.000 (lima juta rupiah). Sedangkan *e-money unregistered* merupakan uang elektronik yang data identitas pemegangnya tidak terdaftar pada penerbit uang elektronik dan batas maksimum yang tersimpan pada uang ini sebesar Rp1.000.000 (satu juta rupiah) (Bank Indonesia, 2020).

2.1.2 Sistem *e-ticketing* KAI

E-ticketing merupakan sebuah sistem yang digunakan untuk mendokumentasikan proses penjualan tiket dari aktivitas perjalanan pelanggan tanpa memerlukan dokumen fisik yang berharga. Menurut Schwake, Sukoharsono dan Handayani (2015) *e-ticketing* merupakan fasilitas pemesanan tiket secara online yang berguna membantu orang lain untuk dapat mempunyai akses ke system tersebut. Hal ini membuat banyak komunitas yang menyediakan jasa penjualan tiket secara online. Selain itu *e-ticketing* juga digunakan sebagai peluang untuk meminimalisir biaya operasional dan meningkatkan kenyamana pelanggannya. Dengan adanya *e-ticketing* juga membuat berkurangnya penggunaan kertas formular sehingga membuat penumpang menjadi lebih fleksibel dalam membuat dan merubah jadwal perjalanannya. Salah satunya adalah reservasi tiket kereta api.

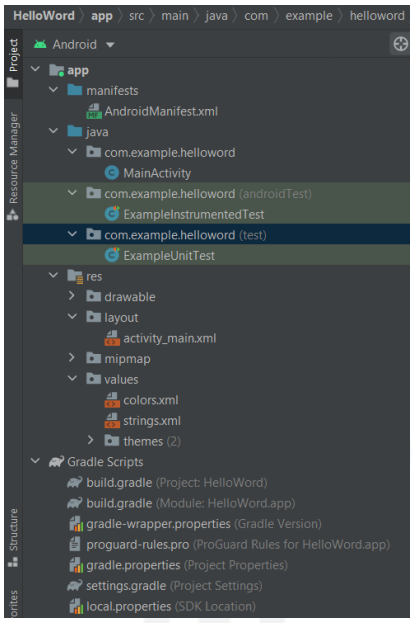
Reservasi tiket kereta api adalah pemesanan kursi kereta api dengan saling bertukar informasi antara konsumen dengan pemilik jasa layanan. Salah satu penyedia jasa pemesanan tiket kereta api adalah PT. KAI (Persero). PT.KAI menyediakan layanan reservasi melalui berbagai metode seperti *website* KAI, KAI *Access*, Tokopedia, Indomaret, Alfamart, dan lainnya. Pemesanan tiket tersebut juga didukung dengan sistem pembayaran yang praktis seperti melalui ATM, Transfer, Internet *Banking, M-Banking* dan lainnya. Hal ini memudahkan pelanggan dalam melakukan reservasi tanpa harus mengantri langsung di loket (Ria, Nugraha, & Firmanto, 2021) .

2.2 *Front-end Framework*

2.2.1 Android *native*

Android *native* merupakan *platform development* yang paling popular dan banyak di gunakan oleh para *developer*. Bahasa yang dapat digunakan adalah Java, Kotlin atau C++. *Framework* ini rilis pada bulan Oktober 2009 dan bersifat *open source*. API *access* yang digunakan adalah *Java Framework* APIs. *Native development kit* (NDK) pada Android menyediakan API *library* dan *tools* yang membantu pembuatan, pengujian, dan men-*debug* aplikasi. NDK digunakan untuk meng*compile* kode C/C++ menjadi *native library* dan dikemas ke dalam APK dengan menggunakan *Gradle*. *Framework* ini menyediakan akses langsung ke komponen *platform* seperti kamera, mikrofon, sensor dan lainnya. *RenderScript* berfungsi untuk menjalankan komputasi sehingga mencapai kinerja yang baik pada aplikasi *native*. *RenderScript* sering digunakan untuk aplikasi yang melakukan *computer vision*, komputasional fotografi, dan pemrosesan gambar.

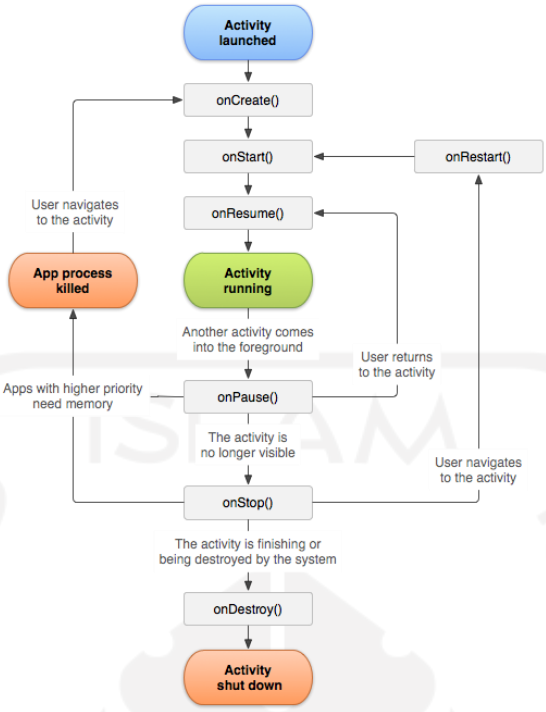
Open GL merupakan API grafis lintas *platform* yang menentukan antarmuka *software* standar untuk pemrosesan 3D grafis pada perangkat keras. Selain itu pada terdapat Android Native Game SDK yang membantu *developer game* untuk menjangkau lebih dari 2,5 miliar perangkat aktif bulanan dengan *platform* Android. Ada beberapa perusahan yang menggunakan seperti Google, Slack, dan lainnya. Keunikan dari Android *native* adalah menyediakan tools tercepat dalam membantu *developer* dalam membuat aplikasi dan game. Struktur folder Android *native* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur folder Android *native*

Pada *file* proyek Android terdapat beberapa bagian yaitu:

a. *build*, folder berisi *output* dari *build*
b. *libs*, folder berisi *library* pribadi
c. *src*, folder berisi *file* kode dan *resource* untuk modul dalam sub-direktori
    1. *androidTest*, folder berisi kode untuk pengujian
    2. *main*, folder berisi *file* utama kode dan *resource* Android. Pada *file* ini terdapat file *AndroidManifest.xml* yang menjelaskan sifat aplikasi dan setiap komponennya. *Java* pada folder ini berfungsi sebagai sumber kode *Java*. Jni berisi kode *native* yang menggunakan *Java Native Interface* (JNI). *Gen* berisi *file* Java yang dihasilkan dari Android Studio. *Res* berisi *resource* aplikasi seperti *drawable*, tata letak, *string* UI. *Assets* berisi *file* yang harus dikompilasi menjadi *file* apk.
d. *test*, folder berisi kode untuk pengujian lokal yang dijalankan di JVM *host.*
e. *bulde.gradle*, folder berisi definisi konfigurasi *build*.



Gambar 2.2 *Lifecycle* Android *native* (android, 2021)

Lifecyle Android native dapat dilihat pada Gambar 2.2. Pada *lifecycle* tersebut terdiri dari beberapa kondisi yaitu:

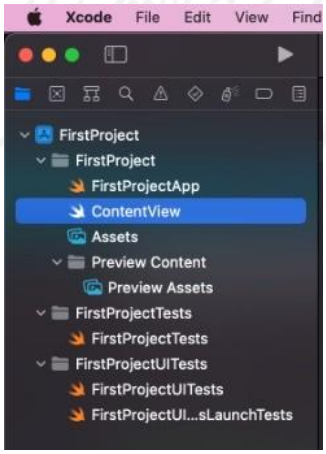
a. *onCreate()* merupakan *method* yang aktif ketika sistem pertama kali membuat aktivitas dan hanya bole terjadi sekali selama siklus aktivitas.
b. *onStart()* merupakan kondisi ketika aplikasi mempersiapkan aktivitas untuk memasuki latar depan dan sebagai tempat aplikasi menginisialisasi kode yang mengelola UI.
c. *onResume()* merupakan kondisi ketika aplikasi berinteraksi dengan pengguna.
d. *onPause()* merupakan kondisi ketika pengguna meninggalkan aplikasi (meski tidak ditutup) atau aktivitas sedang dijeda.
e. *onStop()* merupakan kondisi ketika aktivitas telah selesai berjalan dan segera dihentikan.
f. *onDestroy()* merupakan kondisi ketika aktivitas benar-benar selesai dan sistem ditutup karena perubahan konfigurasi (rotasi perangkat atau mode *multi-window*) (android, 2021).

2.2.2 SwiftUI

SwiftUI merupakan *framework declarative* yang dikembangkan oleh Apple untuk membangun tampilan aplikasi berbasis iOS. Framework ini dirilis pada bulan September 2019. Bahasa yang digunakan adalah bahasa yang digunakan oleh Swift yaitu *objective*-c yang rilis pada tahun 2014.

SwiftUI fokus pada *state changes* atau perubahan tampilan secara otomatis ketika terjadi perubahan data. SwiftUI menyediakan *event handle* yang berfungsi mendeteksi *taps*, *gestures* dan berbagai input sehingga *developer* fokus pada alur data, tampilan dan efek *input* dari *user*. Pemrograman pada *framework* ini menggunakan sintaks deklaratif sehingga mempermudah menyatakan apa yang harus dilakukan oleh *user interface*. Selain itu kode menjadi lebih sederhana, mudah dipahami, mengehemat waktu dan pemeliharaan aplikasi. *Interface* aplikasi pada SwiftUI dapat disesuaikan ke dalam berbagai bahasa dan negara. *Tools* yang digunakan untuk menggunakan *framework* ini adalah Xcode. Xcode akan menkompilasi ulang perubahan UI secara instan. Pada Xcode terdapat kemudahan untuk mengubah *interface* hanya dengan men-*drag* dan meletakannya di kanvas desain atau langsung di kodenya. Xcode juga dapat menukar kode yang diedit secara langsung secara dinamis. Selain itu dengan Xcode dapat membuat banyak *preview* dari berbagai *device* (Apple, 2020). Struktur folder SwiftUI dapat dilihat pada Gambar 2.3. Pada *file* proyek Swift tersusun dari berberapa bagian yaitu:

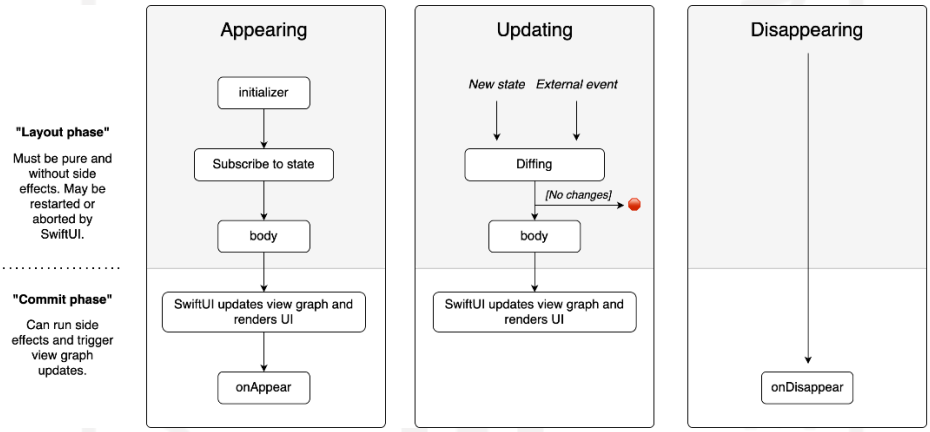
a. *App* berfungsi sebagai *main* aplikasi atau tempat yang memberitahu halaman mana yang akan ditampilkan.
b. *ContentView* berfungsi sebagai tempat menyimpan seluruh kode Swift untuk membuat tampilan aplikasi.
c. *Assets* berfungsi sebagai tempat menyimpan gambar, *color*, dan lainnya.
d. *Preview Assets* berfungsi sebagai tempat untuk mengkonfigurasi data seperti gambar, warna, dan jenis *asset* lainnya yang biasa ditambahkan ke *file asset*.
e. *ProjectTests* berfungsi sebagai tempat untuk membuat pengujian fungsionalitas aplikasi.
f. UI*Tests* berfungsi sebagai tempat membuat pengujian terhadap *user interface* aplikasi.



Gambar 2.3 Struktur folder SwiftUI

Lifecycle dari SwiftUI dapat dilihat pada Gambar 2.4. Pada *lifecycle* tersebut terdapat beberapa fase yaitu:

a. *Appearing* merupakan kondisi ketika *view* tampil ke dalam halaman aplikasi. Pada fase ini sebuah *view* akan diinisialisasi, diterima oleh *state*, dan di*render* untuk pertama kalinya.
b. *Updating* merupakan kondisi yang dilakukan sebagai respon terhadap peristiwa eksternal atau mutasi *state*.
c. *Dissappearing* merupakan kondisi untuk menghapus *view* dari halaman aplikasi (Bulavin, 2020).



Gambar 2.4 *Lifecycle* SwiftUI (Bulavin, 2020)

2.2.3 *Flutter*

Flutter dirilis resmi pada bulan Desember 2018 oleh Google. *Flutter* merupakan SDK untuk mengembangkan aplikasi baik *mobile*, web maupun desktop. *Flutter* adalah *framework* yang membantu *developer* untuk membuat aplikasi *mobile multiplatform*.

Banyak perusahaan yang telah menggunakan framework ini seperti Alibaba, ebay, hingga Google Ads (Flutter, 2018). Terdapat 2 komponen penting dalam *Flutter* yaitu:

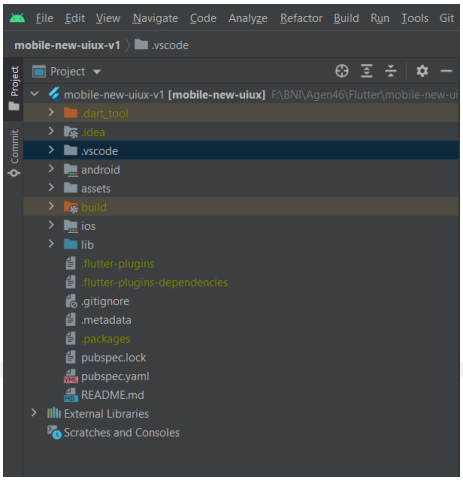
a. Software Development Kit (SDK) adalah kumpulan *tools* yang berfungsi untuk meng*compile code* agar aplikasi dapat dijalankan di berbagai platform.
b. Framework UI adalah komponen UI seperti teks, *button*, *navigation* dan lainnya yang dapat dikustomisasi.

Setiap membuat UI di dalam *Flutter* akan menciptakan berbagai *widget*. Beberapa karakteristik dari *widget* yaitu dapat digunakan untuk membuat UI seperti *column*, *row*, *text*, *stack*, *card*, *form*, *padding*.

Developer juga dapat melakukan *styling* seperti tipe *font, size, color, margin, shadow*. Selain itu *widget* dapat berupa kumpulan bentuk dan formular. Di dalam *Flutter* terdapat fitur *hot-reload*. Fitur ini memudahkan *developer* dalam membangun UI, menambahkan fitur dan memperbaiki *bug* dengan melihat langsung perubahan UI yang terdapat pada aplikasi. *Hot-reload* bekerja dengan menanamkan kode file terbaru ke dalam *Dart Virtual Machine* (VM). Kerangka pada *Flutter* membuat ulang semua *widget* secara otomatis setelah VM diperbarui. Fitur ini yang memudahkan *developer* untuk melihat langsung perubahan UI yang dikembangkan.

Untuk pengembangan aplikasi *mobile*, *Flutter* dapat digunakan untuk mengembangkan aplikasi berbasis Android dan iOS dengan menggunakan bahasa Dart. Dart merupakan bahasa pemrograman berbasis *class* dan berorientasi objek. Di dalam *Flutter* terdapat dua kombinasi *class* yang digunakan yaitu: *Stateless Widgets* dan *Stateful Widgets*. *Class* ini merupakan *widget* yang bertugas menampilkan tampilan *user interface* dengan membangun *widget* lainnya agar menjadi lebih terlihat nyata. *Stateless Widgets* adalah *widgets* yang bersifat statis dan seluruh konfigurasi yang dimuat sudah diinisialisasi dari awal atau hanya dapat menggambar satu kali saat *widget* dimuat sehingga tidak dapat menggunakan fitur *hot-reload*. Sedangkan *Stateful widgets* adalah *widgets* yang bersifat dinamis dan dapat berubah pada kondisi tertentu sehingga dapat menggunakan fitur *hot-reload* (Boukhary & Colmenares, 2019).

Dengan *Flutter* kita dapat mengembangkan aplikasi ke semua jenis *platform*. Hal ini dikarenakan Dart memiliki AOT-*Compile* (*Ahead Of Time*). Ketika *developer* mengembangkan aplikasi dengan bahasa Dart, AOT akan meng*compile code* yang dibuat dari bahasa Dart ke *native* menuju *platform* yang diinginkan. Jika pada Android *code* tersebut akan diterjemahkan ke dalam bahasa Kotlin. Pada Android *code* tersebut akan di*compile* dengan mengunakan mesin C++ dan menggunakan *Native Development Kit* (NDK). Kemudian *code native* pada Android akan ter*compile* lagi dengan Dart *Compiler*. Sedangkan di iOS akan di*compile* dengan mesin LLVM *(Low-Level Virtual Machine)*. Setelah itu *code* akan disesuaikan dengan masing-masing perangkat agar dapat dijalankan pada berbagai *platform*. Struktur folder *Flutter* dapat dilihat pada Gambar 2.5.



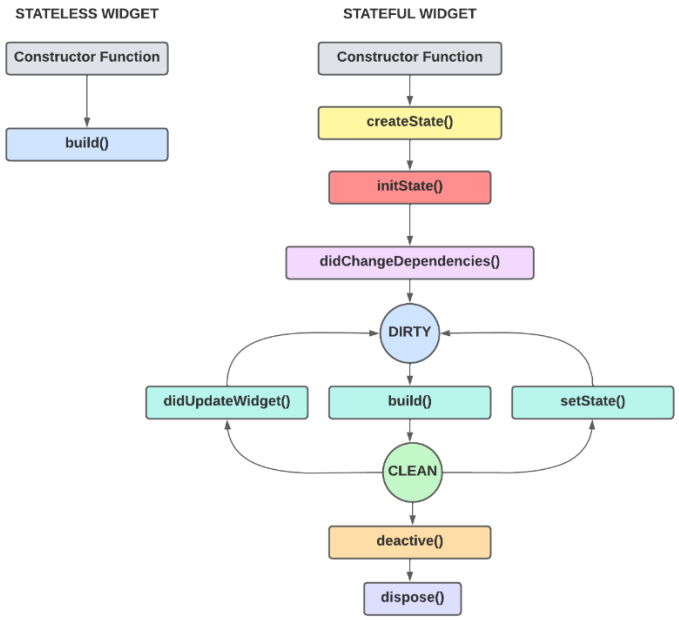
Gambar 2.5 Struktur folder *Flutter*

Pada struktur folder *Flutter* terbagi menjadi beberapa modul aplikasi yaitu:

a. *.dart_tool* : folder ini digunakan oleh *pub* dan *tools* lainnya.
b. *.idea*: berisi segala pengaturan *project* yang disimpan secara spesifik dalam bentuk xml
c. *android*: folder yang diperlukan untuk proses *build* menjadi aplikasi *native* ke *platform* Android. Pada folder ini terdapat *file build.gradle* dan berfungsi sebagai tempat untuk menentukan SDK dari aplikasi. Selain itu terdapat *file AndroidManifest.xml* yang berguna untuk mengatur *permissions* seperti penggunaan *camera*, *sound*, *recording*, *file explorer* dan lainnya. *File* utama pada folder *android* adalah folder *main*.
d. *ios*: folder ini digunakan untuk menjalankan aplikasi ke *platform* iOS.
e. *lib*: merupakan folder utama yang menyimpan seluruh *code* dari aplikasi *Flutter* yang dibuat.
f. *test*: folder ini buat untuk melakukan *testing* dari *code* yang dibuat.
g. *assets*: berfungsi untuk menyimpan *images* atau *font* tambahan
h. *.gitignore*: berisi daftar *file*, ekstensi, dan folder yang akan diabaikan saat menggunakan *Git*.
i. *.metadata*: berfungsi melacak *properties* dari *project Flutter*. *File* ini juga digunakan untuk mengukur kemampuan dan menjalankan *update*.
j. *.packages*: isi dalam *file* ini akan di*generate* secara otomatis oleh *Flutter* SDK dan digunakan untuk mendata *dependencies* dari *project Flutter*.
k. *nama_project.iml*: berisi pengaturan dari project Flutter dan diberi nama sesuai dengan nama *project* nya.
l. *pubspec.yaml*: merupakan *file* konfigurasi yang sering digunakan dalam *project Flutter*.

m. *pubspec.lock*: berfungsi mengunci versi *package* dan berisi versi *library* yang dibuat oleh *pub*.

n. *README.md*: berfungsi sebagai *guide* bagi *developer* yang baru bergabung ke dalam *project Flutter*.



Gambar 2.6 *Lifecycle Flutter*

Lifecycle dari *framework Flutter* dapat dilihat pada Gambar 2.6. Pada *lifecycle* tersebut terdapat beberapa *method* yaitu:

a. *createState()* merupakan *method* ketika membuat sebuah state objek. Setelah membuat objek *framework* akan mengumpulkan *state* objek dengan *BuildContext* dan mengatur *property Boolean* yang dipanggil *mounted* menjadi *true.*

b. *initState()* merupakan *method* yang hanya sekali dipanggil ketika *widget* dibuat. Mengikuti *stream* atau objek apapun yang dapat mengubah data *widget*.

c. *didChangeDependencies()* merupakan *method* yang dipanggil setelah *initState()* dan ketika *dependency* dari objek *state* berubah melalui *InheritedWidget*.

d. *build()* merupakan kondisi ketika *widget tree* dibangun. *Build* akan dipanggil setiap kali *state widget* (*didUpdateWidget()* atau *setState()* metode ini dipanggil).

e. *setState()* merupakan *method* yang sering dipanggil dari *framework Flutter* sendiri dan dari *developer*. *Method* ini digunakan setiap kali mengubah status internal dari suatu *state*.

f. *didUpdateWidget()* merupakan *method* yang dipanggil setiap kali konfigurasi *widget* berubah seperti ketika *parent widget* meneruskan beberapa *variable* ke *children widget* melalui *constructor*.

g. *deactivate()* merupakan *method* yang dipanggil ketika objek dihapus dari *widget tree* tetapi data dimasukan kembali sebelum terjadi perubahan pada *frame* yang dipilih selesai.

h. *disposes()* merupakan kondisi keika *widget/ state* objek dihapus dari *widget tree* secara permanen. Contoh saat menggunakan *pushReplacement()* dari navigator. Setelah dipanggil *method dispose()* maka *property mounted* saat ini bernilai *false*.

Adapun beberapa kelebihan yang dimiliki oleh Flutter yaitu:

- *Expressive beautiful* UI (*User Interface*).

  Flutter menyediakan banyak jenis *widget*. *Developer* dapat menggunakan *widget* untuk membuat tampilan aplikasi speerti *layout*, navigasi, *style* tema, *font, animation*, hingga pengaturan *scrolling*. Selain itu *widget* tersebut dapat dikustomisasi sesuai dengan kebutuhan. *Fluter* juga *support* Cupertino (iOS *style*).

- *Native performance*.

  Penggunaan *Flutter* memudahkan *developer* dalam membuat aplikasi *mobile* berbasis *multiplatform* hanya dengan satu basis *code* saja. Hal ini membuat *developer* menjadi lebih efisien dari segi biaya pengembangan sehingga menjadi lebih produktif. Selain itu aplikasi dengan *Flutter* memiliki performa yang tidak kalah dari aplikasi *native.* Hal ini dikarenakan *Flutter* dapat meng*compile code* aplikasi *native* dengan mesin render yang canggih.

- *Fast development.*

  Flutter memiliki fitur *hot reload* yang membantu *developer* dalam membuat perubahan yang terjadi pada aplikasi dapat dilihat secara langsung tanpa mengkompilasi ulang. Selain itu *hot reload* membantu waktu *reload* dengan cepat tanpa kehilangan *state* pada emulator, simulator maupun *hardware*. Fitur ini membantu *developer* apabila ingin menambahkan, memperbaiki *bug*, hingga melakukan eksperimen pada tampilan aplikasi.

2.2.4 Perbandingan Android *native*, SwiftUI, dan *Flutter*.

Penelitian terdahulu yang dilakukan oleh Hina, Kamran, Faiza, Dr. Qasim, dan Dr. Isma yang berjudul "*Comparative Study of Android Native and Flutter App Development*", 2021 menjelaskan tentang perbandingan kinerja yang dari kedua *framework*.

Perbadingan kinerja diukur berdasarkan dari parameter *CPU usage*, *memory usage*, *power usage*, dan FPS (*frame per second*). Metodologi yang digunakan adalah *testing* dan survei dari opini para *developer*. Penelitian dilakukan dengan menggunakan aplikasi dengan fitur *sign in*, *sign up*, dan *home screen*. Performa dari setiap parameter akan dihitung dengan menggunakan *tools* Apptim. *Device* yang digunakan untuk pengujian menggunakan Huawei dengan spesifikasi API level 7, RAM 3 GB, android versi Oreo (8.10). Berdasarkan pengujian yang telah dilakukan diketahui bahwa dengan parameter *CPU usage* dan *memory usage* Android *native* lebih unggul sedikit. Pada parameter *power usage* Android *native* dan *Flutter* memiliki nilai yang sama. Pada parameter FPS, *Flutter* memiliki performa yang lebih tinggi dibandingkan Android *native*. Selain itu selama mengirim *request*, Android *native* men*download* data sebesar 19,19 KB/sec sedangkan *Flutter* lebih unggul sebesar 16,37 KB/sec. Survei online yang dilakukan oleh peneliti dengan mempertimbangkan opini *developer* didapatkan bahwa aplikasi *Flutter* memiliki kinerja yang lebih baik dan memiliki kurva pembelajaran yang lebih sedikit jika dibandingkan dengan Android *native* (Hussain, Khan, Siddiqui, Farooqui, & Ali, 2021). Terdapat perbandingan kelebihan dan kekurangan dari ketiga framework tersebut yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan Android *native*, SwiftUI, dan *Flutter*

Framework	**Kelebihan**	**Kekurangan**
Android *native*	• Biaya pengembangan rendah dan ROI tinggi. • Bersifat *open source*. • Mudah dintegrasikan ke beberapa *platform*. • Memiliki banyak sales channel untuk menjual aplikasi (Google Play Store). • Lebih mudah diadopsi karena dari bahasa Java. • *Deployment* lebih cepat. • Tingkat keamanan aplikasi tinggi. • Dapat dikustomisasi. • Komunitas besar dan dokumentasi lengkap.	• Ukuran UI yang kurang fleksibel. • Tidak semua API cocok. • Keterbukaan kode membuat rentan terhadap ancaman. • *Layout* dan *animation* rumit dikodekan di Android. • Banyak *proccess* di *background* sehingga menyebabkan penggunaan baterai boros. • Fragmentasi *device* tinggi.

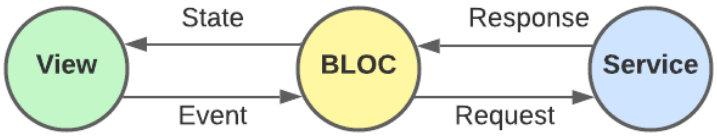
*Swift*UI	• Mudah dipelajari. • Dapat membuat tampilan secara langsung dengan cara *drag* dan *drop*. • Jarang terjadi *crash* saat *runtime* karena kode UI akan ikut terkompilasi secara bersama dengan *compiler* Swift. • *Source code* mudah di *maintenance*. • Teknologi *compiling* LLVM yang cepat. • *Syntax* lebih modern dan simple. • Dapat menggunakan banyak *animation* dan *transition*.	• Bahasa pemrograman masih baru. • *Talent developer* masih terbatas. • Tidak dapat digunakan pada versi proyek sebelumya. • Hanya mendukung sistem operasi iOS 13 dan Xcode 11. • Dokumentasi dan *library* masih kurang sehingga sulit mencari solusi *bug* atau masalah. • Tidak dapat mengecek struktur tampilan di Xcode *preview*.
Flutter	• Besifat *open source*. • Tampilan *user interface* yang menarik. • *Widget* yang dapat dikustomisasi. • Tersedia fitur *hot-reload*. • Mendukung Cupertino (iOS *style*). • Performa yang bagus seperti *native*. • Mendukung pengembangan *cross-platform* dan *multiplatform* hanya dengan satu basis *code*. • Pengembangan yang cepat. • Dapat digunakan di semua IDE (Android Studio, Intellij IDE, dan VSCode). • Mudah dipelajari dan digunakan. • Dokumentasi yang lengkap.	• Ukuran *file* yang besar. • *Library* pihak ketiga masih belum lengkap. • Kominitas masih berkembang. • Harus mempelajari bahasa baru yaitu Dart.

BNI membutuhkan *refreshment* aplikasi terkini dengan desain *user interface* yang baru pada aplikasi *m-banking* Agen46. Pembuatan aplikasi ini menggunakan *Flutter* untuk menghemat *resource* karena dengan *framework* ini pengembangan dapat dilakukan dalam sekali *build* baik versi Android maupun iOS. Selain itu terdapat *capability* dalam Flutter yang menunjang pemgembangan *m-banking* Agen46 seperti *biometric*. Dengan kelebihan-kelebihan dari *Flutter* membuat BNI memilih *framework* ini sebagai teknologi pengembangan *m-banking* nya. Hal ini diharapkan mampu agar pengembangan yang dilakukan menjadi lebih efisien.

2.3 *Design Pattern*

2.3.1 *BLoC Pattern*

Business Logic Component Pattern adalah pola desain yang membantu memisahkan *user interface* dengan *business logic. Pattern* ini juga memungkinkan *developer* untuk membagi komponen pada proyek menjadi komponen *presentational, BLoC,* dan *backend*. Alur proses dengan *BLoc Pattern* dapat dilihat pada Gambar 2.7.



Gambar 2.7 *BLoc pattern*

Pengembang dapat fokus untuk mengkonversi sebuah *event* menjadi *state* saat menggunakan *BLoC Pattern*. Hal ini memudahkan pengembang untuk memahami kode yang dikembangkan sebelumnya serta membuat aplikasi yang lebih kompleks menjadi komponen yang lebih kecil agar lebih mudah untuk diuji. Ada beberapa komponen dalam menerapkan *BLoC Pattern* yaitu:

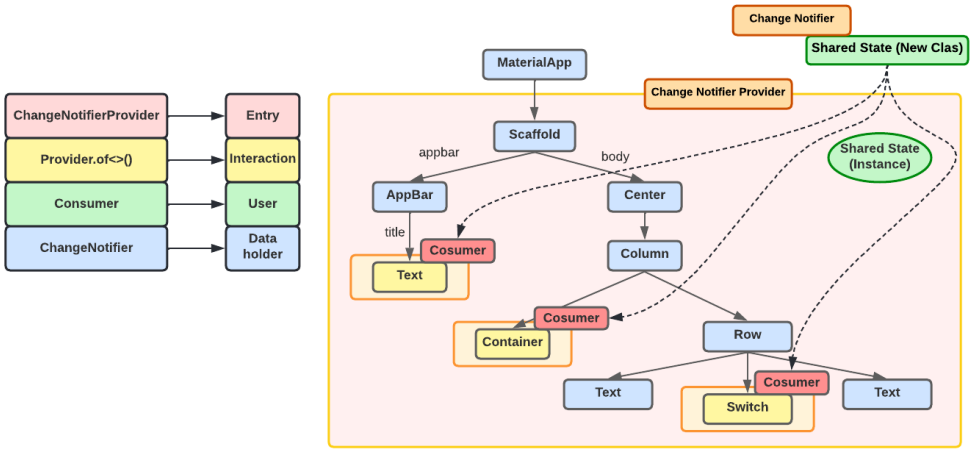
a. *View* berfungsi sebagai tempat memasukkan *input* dan menampilkan *output*.
b. *Bloc*
    1. *Bloc* berfungsi sebagai penghubung antara *view* dengan *service*.
    2. *State* berfungsi untuk memproses *response* dari *service* (yang ditampung oleh model) dan menyatakan apakah *response failed*, *error* atau *success*.
    3. *Event* berfungsi sebagai kondisi ketika proses meng-*hit* atau mengirim ke *service.*
c. *Model* berfungsi menerima *response* dari *Restful API* berupa JSON yang ditampung dalam sebuah *class*.
d. *Service*
    1. *Service* berfungsi untuk meng-hit Restful API atau sebagai servernya.
    2. *Params request* berfungsi sebagai data berupa *class* yang dibutuhkan ketika meng-hit ke Restful API.
e. *Route* berfungsi sebagai penghubung antara *view* satu dengan yang lainya.

BLoC Pattern memiliki alur *logic* yang simple dan mudah dipahami sehingga *pattern* ini dapat digunakan oleh *developer* dengan berbagai tingkat keterampilan. Selain itu *pattern* ini dapat membantu pembuatan aplikasi kompleks yang tersusun dari komponen-komponen yang lebih kecil. Pembagian aplikasi menjadi komponen-komponen kecil membuat setiap bagian menjadi lebih mudah untuk diuji (Hoang, 2019).

2.3.2 *Provider Pattern*

Pattern ini dibuat oleh Remi Rousselet. *Provider* adalah sekumpulan *class* yang menyederhanakan pembuatan solusi *state management*.

Pada provider, *widget* akan mendengarkan perubahan *state* dan meng*update* nya setelah diberitahu. *Widget* akan di*build* ulang pada seluruh *widget tree* saat ada perubahan *state* namun hanya *widget* yang dipengaruhi saja sehingga hal ini akan mengurangi pekerjaan serta membuat aplikasi dapat berjalan lebih cepat dan *smooth*. Setelah *state* berubah, *widget* khusus akan di*build* ulang tanpa mempengaruhi *widget* lainnya dalam struktur *widget*. *Provider pattern* dapat dilihat pada Gambar 2.8.



Gambar 2.8 *Provider pattern*

Ada beberapa bagian penting dalam *provider pattern* yaitu:

a. *Provider* berfungsi mengambil dan menyebarkan *value* apapun. *Provider.of<>()* dapat mengakses data yang ada di dalam *class*nya. Tugasnya adalah membaca data dari *class*nya. Terdapat *property* data dan *listen*. *Listen* memiliki *default true* dan harus di *set* menjadi *false*.

b. *ChangeNotifier* merupakan *class* yang berfungsi menyimpan *state* yang akan diperbarui dan *widget* yang menggunakannya akan diberitahu. *Class* harus meng*extend ChangeNotifier* jika menggunakan *package Provider*. Kemudian membuat sebuah *class* baru dan meng*extends ChangeNotifier*.

c. *ChangeNotifierProvider* merupakan *widget* yang membungkus dan mengimplementasikan *instance ChangeNotifier* agar dapat diakses atau digunakan oleh *widget child*. Ketika perubahan disebarkan *widget* akan mem*build* ulang *widget tree*. *Class* ini digunakan saat masuk ke aplikasi. *MaterialApp* perlu dibungkus dengan *ChangeNotifierProvider*. Terdapat *property create* dan *child*.

d. *Consumer* merupakan sebuah *widget* yang mendengarkan perubahan *state* di *class* yang mengimplementasikan *ChangeNotifier* kemudian mem*build* ulang *widget* turunanya.

Hal ini akan membantu ketika ingin menampilkan perubahan data yang ada di UI atau *view*. Terdapat *property context*, data, dan *child*. Data merujuk pada *instance* dari *class* nya. *Child* merujuk pada *widget* itu sendiri (Dev, 2022).

2.3.3 Perbandingan *BLoC pattern* dan *provider pattern.*

Penelitian pertama terdahulu yang dilakukan oleh Regawa Rama Prayoga, Ghifari Munawar, Rahil Jumiyani, dan Alifia Syalsabila (2021) dalam penelitiannya berjudul "*Performance Analysis of BLoC and Provider State Management Library on Flutter*". Dalam penelitiannya menjelaskan mengenai efek penggunaan *library state management* pada *build* proses dengan teknologi *Flutter*. *State management* yang digunakan pada penelitian ini adalah *setState*, *BLoC* dan *provider*. Dalam studinya dilakukan pengujian efisiensi penggunaan *state management* dengan menggunakan *BLoC* dan *provider* dalam prosses *build* jika dibandingkan dengan mekanisme standar *Flutter* dengan *setState*. Ketiga *state* tersebut diuji berdasarkan beberapa indikator yaitu berdasarkan kinerja penggunaan CPU, *memory* dan waktu eksekusi dengan menggunakan Snapdragon Profiler *tools*. Metode yang digunakan penelitian ini dilakukan dengan beberapa tahapan dari mulai tinjauan pustaka, analisis literatur hasil studi, eksperimen, analisis hasil eksperimen, hingga kesimpulan. Tinjaun pustaka dilakukan untuk mempelajari sumber dan mengumpulkan segala kebutuhan yang diperlukan untuk melakukan penelitian. Analisis literatur hasil studi dilakukan untuk melakukan analisis dari hasil studi serta mendapatkan faktor-faktor yang mempengaruhi kinerja CPU, *memory* dan waktu eksekusi dalam pembuatan aplikasi.

Eksperimen dilakukan untuk membuat objek riset yang dapat diukur kinerjanya. Aplikasi yang dibuat adalah aplikasi katalog film dengan nama MovDB. Film MovDB ini menyediakan informasi film seperti judul, poster, *genre*, hinggga ringkasan filmnya. Eksperimen yang dilakukan dengan menggunakan 1000 dan 10.000 data. Selain itu setiap sekenario akan dijalankan sebanyak tiga iterasi. Hasil dari iterasi kemudian dirata-rata. Skenario inilah yang mengukur kinerja dari penggunaan CPU, *memory*, dan waktu eksekusi. Percobaan dilakukan dengan menggunakan perangkat android dengan spesifikasi Android 10, RAM 4 GB, dan prosesor Qualcomm Snapdragon460. Berdasarkan analisis perbandingan eksperimen yang dilakukan pada penelitian tersebut diketahui bahwa penggunaan library state management memiliki kinerja tampilan yang lebih baik saat digunakan pada *leaf widget* daripada *parent widget.*

Berdasarkan analisis didapatkan data *state BLoC* dari penggunaan CPU sebesar 0,45 (2,14% lebih efisien), penggunaan *memory* sebesar 23,27 MB (8,19% lebih efisien), dan waktu eksekusi sebesar 3 detik (16,36% lebih efisien). Berdasarkan beberapa tahap yang telah dilakukan pada peneletian ini didapatkan kesimpulan bahwa *object widget* yang harus di*render* saat proses *build* pada *BLoC Pattern* memiliki jumlah yang lebih sedikit sehingga prosesnya menjadi lebih ringan. Hal ini menyebabkan alokasi *memory* yang digunakan menjadi lebih efisien karena jumlah *object* yang disimpan tentu lebih sedikit. Tidak hanya itu semakin sedikit jumlah *widget* yang dibuat, semakin singkat proses *build*, dan waktu *render* maka waktu eksekusinya menjadi lebih efisien (Mantik, Rama Prayoga, Munawar, Jumiyani, & Syalsabila, 2021).

Penelitian kedua terdahulu oleh Muhammad Zulfa Assyfa dengan judul "Implementasi Desain *Pattern BLoC* dan *Pattern Repository* pada Aplikasi *Energy Management System* PT. PLN Berbasis *Mobile* dengan *Flutter*". Dalam penelitiannya membahas tentang pembuatan perangkat lunak berbasis *mobile* dengan *Flutter* yang berfungsi untuk membantu kegiatan monitoring dan pembayaran pemakaian listrik PLN Group. Metode penelitian yang dilakukan melewati beberapa tahap dari mulai analisis literatur, perancangan sistem, pemograman sistem, hingga *testing* dan *debugging* aplikasi. Metode desain yang digunakan adalah *BLoC Pattern*. Penggunaan *BloC Pattern* ini diterapkan dalam pembuatan *dashboard* aplikasi EMS. *Dashboard* tersebut digunakan untuk memonitoring pemakaian energi. Pada *dashboard* tersebut terdapat beberapa informasi yaitu jumlah pelanggan, jumlah area, pemakaian energi bulan ini dan pemakaian energi bulan lalu. Selain itu terdapat juga fitur lain yaitu *dashboard* grafik, *map monitoring*, rekap pemakaian, rekap pemakaian *detail*, *detail* AMR, dan *profile user*. Aplikasi tersebut hanya dapat diakses oleh *user internal* Indonesia. PLN. Hal ini dikarenakan aplikasi tersebut berfungsi untuk keperluan internal PLN *Group*.

BloC Pattern diterapkan dalam pemisahan file *BloC* yang berisi *business logic*, *event* dan *state*. Berdasarkan pengujian dengan *blackbox testing* fungsionalitas yang terdapat pada aplikasi tersebut berjalan dengan baik. Selain itu dilakukan juga pengujian performa kecepatan dari tampilan aplikasi. Pengujian dilakukan dengan spesifikasi perangkat berupa *Windows 10 Home*, penyimpanan 1 TB, dan RAM 8 GB. Berdasarkan pengujian performa antara aplikasi dengan *BloC Pattern* dan tanpa *BloC Pattern* diketahui bahwa aplikasi yang menerapkan *BloC Pattern* memiliki keunggulan dalam kecepatan berjalannya suatu program. Dalam uji performa yang dilakukan *BloC* unggul tiga kali lebih cepat dari empat kali percobaan (Assyfa, 2020).

Terdapat perbandingan antara BLoC pattern dengan provider pattern yang dapat dilihat pada Tabel 2.2.

Tabel 2.2 Perbandingan *BLoC pattern* dan *provider pattern*

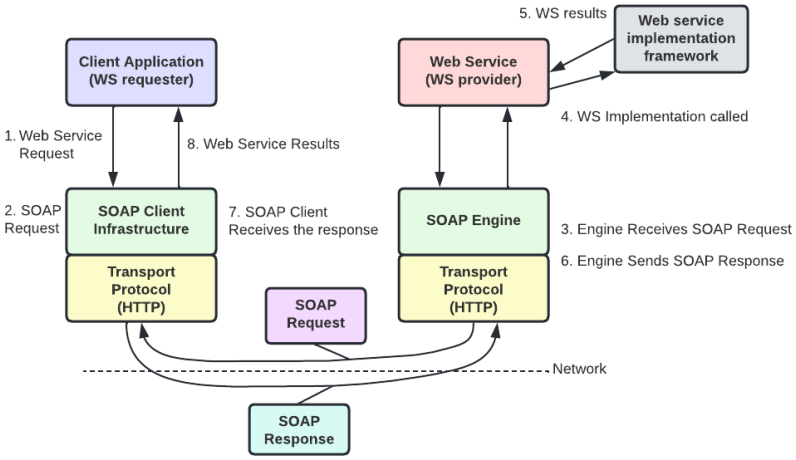
Design Pattern	**Kelebihan**	**Kekurangan**
Provider Pattern	• Lebih fleksibel kerena perubahan *state* hanya terjadi pada *widget* yang membutuhkannya. • Cocok untuk menyebarkan *value* dan objek yang baru. • *Package* telah terperlihara dan teruji dengan baik. • *Toolkit* yang lengkap untuk *state management* yang searah. • Cocok bagi *developer Flutter* yang ingin membangun aplikasi dengan kebutuhan *state* yang sederhana. • Meningkatkan skalabitas. • Mengurangi *boilerplate*. • Menyederhanakan alokasi data dan pembuangan *resource* (data). • Pengembangan yang *friendly* karena dengan *provider, state* pada aplikasi akan terlihat di Flutter *devtool*.	• Sulit untuk digunakan dan dipahami. • *Consumer* akan meng*update* semua *widget* turunannya, meskipun *state* yang tidak relevan telah di*update*. • *Provider* bergantung pada SDK *Flutter* yang membuat *business logic* dan *framework* tidak terpisah. Hal membuat desain arsitektur menjadi buruk. • Pembaruan *state* tidak berdasarkan *event* sehingga memperumit *tracking* dan pemahaman mengenai perubahan *state* pada aplikasi. • Kurang cocok untuk aplikasi yang kompleks.
BLoC Pattern	• Memiliki performa yang lebih baik untuk ukuran data yang besar. • *Logic* simple dan mudah dipahami oleh para *developer*. • Terpisah antara *business logic* dan *user interface*. • Cocok untuk aplikasi yang kompleks untuk membagi menjadi komponen-komponen kecil. • Mudah untuk diuji. • Arsitektur aplikasi dapat tersusun dengan baik sehingga mudah dikelola. • Mengurangi *syntax custom.* • *State* dapat di manajemen dengan baik dan lebih *clean code*.	• Membutuhkan banyak *coding.* • Harus menggunakan dua aliran. • Banyak *boilerplate code* (pengulangan kode di banyak tempat) tanpa ada variasi.

Proyek *m-banking* Agen46 memiliki fitur yang sangat lengkap sehingga dalam pengembangan migrasi ke dalam *framework Flutter* membutuhkan *desain pattern*. *Pattern* yang digunakan adalah *BLoC pattern*. *BLoC Pattern* merupakan satu *state management* dalam *framework Flutter*. *State management* digunakan untuk mengatur data atau *state* saat aplikasi berjalan dengan cara memisahkan antara bagian *logic* dengan *view* nya. Dengan kelebihan yang dimiliki *BLoC* membuat aplikasi yang memiliki kompleksifitas yang tinggi dapat dibagi menjadi komponen-komponen.

Hal ini membuat proyek lebih mudah diuji dan memudahkan *developer* dalam maintenance kodenya. Selain itu dengan penggunaan *BLoC* membuat arsitektur proyek aplikasi menjadi lebih jelas dan terstruktur.

2.4 API

2.4.1 SOAP API



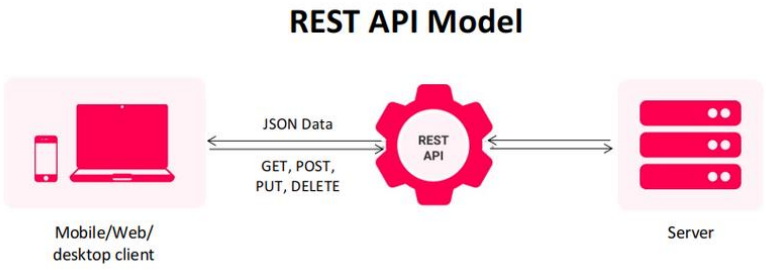
Gambar 2.9 SOAP API

SOAP singkatan dari (*Simple Object Access Protocol*) yang merupakan protokol pengiriman pesan yang menggunakan format XML (*Extensible Markup Language*) untuk menyediakan komunikasi melalui jaringan. SOAP merupakan komponen penting dalam *Service Oriented Architecture* (SOA) dan yang berkaitan dengan *web service*. SOAP dikembangkan oleh Mentor, User Land Company dan Microsoft pada tahun 1998. Pertama kali rilis pada tahun 1999 dengan versi 1.2. Pada versi tersebut SOAP digunakan dengan kecerdasaan yang tinggi. Tujuan utama SOAP adalah meneruskan data melalui jaringan dengan menggunakan HTTP. HTTP (*Hypertext Transfer Protocol*) ini digunakan untuk mentransfer informasi apapun melalui internet. SOAP juga merupakan *platform* dan bahasa yang independen. Alur proses SOAP API dapat dilihat pada Gambar 2.9. Ada beberapa bagian dalam SOAP yaitu:

a. *Envelope* merupakan aspek yang menentukan format pesan SOAP dan menentukan dokumen SOAP yang diberikan tak karakteristik format pesan dari *tak* awal hingga *tak* akhir.

b. *Header* merupakan elemen yang berisi atribut data opsional mengenai informasi tentang verifikasi seperti *authorization*.
c. *Body* merupakan elemen penting yang berisi data XML. *Body* tersebut berisi informasi *request* dan *response* yang akan dikirim ke aplikasi.
d. *Fault* merupakan elemen yang berfungsi menyampaikan informasi tentang kesalahan yang terjadi ketika pengiriman pesan seperti ketidakcocokan, kesalahan pesan dari *client*, dan kesalahan server. Selain itu pada *fault* juga berisi *error* dan status informasi (Soni & Ranga, 2019).

2.4.2 REST*ful* API



Gambar 2.10 REST*ful* API

REST*ful API* adalah API (*Application Programming Interface*) yang mengimplementasikan konsep REST (*Representational State Transfer*). REST diciptakan oleh Roy Fielding pada tahun 2000. API ini bertugas sebagai penghubung yang memastikan sebuah aplikasi untuk dapat berkomunikasi atau berbagi data satu sama lain tanpa harus menggunakan bahasa pemrograman yang sama. API terdiri dari beberapa elemen seperti *protocols, function, tools* dan lainnya. Komunikasi yang digunakan oleh REST adalah arsitektur berbasis web dan umumnya menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai komunikasi datanya. Alur proses dengan REST*ful* API dapat dilihat pada Gambar 2.10. Ada beberapa komponen utama yaitu:

a. Desain URL

Penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik dan mudah dimengerti *developer*. URL API biasa disebut dengan *endpoint*. URL dalam REST*ful* API sangatlah penting. Hal ini karena diakses melalui teknologi WWW (*World Wide Web*). Contoh penulisan URL yang baik adalah sebagai berikut:

- http://example.com/payments

- http://example.com/payments/2

b. HTTP *Verbs*

HTTP *Verbs* merupakan *method request* ketika *client* melakukan *request*. *Method* yang sering digunakan adalah GET (membaca data atau mendapatkan data), POST (membuat data baru dengan menyisipkan data dalam *body* saat *request* dilakukan), PUT (mengupdate data) , dan DELETE (menghapus data). Selain itu terdapat juga PATCH (memodifikasi data tertentu), HEAD (mengambil informasi URL dari server), dan OPTIONS (mencari tahu *method* HTTP mana yang dapat diakses *client*) (Suzanti, Fitriani, Jauhari, & Khozaimi, 2020).

c. HTTP Status *Code*

HTTP Status *Code* merupakan kode standarisasi yang memberitahu hasil *request* kepada *client*. Terbagi menjadi 5 kelompok yaitu :

- 1XX : *response code* yang menampilkan *response* informasi bahwa permintaan diterima atau melanjutkan proses.
- 2XX : *response code* yang menampilkan bahwa *request* berhasil.
- 3XX : *response code* yang menampilkan pesan pengalihan atau perlu tindakan lebih lanjut yang harus diambil untuk menyelesaikan permintaan.
- 4XX : *response code* yang menampilkan bahwa *request* terjadi kesalahan pada sisi *client.*
- 5XX : *response code* yang menampilkan bahwa *request* terjadi kesalahan pada sisi *server*.

d. Format *Response*

Format yang digunakan pada REST*ful* API dapat berupa XML (*Extensible Markup Language*) atau JSON (*Javascript Object Notation*). Setelah mendapatkan data *response*, *client* baru dapat menggunakan dengan cara mem-*parsing* data tersebut dan diolah berdasarkan kebutuhan (Wijayanto, Maryanto, Rahayu, & Iskandar, 2019).

2.4.3 Perbandingan REST*ful* API dan SOAP API

Penelitian terdahulu yang dilakukan oleh Anggraini, Haruno, dan Donny dalam penelitiannya yang berjudul "*REST AND SOAP COMPARISON ON WEB SERVICE TECHNOLOGY FOR ANDROID BASED DATA SERVICE*", 2019 menjelaskan tentang perbandingan antara REST API dengan layanan *web* SOAP dalam menyediakan data. Penelitian tersebut bertujuan untuk mendukung peningkatanan layanan data pada sebuah *web*.

Metodologi yang digunakan menggunakan teknik pengujian data pada sebuah *web service*. Data yang diuji terbagi menjadi 2 yaitu data dengan format ukuran yang sama dan data dengan format ukuran yang berbeda. Pengujian dilakukan menjadi 3 tahap, tahap pertama mengumpulakn 10 data, tahap kedua 20 data, dan tahap ketiga 30 data dengan melakukan *loading* data pada masing-masing *web service*. Dari pengujian yang telah dilakukan didapatkan hasil bahwa *web service* dengan REST API berjalan lebih cepat dibandingkan dengan SOAP. Dengan total data yang didapatkan waktu pengumpulan pada perangkat yang berbeda dengan format ukuran yang sama di REST API menghasilkan 3,4 detik dan SOAP sebesar 3,9 detik. Ketika menerima data pada perangkat yang berbeda dengan format ukuran yang berbeda pada REST API sebesar 4,7 detik sedangkan SOAP sebesar 5,3 detik (Kusumaningrum, Sajati, & Anarianto, 2019). Terdapat perbedaan antara REST API dengan SOAP API yang dapat dilihat pada Tabel 2.3.

Tabel 2.3 Perbedaan REST API dan SOAP API

Uraian	**REST API**	**SOAP API**
Protokol komunikasi	HTTP, HTTPS	HTTP, HTTPS, SMTP, FTP
Penggunaan *bandwith*	Relatif hemat *bandwith*, karena *markup-markup* ekstra seperti yang terdapat pada XML tidak terpakai	Jika *request* banyak, relatif boros *bandwith*. Hal ini karena banyaknya *markup* dalam penulisan format XML
Trend penggunaan	Mulai populer dan banyak dipakai seperti twitter, yahoo, flickr, google, amazon, dan lainnya	Masih ada seperti integrasi aplikasi ke sistem legasi sebuah perusahaan, namun banyak yang beralih ke REST.
Aturan penulisan	Tidak ada spesifikasi khusus	Ketat, mengikuti spesifikasi XML (SOAP v1.2)
Format *response*	XML, JSON atau format *plain text* lainnya. Hal ini dapat memudahkan penerima *response* saat membaca dan memahami	XML dengan spesifikasi SOAP agak sulit untuk dibaca dan dipahami
Attachment file	Tidak bisa	Bisa (karena dapat mengembalikan *response* dalam format *binary*)
Sifat *web service*	Terbuka, dapat diakses oleh siapapun (*web* API)	Tertutup, lebih ditunjukan untuk *vendor* atau perusahaan tertentu
Caching web	Mudah, karena menggunakan URI	Relatif sulit
Penggunaan standar	Standar yang sudah ada seperti XML dan HTTP	Standar lama (XML, HTTP) dan baru (SOAP) digunakan bersamaan
Tool pengembangan	Beberapa, karena tidak terlalu dibutuhkan	Banyak, komersial maupun *open source*

Tool manajemen	Menggunakan *tool* yang sudah ada pada sistem jaringan	Perlu, harga terkadang mahal
Ekstensible	Relatif tidak ekstensible	Bisa, banyak ekstensi termasuk standar WS
Kemudahan implementasi	Mudah	Mudah jika sudah memiliki lingkungan dengan basis SOAP

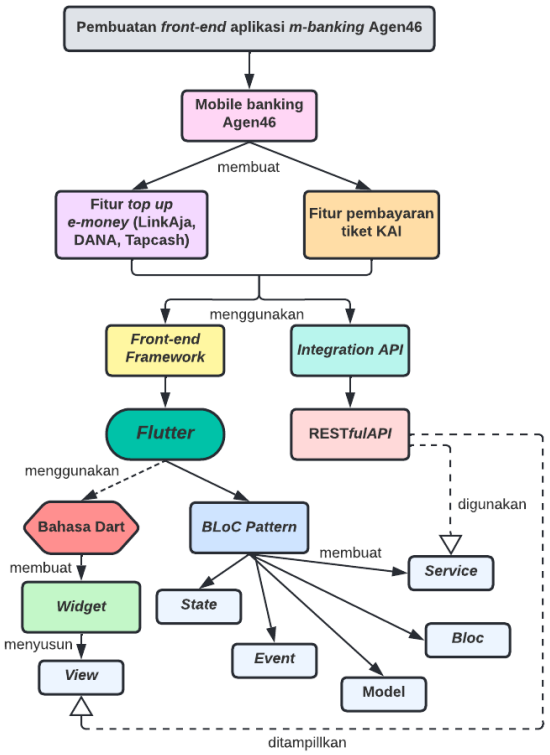
Berdasarkan kelebihan yang dimiliki oleh REST*ful* API sehingga BNI memilih API tersebut dalam pembuatan integrasi API pada aplikasi *m-banking* Agen46. Kemudahan yang dimiliki oleh REST dapat mempermudah pengelolaan alur pengiriman data *request* serta *response* yang terjadi antara *frontend* dengan *backend*. Terdapat juga perbandingan antara REST*ful* API dengan SOAP API berdasarkan performanya yang dapat dilihat pada Tabel 2.4.

Tabel 2.4 Perbandingan REST API dan SOAP API

API	Kelebihan	Kekurangan
REST	• Mudah dipahami dan dibaca oleh penerima response. • Sangat efisien dan cepat. • Bersifat independen. • Terstruktur dan fleksibel (dapat digunakan beragam jenis bahasa pemrograman dan *platform*). • Penggunaan *bandwith* yang lebih hemat. • Banyak digunakan. • Tidak bergantung pada *tools* pengembangan. • *Web service* bersifat terbuka. • Impelementasi lebih mudah dan simple. • *Caching web* lebih mudah karena menggunakan URI.	• Kurang aman karena REST melewati bagian dari *protocol* HTTP dalam penggunaannya. • Tidak dapat digunakan pada sistem terdistribusi (lebih dari satu perantara) karena model bersifat *point-to-point*. • Sangat bergantung pada model *transport* HTTP/HTTPS. • Membutuhkan waktu akses yang cukup lama dibandingkan dengan *native library*. • Kurangnya dukungan standar keamanan, kebijakan, keandalan pesan.
SOAP	• Keamanan yang lebih ketat (WS-*Security*). • Terstandarisasi. • *Transport* yang lebih bebas (HTTP, SMTP, TCP dan lainnya). • Bahasa, *platform*, dan *transport* bersifat independen. • Dapat digunakan pada sistem terdistribusi. • Otomatisasi saat digunakan dengan bahasa tertentu. • Jika terjadi kegagalan, SOAP dapat mengulang *request* dengan *built-in retry logic*. • Memberikan ekstensibilitas yang signifikan dalam bentuk standar WS*.	• Performa kurang bagus. • Kurang fleksible. • Lebih kompleks dan sulit dipahami. • Membutuhkan banyak kode program. • Sulit dikembangkan dan membutuhkan *tools*. • Penggunaan *bandwith* yang boros karena adanya *markup* pada format XML. • *Web service* bersifat tertutup.

BAB III
PELAKSANAAN MAGANG

3.1 Pembuatan *frontend* dan integrasi *API* aplikasi *m-banking* Agen46



Gambar 3.1 Diagram pembuatan fitur

Pembuatan fitur aplikasi mobile banking Agen46 dikerjakan bersama dalam sebuah tim yang terdiri dari *product owner*, *team member* (*frontend mobile engineer*), dan *scrum master*. Manajemen proyek dilakukan dengan menggunakan metode *Agile* dengan sistem manajemen kerja *scrum* dan *weekly sprint planning* sebagai media pelaporan hasil progress pengembangan yang telah dilakukan dan konsultasi terkait kendala yang dihadapi selama proses pengembangan berlangsung. Fitur yang dibuat terdiri dari empat fitur yaitu *top up e-money* (LinkAja, DANA, TapCash) dan pembayaran tiket KAI. Dalam hal ini developer bertugas untuk membuat *frontend* dan integrasi API nya. Diagram pembuatan fitur pada aplikasi m-banking Agen46 dapat dilihat pada Gambar 3.1. Pembuatan dilakukan dalam beberapa tahapan yaitu:

3.1.1 Pendefinisian Proyek

Mobile banking Agen46 sebagai proyek pengembangan aplikasi dengan desain *user interface* yang baru dari aplikasi yang sudah ada sebelumnya. Proyek ini sudah berjalan namun masih dalam proses *development* untuk migrasi dengan *framework Flutter*. Aplikasi yang dikembangkan dengan *Flutter* ini nantinya dapat berjalan sebagai aplikasi berbasis *multiplatform* (iOS dan Android). Aplikasi ini nantinya akan digunakan oleh nasabah untuk melakukan berbagai jenis kegiatan transasaksi dengan *mobile banking*. Selama proses pengerjaan developer diberikan panduan berupa desain UI/UX terbaru dan dokumen spesifikasi API. Desain UI/UX terbaru telah dikerjaan oleh tim UI/UX *designer*, sehingga hanya diberi akses ke dalam *platform* Figma. Selama pengerjaan, juga didampingi oleh mentor, ketua tim proyek BBC, dan senior *frontend engineer* BBC dari PT. Bank Negera Indonesia Tbk.

3.1.2 Inisialisasi Proyek

Pada tahap analisis kebutuhan, spesifikasi proyek Agen46 ditentukan agar pengembangan yang dilakukan dapat sesuai dengan harapan dan tujuan. Dalam hal ini ada beberapa dokumentasi yang mendukung pelaksanaan pengembangan seperti dokumentasi spesifikasi API dan dokumentasi desain UI/UX. Desain tampilan aplikasi sudah dibuat oleh tim UI/UX *designer*. Tampilan fitur ini memiliki perbedaan yang cukup besar dari aplikasi sebelumnya yang sudah ada. Selama proses pengembangan tim *frontend engineer* juga diberi akses kedalam Figma untuk melihat desain aplikasinya. Beberapa teknologi yang digunakan dalam pengembangan fitur dapat dilihat pada Tabel 3.1. Setelah spesifikasi teknologi ditentukan, tahap selanjutnya adalah membuat *git repository* baru dan menerima *git repository* yang telah disiapkan agar dapat membuat fitur dengan *framework* yang telah ditentukan. Pengembangan proyek aplikasi ini terdiri juga terbagi dalam beberapa peran penting, di antaranya adalah:

a. *Product Owner* yang bertanggung jawab serta memastikan bahwa aplikasi yang dibuat oleh *scrum team* memenuhi nilai bisnis dan kebutuhan pengguna. Selain itu *product owner* juga bertugas membagi *backlog* untuk setiap fitur yang dikembangkan oleh *scrum team.*
b. *Team Member/Frontend Mobile Engineer* yang bertugas melakukan eksekusi dan pengembangan aplikasi hingga selesai. Pengembangan yang dilakukan berdasarkan kebutuhan sistem yang telah ditentukan sebelumnya. *Team member* terdiri dari *frontend, backend*, ui/ux *design*, *quality assurance* dan lainnya.

c. *Scrum Master* yang bertugas untuk memastikan bahwa proses *scrum* dalam pengerjaan proyek berjalan dengan lancar. Selain itu *scrum master* juga bertugas memastikan praktik *agile* dijalankan oleh semua anggota tim serta menjadi penghubung antara *product owner* dengan tim pengembang.

Tabel 3.1 Spesifikasi Teknologi *m-banking* Agen46

Aspek	**Teknologi**
Bahasa Pemrograman	Dart 2.14.4
Framework	*Flutter* 2.5.3
Source Code Version	*Gitlab*
Code Editor	*Android Studio*
Deployment Platform	Jenkins

3.1.3 Perencanaan Proyek

Selama proses pengembangan engineer harus melihat panduan dokumentasi spesifikasi API dan desain UI/UX yang sudah diberikan. Apabila terdapat kesalahan ataupun perbedaan data antara bagian *backend* dengan *frontend* maka *frontend mobile engineer* harus berkoordinasi dahulu dengan *product owner*. *Product owner* dalam hal ini adalah *senior engineer*. *Device* yang digunakan untuk meihat hasil pengembangan adalah simulator *device*. Setiap engineer diberikan satu hak akses *username* dan *password* yang unik untuk menjalankan simulasi transaksi yang dilakukan.

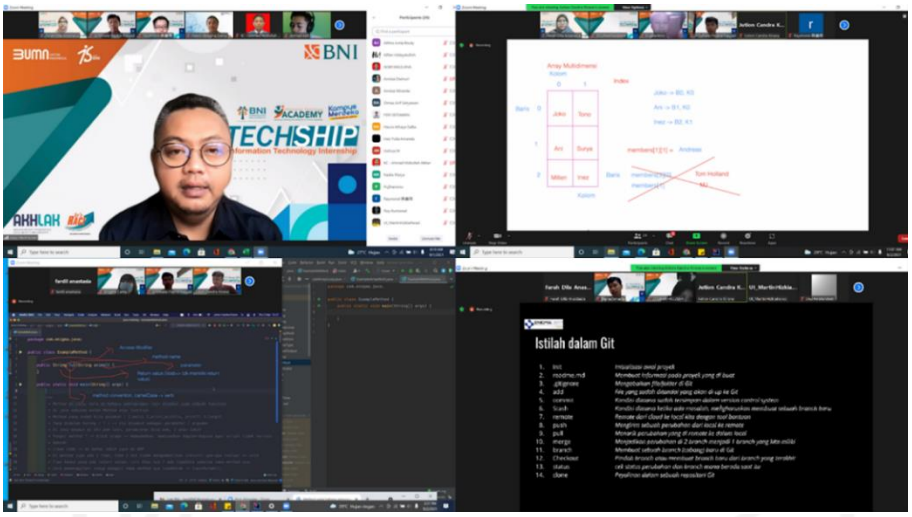
Fitur yang dikembangkan yaitu fitur *top up e-money* dan pembayaran tiket transportasi KAI. Fitur tersebut nantinya akan di*develop* dari mulai pembuatan *frontend* dan pembuatan integrasi API nya. Pada fitur tersebut akan dikembangkan dari mulai *input* kode atau nomor transaksi hingga halaman bukti transaksinya. Selain itu terdapat pembuatan *git repository* yang digunakan untuk mendokumentasikan fitur yang telah dikembangkan. Setiap satu fitur disimpan dalam satu *branch git*. Selama magang hingga bulan Februari ini telah melakukan aktivitas-aktivitas di PT. Bank Negara Indonesia Tbk. Aktivitas-aktivitas tersebut dapat dilihat pada Tabel 3.2. Pembagian waktu tersebut belum termasuk beberapa aktivitas yang berkaitan dengan masa *onboarding*, *bootcamp*, *self learning Flutter*, *meeting* dengan mentor, dan *meeting* dengan ketua tim proyek. Pengerjaan fitur yang dilakukan sebelumnya mendapat *feedback* dari ketua tim proyek, maka perbaikan yang diperlu dilakukan akan menambah waktu pengerjaan.

Tabel 3.2 Pembagian aktivitas magang

No	Aktivitas	Durasi Waktu
1	Mengikuti *bootcamp*	Selama bulan September 2021
2	Mempersiapkan *software* dan teknologi perusahaan.	1 hari pada pekan pertama bulan Oktober 2021
3	Mempelajari *Flutter* secara mandiri	Selama bulan Oktober 2021
4	Mengerjakan tampilan fitur *top up* uang digital LinkAja.	Selama bulan November 2021
5	Mengerjakan integrasi API fitur *top up* uang digital LinkAja.	Selama 3 pekan pertama bulan Desember 2021
6	Melakukan *deployment* sandboxing fitur *top up* uang digital LinkAja ke DigiLabs PT. Bank Negara Indonesia Tbk.	1 hari pada pekan keempat bulan Desember 2021
7	Membuat dokumentasi pengembangan fitur *top up* uang digital LinkAja.	3 hari pada pekan keempat bulan Desember 2021
8	Mengerjakan fitur pembayaran tiket transportasi KAI.	Selama 2 pekan pertama bulan Januari 2022
9	Membuat dokumentasi pengembangan fitur pembayaran tiket transportasi KAI.	Selama 1 hari pekan kedua bulan Januari 2022
10	Mengerjakan fitur *top up* uang digital DANA.	Selama 1 pekan ketiga bulan Januari 2022
11	Membuat dokumentasi pengembangan fitur *top up* uang digital DANA.	3 hari pada pekan keempat bulan Januari 2022
12	Mengerjakan fitur *top up* uang digital Tapcash.	Selama 2 pekan pertama bulan Februari 2022
13	Membuat dokumentasi pengembangan fitur *top up* uang digital Tapcash.	Selama 2 haru pekan ketiga bulan Februari 2022
14	*Meeting* dengan mentor	Setiap 1 bulan sekali
15	*Meeting* dengan ketua tim proyek	Setiap 3 hari sekali dan seminggu sekali
16	*Meeting* dengan *senior engineer*	Setiap terjadi kendala dengan *backend*

Mengikuti *bootcamp*

Sebelum memasuki fase *on-job* pengembangan proyek, peserta diwajibkan untuk mengikuti *bootcamp* selama satu bulan. Pelaksanaan *bootcamp* dilakukan dari tanggal 1-30 September 2021. Bootcamp ini diselenggarakan oleh PT. Bank Negara Indonesia Tbk yang bekerjasama dengan Enigma Camp. Enigma Camp merupakan salah satu penyelenggara pengelolaan *talent* IT yang berfokus pada kesiapan kerja dengan program *training* intensif. Kegiatan *bootcamp* dilakukan secara daring. Kegiatan *bootcamp* dilaksanakan dari Senin hingga Jumat dari pukul 08.00 – 17.00. Waktu istirahat sebanyak dua kali yaitu pada pukul 12.00 – 13.00 dan pukul 15.30 – 16.00 WIB. Kegiatan *bootcamp* dibimbing dua orang *trainer* yaitu Kak Tika Yesi Kristiani dan Kak Jution Candra Kirana.



Gambar 3.2 Kegiatan *bootcamp*

Peserta magang mempelajari beberapa *stack tech* yang nantinya akan digunakan saat magang yaitu *Java Fundamental*, Git, *database* JDBC, HTTP, REST API, *Spring Boot*, REST *Controller*, *Bean Stereotype*, *Apache Kafka*, *Docker*, *React* JS, *Flutter*, dan *Scrum Agile*. Setiap tiga hari peserta diberikan tugas dan *case study* secara berkelompok yang harus diselesaikan sebagai *assessment*. Tugas tersebut akan dikumpulkan pada *Git* milik Enigma Camp. Pada akhir kegiatan *bootcamp* akan diberikan transkrip nilai hasil pembelajaran selama satu bulan melalui email. Kegiatan *bootcamp* dapat dilihat pada Gambar 3.2.

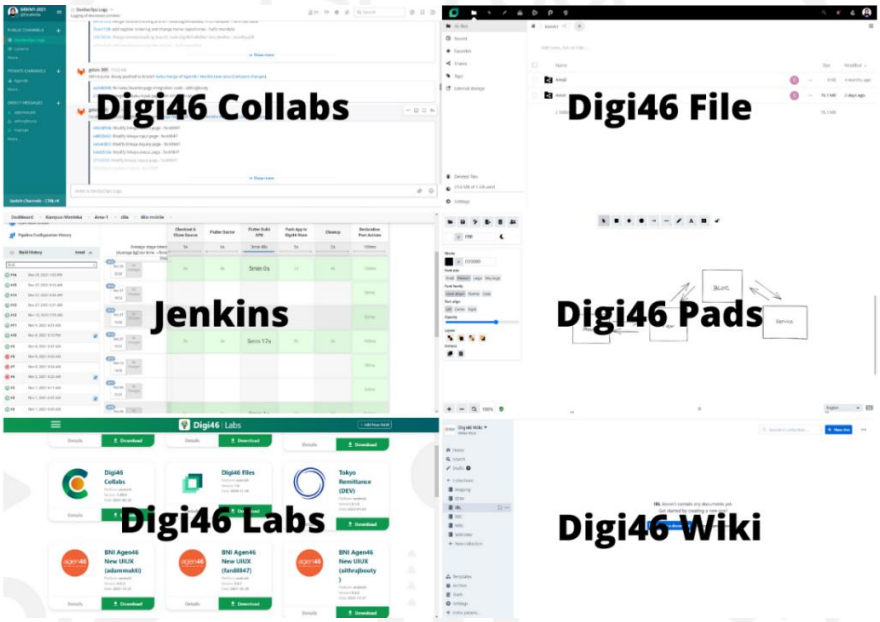
Mempersiapkan *software* dan teknologi perusahaan

Dari 20 peserta magang nantinya akan dibagi lagi menjadi dua mentor yaitu bagian area satu dan bagian area dua. Dari tiap bagian area memiliki mentor dan kewenangan masing-masing yaitu area satu memegang bagian *frontend dan backend* serta area dua memegang wewenang untuk bagian *backend*. Penugasan magang berada di area satu yang dimentori langsung oleh Bapak Taufikurrahman. Sebelum memasuki masa *on job* proyek diadakan *meeting* dengan mentor terkait pembagian proyek dan tim proyek.

Tabel 3.3 *Tools development*

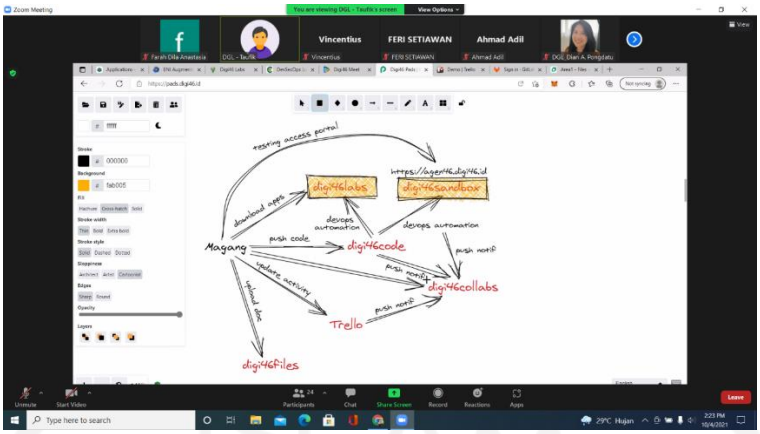
No	Kategori	Keterangan
1	Kolaborasi	Digi46 *Collabs*
		Digi46 *Pads*
		Gitlab
2	*Deployment*	Jenkins
		Digi46 Labs
3	*Code Editor*	Android Studio

4	*Task Management*	Trello
		Excel
5	Komunikasi	Zoom
		Whatsapp
		Digi46 *Collabs*
6	*Framework*	*Flutter*
		Dart
7	Dokumentasi	Digi46 Wiki
		Digi46 *File*
		Canva
		Google *docs*



Gambar 3.3 *Tools* internal milik BNI

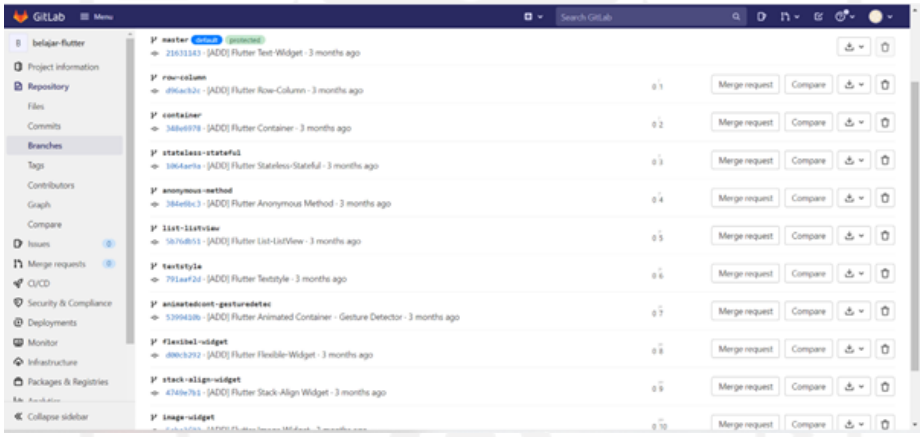
Pada meeting ini juga dijelaskan mengenai *tools-tools* serta teknologi yang nantinya akan digunakan selama proses magang berlangsung. Dalam area satu terdiri dari tim e-Form Pra Kerja, Mobile Remittance, Portal EDM, Ibank Remit, dan Agen46. Ada beberapa tools yang digunakan untuk membantu selama proses *development* yang dapat dilihat pada Tabel 3.3. Selain itu terdapat beberapa *tools* internal milik BNI yang dapat digunakan selama pengembangan. *Tools* internal BNI dapat dilihat pada Gambar 3.3. Setelah diberikan penjelasan terkait *tools* yang akan digunakan, juga dilakukan proses instalasi dan pendaftaran akun ke dalam *tools* tersebut kepada bagian instalasi dan akses management *tools* yaitu Mas Maman. Meeting tersebut juga dijelaskan oleh mentor mengenai budaya kerja, jam kerja, serta harapan mentor kepada peserta magang lainnya selama mengikuti proses pembelajaran mengikuti magang di PT. Bank Negara Indonesia Tbk. *Meeting* dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Meeting* pengenalan *software* dan teknologi perusahaan

Mempelajari *Flutter* secara mandiri

Selama magang diberi waktu selama tiga pekan untuk mematangkan bahasa pemrograman yang nantinya akan digunakan ketika melakukan pengembangan aplikasi.



Gambar 3.5 Proses pembelajaran *Flutter*

Proses pembelajaran ini dilakukan secara mandiri. Mentor turut berperan aktif dalam memberikan sumber referensi pembelajaran. Terdapat kegiatan untuk mempelajari *Flutter* dari berbagai sumber seperti dokumentasi *Flutter*, tutorial *youtube* Pak Erico Darmawan, dan sumber referensi yang diberikan oleh Pak Taufikurrahman. Selama proses pembelajaran juga turut aktif mempraktikkan setiap materi yang dipelajari. Dalam pemrograman Dart terbagi menjadi beberapa materi yang berkaitan dengan *widget-widget, animation, shared preference, function, null* (*safety & check*), dan HTTP (*request & response*). Proses pembelajaran *Flutter* dapat dilihat pada Gambar 3.5.

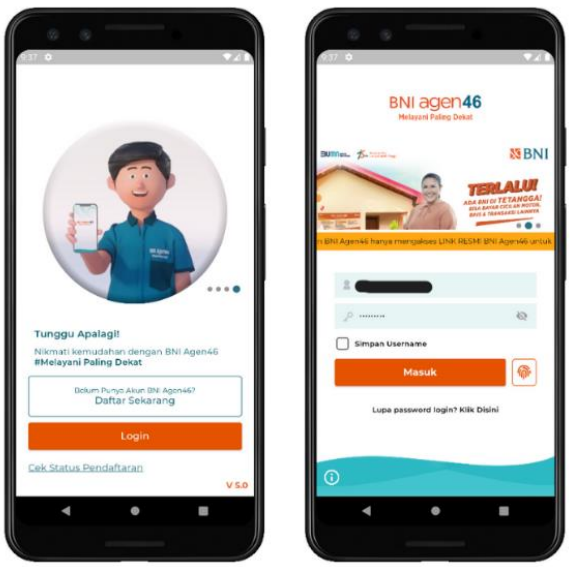
3.1.4 Pelaksanaan Proyek

Tahapan pengerjaan pengembangan fitur aplikasi dengan cara mengimplementasikan desain dan kebutuhan yang diperlukan ke dalam *code* pemrograman. Pada tahap ini *frontend engineer* menerapkan *BloC Pattern* untuk membuat fitur dan membuat integrasi API. Selama proses pengembangan *engineer* harus melihat panduan dokumentasi spesifikasi API dan desain UI/UX yang sudah diberikan. Apabila terdapat kesalahan ataupun perbedaan data antara bagian *backend* dengan *frontend* maka *frontend mobile engineer* harus berkoordinasi dahulu dengan *product owner*. *Product owner* dalam hal ini adalah *senior engineer*. *Device* yang digunakan untuk meihat hasil pengembangan adalah simulator *device*. Setiap *engineer* diberikan satu hak akses *username* dan *password* yang unik untuk menjalankan simulasi transaksi yang dilakukan.

Fitur *top up* LinkAja

a. Halaman *splashscreen* dan *login.*

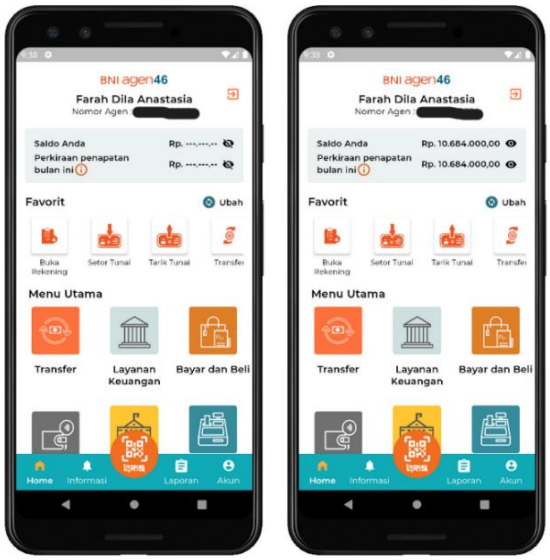
Halaman ini merupakan tampilan pertama saat pengguna membuka aplikasi *m-banking* Agen46. Pada tampilan halaman *splashscreen* pengguna dapat mengklik *button* "*Login*". Setelah mengklik *button* "*Login*" akan diarahkan ke halaman *login*. Pada halaman ini pengguna harus mengisi *username* dan *password*. Setelah mengisi pengguna bisa mengklik *button* "Masuk". Halaman splashscreen dan login dapat dilihat pada Gambar 3.6.



Gambar 3.6 Halaman *splashscreen* dan *login*

b. Halaman utama aplikasi.

Halaman ini merupakan halaman utama saat pengguna berhasil *login*. Pada halaman ini tersedia beberapa informasi dari mulai nama pelanggan, nomor agen, jumlah saldo, dan perkiraan pendapatan per bulan atau saldo yang masuk pada bulan tersebut. Selain itu di bawahnya terdapat menu favorit dan menu utama yang dapat dipilih sesuai dengan jenis transaksi yang diinginkan. Halaman utama aplikasi dapat dilihat pada Gambar 3.7.



Gambar 3.7 Halaman utama aplikasi

c. Menu *top up* uang elektronik.

Menu ini merupakan implementasi dari penambahan percabangan *case* pada *routes.dart*. Pada bagian ini ditambahkan *case* "*linkAjaPayment*" yang akan mengarahkan pengguna menuju halaman yang dituju dengan *class PageRouteBuilder()*. *PageRouteBuilder()* ini berfungsi untuk menentukan *route* halaman yang dituju saat pengguna mengklik yaitu *LinkAjaFavoritPage()*. *BlocProvider* pada dasarnya berfungsi untuk menghubungkan antara *bloc* dengan *user interface*. *BlocProvider()* bertanggung jawab untuk membuat *LinkAjaBloc()* dan *LinkAjaService()*. Hal ini digunakan sebagai *widget dependency injection* (DI) sehingga satu *bloc* dapat diberikan ke beberapa *widget* dalam *subtree*. Kode program menu *top up* uang elekronik dapat dilihat pada Gambar 3.8.

```
case linkAjaPayment:
  return PageRouteBuilder(
    pageBuilder: (context, animation, secondaryAnimation) => BlocProvider(
      create: (context) => LinkAjaBloc(LinkAjaService()),
      child: LinkAjaFavoritPage(),
    ),
    transitionsBuilder context, animation, secondaryAnimation, child){
      var begin = Offset(0.0, 1.0);
      var end = Offset.zero;
```

```
      var curve = Curves.ease;

      var tween =
        Tween(begin: begin, end:end).chain(CurveTween(curve: curve));

      return SlideTransition(
        position: animation.drive(tween),
      child: child,
      );},);
```

Gambar 3.8 Kode program menu *top up* uang elektronik

d. Halaman daftar favorit.

Saat berada pada halaman daftar favorit terdapat *button* "Buat Input Baru". *Button* terbentuk dari *widget ElevatedButton()* yang dibungkus dalam sebuah *Container()*. Di dalam *widget ElevatedButton()* diberikan aksi berupa *onPressed()*. Didalam onPressed() terdapat *class pushToPage()* untuk mengarahkan nasabah menuju halaman *LinkAjaInputPage()* ketika *button* "Buat Input Baru" diklik. Kode program *button* "Buat Input Baru" dapat dilihat pada Gambar 3.9.

```
child: Container(
  width: screenWidth(context),
  height: 40,
  child: ElevatedButton(
    style: ElevatedButton.styleFrom(
      primary: BniTheme.orangeButton,
      onPrimary: Colors.white,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(30.0))),
    onPressed: (){
      pushToPage(
      pageBuilder: BlocProvider(
        create: (context) => LinkAjaBloc(LinkAjaService()),
        child: LinkAjaInputPage (),
      ),);},
    child: Text(
        'Buat Input Baru',
    style: BniTheme.textButtonWhite,
    textAlign: TextAlign.center,
    ),),),
```

Gambar 3.9 Kode program *button* "Buat Input Baru"

Pemanggilan data *user* favorit didapat pada saat pengguna mengeset *user* baru sebagai *user* favorit pada halaman *top up* nominal. Data yang diambil dari *service* berupa nama pelanggan dan nomor transaksi. Data tersebut akan disimpan dalam *widget* bernama *builderDaftarFavorit()*. Saat pengguna melakukan transaksi dengan *user* favorit maka data nama pelanggan dan nomor transaksi akan dikirim langsung ke halaman *LinkAjaTopupPage()* dengan *function pushToPage()*. Pada *function* tersebut terdapat *BlocProvider()* yang akan mengarahkan data tersebut ke *LinkAjaBloc()* dan *LinkAjaService()*. Kode program daftar favorit dapat dilihat pada Gambar 3.10.

```
Widget builderDaftarFavorit(String namaPelanggan, String idPelanggan) {
  return Padding(
    padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 5),
    child: InkWell(
      onTap: () {
        setState(() {
          namaBilling = namaPelanggan;
          nomerBilling = idPelanggan;
        });

        pushToPage(
          pageBuilder: BlocProvider(create: (context) =>
LinkAjaBloc(LinkAjaService()),
            child: LinkAjaTopupPage(noHp: null, idPelanggan: nomerBilling,
namaPelanggan: namaBilling,  isFromFavorit: true,) ,)
        );},
```

Gambar 3.10 Kode program daftar favorit

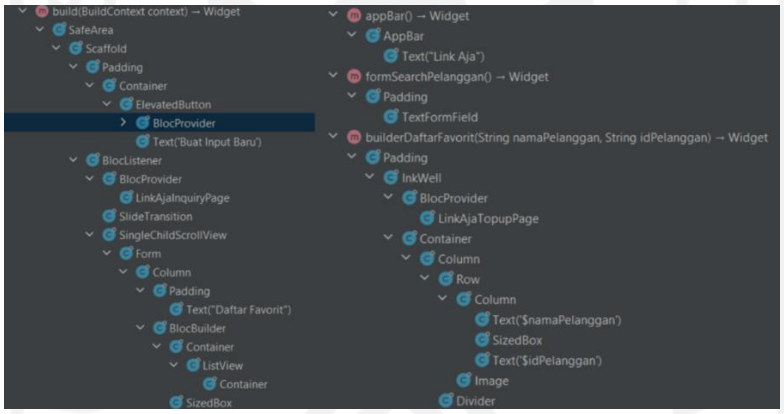
Halaman daftar favorit terdapat pada fitur top up LinkAja, DANA, dan TapCash. Ketiga fitur tersebut memiliki tampilan widget yang sama pada daftar favorit. Gambar struktur *widget* halaman daftar favorit dapat dilihat pada Gambar 3.11. Halaman *list* daftar favorit merupakan implementasi dari *widget* yang di *build* dalam sebuah *context* dimana *context* tersebut tersimpan dalam sebuah *class SafeArea()*. *Class SafeArea()* di dalam *Flutter* berfungsi agar tampilan tampak fleksibel menyesuaikan *device* yang digunakan. Di dalam class *SafeArea()* terdapat turunan *child class* berupa *Scaffold()*. *Scaffold()* berfungsi sebagai *widget* utama untuk membuat halaman pada *Flutter*. Struktur *frontend* pada halaman ini terdiri dari empat *widget* yaitu *BuildContext(), appBar(), formSearchPelanggan(), builderDaftarFavorit()*. Didalam *widget BuildContext()* tersusun dari *class SafeArea()* dan *Scaffold()*.

Pada *widget appBar()* terdapat *class Text()*. Selai itu terdapat penambahan *property elevation(), backgroundColor(), iconTheme()*, dan *title()*. *Class Text()* berfungsi untuk menuliskan judul berdasarkan nama metode *top up* seperti “LinkAja” . Teks tersebut menggunakan *style BniTheme* dan berwarna *blackSemiBold*. *Widget BuildContext()* terbentuk *class ElevatedButton(). ElevatedButton* digunakan untuk membuat *button* berjudul “Buat Input Baru” yang terbungkus pada *class Container()*. Pada *button* tersebut terdapat *class BlocProvider()* yang berfungsi sebagai penghubung antara *bloc* ke *user interface* menuju halaman selanjutnya yaitu *LinkAjaInputPage()*.

Halaman *LinkAjaInputPage()* berada dalam sebuah aksi *button* bernama *onPressed()* dan *class pushToPage()*. Pada c*lass Scaffold()* terdapat *class BlocListener()* yang berfungsi untuk menjalankan *function* berdasarkan *state* nya. Selain itu pada *class BlocListener()* terdapat *class BlocProvider()* yang berguna sebagai penghubung antara *bloc* ke halaman *LinkAjaInquiryPage()*.

Hal tersebut akan berfungsi ketika nasabah meng-*set user* baru sebagai *user* favorit. *List* daftar *user* favorit terbungkus pada *class SingleChildScrollView()*. Daftar *user* favorit akan tampil dalam *class ListView()*. Data *user* favorit akan dipanggil bantuan *class BlocBuilder()* yang dibuat dalam *widget* terpisah pada *widget builderDaftarFavorit()*.

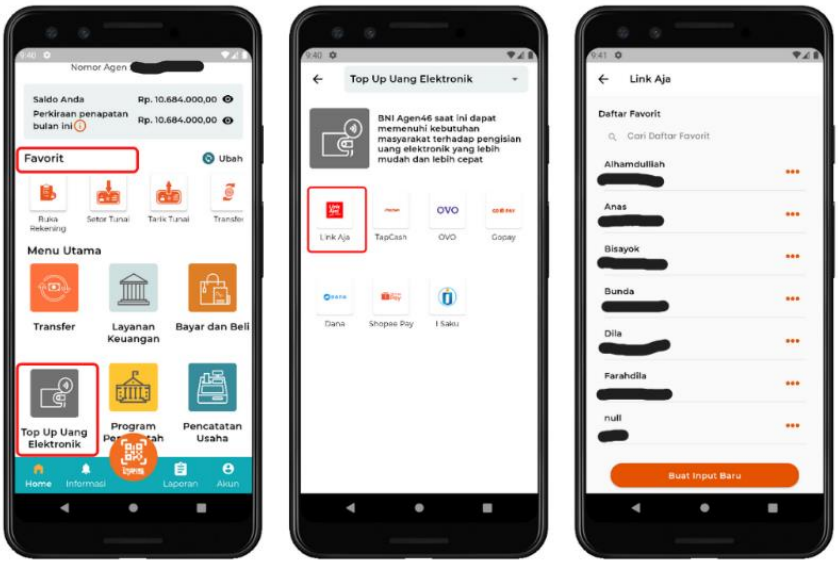
Nama dan nomor hp *user* favorit pada *widget builderDaftarFavorit()* terbuat dari *class Text()*. Teks tersebut tersusun dalam *class Column()* dan *Row()*. Jarak antar data dipisahkan oleh garis dengan *class Divider()*. Pada *class builderDaftarFavorit()* terdapat *class BlocProvider()* untuk mengarahkan *bloc* ke halaman *LinkAjaTopupPage() dengan* aksi *onTap()*. C*lass InkWell()* berfungsi untuk mengarahkan nama *user* favorit yang dipilih ketika pengguna melakukan *top up*. *Widget formSearchPelanggan()* berfungsi untuk mencari data dalam *list* daftar favorit. Bagian ini terbentuk dari *class TextFormField()*. *Class* ini merupakan *form field* yang dapat dikustomisasi dengan menambahkan *property* lain seperti *keyboardType*, *enableInteractiveSelection*, *readOnly*, *maxLines*, *style*, *decoration*, dan *controller*.



Gambar 3.11 Struktur *widget* halaman daftar favorit

Pengguna dapat melakukan *top up* LinkAja dengan cara mengklik menu "*Top Up* Uang Elektronik" pada halaman *home*. Menu tersebut berada pada menu utama ataupun yang ada pada daftar menu favorit pada bagian atas. Ketika pengguna mengklik menu "*Top Up* Uang Elektronik" maka akan diarahkan ke dalam halaman *top up* uang elektronik uang elektronik yang berisi pilihan menu *top up* seperti DANA, LinkAja, Tapcash, dan lain-lain. Setelah itu pengguna bisa mengklik logo bertuliskan metode "LinkAja". Menu "LinkAja" akan mengarahkan ke halaman utama "LinkAja". Pada halaman utama LinkAja terdapat *list* daftar favorit *user* dan nomor transaksi yang telah diset sebagai *user* favorit.

Pengguna bisa menggunakan salah satu *list user* favorit untuk melakukan transaksi dan menggunakan *button* “Buat Input Baru” untuk melakukan transaksi dengan nomor yang belum pernah digunakan sebelumnya. Halaman daftar favorit LinkAja dapat dilihat pada Gambar 3.12.



Gambar 3.12 Halaman daftar favorit LinkAja

e. Halaman *input* nomor transaksi.

Data nomor transaksi diisi dalam sebuah *widget formNoHpNumber()* tepatnya pada *class TextFormField()*. Pada *class TextFormField()* diberi *property validator* untuk memastikan bahwa *form* nomor transaksi tidak bernilai *null*. Selain itu terdapat *property onChanged* untuk memberikan efek perbedaan saat *form* diisi atau saat *form* bernilai *null*. Ketika *form* nomor transaksi berhasil diisi, *isEditedNoHp* bernilai *true* sehingga *button* “Lanjut” berwarna *orange* dan berstatus *enable* untuk diklik. Namun jika *isEditedNoHp* bernilai *false* maka *button* “Lanjut” tidak berwarna dan berstatus *disable* untuk diklik. Kode program *form input* dapat dilihat pada Gambar 3.13.

```
Widget formNoHpNumber() {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 7, horizontal: 20),
    child: TextFormField(
      keyboardType: TextInputType.number,
      enableInteractiveSelection: false,
      readOnly: false,
      maxLines: 1,
      style: TextStyle(fontSize: 14, letterSpacing: 0.5),
      decoration: mainInputDecoration("No Hp", 10, 20),
      validator: (value) {
        if (value == null || value.isEmpty) {
```

```
          return "Harap Masukan No Hp";
        }
        return null;
      },
      onChanged: (value) {
        if (value.isEmpty) {
          setState(() {
            isEditedNoHp = false;
          });
        } else {
          setState(() {
            isEditedNoHp = true;
          });}},
      controller: noHpCtrl,
    ),);}
```

Gambar 3.13 Kode program *form input*

Data nomor transaksi yang telah diisi akan dikirim ke halaman *LinkAjaTopupPage()* dengan *property onPressed()*. Nomor transaksi yang dikirim ke halaman *LinkAjaTopUpPage()*, berdasarkan input pada halaman sebelumnya yaitu *LinkAjaInputPage()*. Nomor tersebut tersimpan pada *variable noHpCtrl.text*. *Variable* namaPelanggan, idPelanggan, dan isFromFavorit disini bernilai *null*. Hal ini dikarenakan data yang akan dikirim ke halaman *LinkAjaTopupPage()* merupakan data *user* baru dari halaman *LinkAjaInputPage()*. Kode program *button* "Lanjut" *input* nomor transaksi dapat dilihat pada Gambar 3.14.

```
Widget buttonLanjutEnable() {
  return ElevatedButton(
    style: ElevatedButton.styleFrom(
        primary: BniTheme.orangeButton, // background
        onPrimary: Colors.white, // foreground
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(30.0))),
    onPressed: () {
      if (_formKey.currentState.validate()) {
        pushToPage(
          pageBuilder: BlocProvider(
            create: (context) => LinkAjaBloc(LinkAjaService()),
            child: LinkAjaTopupPage(
              namaPelanggan: null,
              idPelanggan: null,
              isFromFavorit: false,
              noHp: noHpCtrl.text,
            ),),);}},
    child: Text(
      'Lanjut',
      style: BniTheme.textButtonWhite,
      textAlign: TextAlign.center,
    ),);}
```

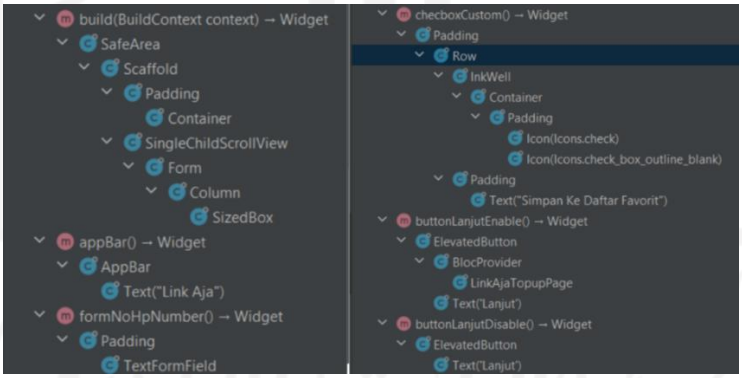
Gambar 3.14 Kode program *button* "Lanjut" *input* nomor transaksi

Halaman input terdapat pada fitur *top up* LinkAja, DANA, Tiket KAI dan TapCash. Ketiga fitur tersebut memiliki tampilan *widget* yang mirip dari segi *form*. Perbedaan nya terletak pada jenis data yang akan diinputkan. LinkAja dan DANA menggunakan nomor *handphone*.

Pembayaran tiket KAI menggunakan kode pembayaran sedangkan TapCash menggunakan nomor kartu. Pada bagian ini terdiri dari beberapa bagian *widget* yaitu *BuildContext(), appBar(), formNoHpNumber(), buttonLanjutEnable(),* dan *buttonLanjutDisable()*. Struktur *widget* halaman *input* dapat dilihat pada Gambar 3.15.

Widget BuildContext() menggunakan *Scaffold()* dan *SafeArea()* untuk mengatur *context* di dalamnya. *Scaffold()* menggunakan *class Container()* untuk membungkus beberapa *widget* seperti *formNoHpNumber(), buttonLanjutEnable()* dan *buttonLanjutDisable()*. *Class Container()* pada halaman ini juga diberi *class Padding()* untuk agar terdapat jarak antara *widget parent* dengan *widget child* nya. Bagian *body* dari halaman ini menggunakan properti *SingleChildScrollView()* dan efek *physics* berupa *BouncingScrollPhysics()* agar konten widget didalamnya dapat di*scroll*.

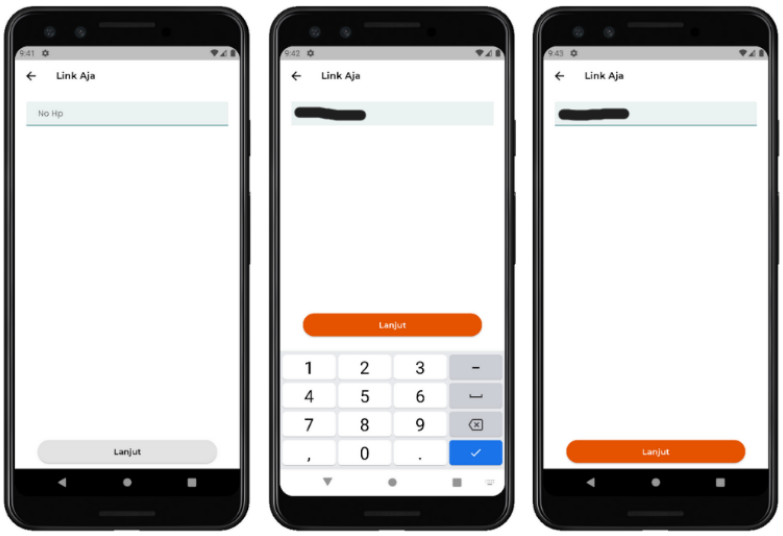
Widget appBar() tersusun dari *class Text()* berjudul sesuai dengan nama fiturnya seperti "Link Aja". *Property elevation*, *backgroundColor*, *iconTheme*, dan *style* ditambahkan untuk mengatur gaya penulisan judul. *Form* yang digunakan adalah *widget TextFormField()*. Pada *widget* ini ditambahkan *property* seperti *padding*, *keyboardType*, *style*, dan *decoration* untuk mengatur tampilan *form* saat diinputkan nomor *handphone*. *Property validator* digunakan untuk memastikan *value* yang diinputkan tidak bernilai *null* dan sesuai dengan format *request* nya.



Gambar 3.15 Struktur *widget* halaman *input*

Button pada halaman ini terbagi menjadi dua kondisi yaitu saat *enable* dan *disable*. *Enable* ketika posisi *form* nomor *handphone* telah terisi sedangkan *disable* ketika *form* nomor *handphone* belum terisi (*null*). Perubahan dari kedua kondisi tersebut terletak pada warna *button*. *Button* yang digunakan adalah *ElevatedButton()*. Button ini juga menambahkan *property style*, *color Primary*/*onPrimary*, *shape*, *borderRadius*, dan *Text*. Pada *widget buttonLanjutEnable()* terdapat aksi *onPressed()*.

Pada aksi *onPressed()* diberi *class pushToPage()* dan class *BlocProvider()* untuk mengarahkan menuju halaman selanjutnya yaitu *LinkAjaTopUpPage()*. Pada *widget buttonLanjuDisable()* terdapat properti *onPressed()* yang berfungsi memberikan *log* info ketika kondisi nya tidak terpenuhi.



Gambar 3.16 Halaman *input* nomor transaksi LinkAja

Jika pengguna telah mengklik *button* "Buat Input Baru" maka akan diarahkan ke halaman *input* nomor transaksi baru. Pengguna dapat mengisikan nomor transaksi yang ingin dituju. Ketika pengguna mengklik area *form* untuk mengisi transaksi, tampilan akan secara otomatis menampilkan *keyboard* bertipe *number*. Jika sudah terisi dengan nomor transaksi pengguna dapat mengklik button "Lanjut". Button "Lanjut" akan berpengaruh terhadap isian *form* nomor transaksi. Jika berhasil diisi maka *button* "Lanjut" akan berstatus *enable* dan dapat diklik. Namun sebaliknya jika *form* tidak terisi maka *button* "Lanjut" berstatus *disable* dan tidak dapat diklik. Setelah mengklik *button* "Lanjut" maka akan diarahkan ke halaman selanjutnya yaitu halaman *top up* nominal. Halaman *input* nomor transaksi LinkAja dapat dilihat pada Gambar 3.16.

f. Halaman *top up* nominal.

1. *Widget button* "Lanjut".

   Semua data *request* yang akan dikirim menuju *service* berawal dari *view*. Berikut merupakan *widget buttonLanjutEnable()* dari *button* "Lanjut" pada halaman *top up* nominal. *Button* tersebut terbentuk dari *class ElevatedButton()*. *Button "Lanjut"* akan mengirim data *top up* nominal dengan property *onPressed()*.

Data yang diinputkan saat pengguna melakukan top up akan dimasukan ke dalam *function getInquiryLinkAja()*. Kode program *widget button* "Lanjut" dapat dilihat pada Gambar 3.17.

```
Widget buttonLanjutEnable() {
  return ElevatedButton(
    style: ElevatedButton.styleFrom(
        primary: BniTheme.orangeButton, // background
        onPrimary: Colors.white, // foreground
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(30.0))),
    onPressed: () {
      if (_formKey.currentState.validate()) {
        getInquiryLinkAja();
      }
    },
    child: Text(
      'Lanjut',
      style: BniTheme.textButtonWhite,
      textAlign: TextAlign.center,
    ),);}
```

Gambar 3.17 Kode program *widget button* "Lanjut"

2. *Function getInquiryLinkAja().*

Berikut merupakan *function getInquiryLinkAja()*. Pada *function* tersebut terdapat *LinkAjaBloc()* yang berfungsi untuk meneruskan *event* ke *service*. Selain itu terdapat *event* yang dikirim berupa *FetchLinkAjaInquiry()*. *Function LinkAjaInquiryParameterPost()* berfungsi sebagai *data class* yang dikirim ketika *request* ke REST*ful* API. Data tersebut dibuat berdasarkan dari format *request inquiry* yang ada pada dokumen spesifikasi API. *Variable* h_msisdn diambil berdasarkan *variable* saat memasukan nomor transaksi yaitu noHpCtrl.text. Kode program *function getInquiryLinkAja()* dapat dilihat pada Gambar 3.18.

```
final linkAjaBloc = BlocProvider.of<LinkAjaBloc>(context);
  linkAjaBloc.add(
    FetchLinkAjaInquiry(LinkAjaInquiryParameterPost(

      // isinya sesuaikan dengan yg ada di LinkAjaInquiryParameterPost
      kode_mitra: kodeMitra,
      h_amount: nominal.toString(),
      h_msisdn: noHpCtrl.text,
      //.. dan data lain
    )),);}
```

Gambar 3.18 Kode program *function getInquiryLinkAja()*

3. LinkAja *bloc.*

LinkAjaBloc() merupakan *bloc* yang bertugas mengirim *event request* ke *service*. Nama *event* yang dikirim *bloc* berupa *LinkAjaEvent()*. Kode program LinAja *bloc* dapat dilihat pada Gambar 3.19.

```
class LinkAjaBloc extends Bloc<LinkAjaEvent, LinkAjaState> {
  LinkAjaBloc(this.linkAjaService)
      : super(LinkAjaIntial());
```

Gambar 3.19 Kode program LinAja *bloc*

4. LinkAja *inquiry event.*

Pada LinkAja *event* terdapat nama *event* yang dikirim ke *service* yaitu *FetchLinkAjaInquiry()*. Selain itu terdapat *service* bernama *LinkAjaInquiryParameterPost()*. *LinkAjaInquiryParameterPost()* merupakan *value* atau data yang dikirim saat meng*hit* ke REST*ful* API. Penggunaan *equitable* dengan *get props* dilakukan untuk memperbarui *state* sesuai dengan perubahan objeknya. *List<Object> get props* berfungsi memastikan apakah *list property* yang dipakai untuk menentukan merupakan dua *instance* yang sama. Kode program LinkAja *inquiry event* dapat dilihat pada Gambar 3.20.

```
class FetchLinkAjaInquiry extends LinkAjaEvent {
  final LinkAjaInquiryParameterPost value;
  FetchLinkAjaInquiry(this.value);

  @override
  List<Object> get props => [value];
}
```

Gambar 3.20 Kode program LinkAja *inquiry event*

5. LinkAja *inquiry service.*

LinkAja *service* merupakan data yang dikirim ketika *bloc* melakukan *request* ke service. Pada *service* tersebut terdapat *UriApi* berupa *linkAjaInquiryUrl* yang berfungsi sebagai alamat *route* untuk mengirim *response* dari *service*. *Request* yang diterima oleh *service* nantinya akan dicocokan oleh *linkAjaInquiryModelFromJson*. Jika *request* sesuai dengan data yang ada pada model maka *service* akan memberi *response* dengan *statusCode* 200. Selain itu *service* akan memberikan *response* berupa data model dari *linkAjaInquiryModelFromJson*. Penggunaan *async* bertujuan untuk memberikan perintah bahwa *code* tersebut dijalankan secara *asynchronous*. Selain itu *await* berfungsi untuk memberikan perintah di dalam *asynchronous* bahwa function *await* harus selesai terlebih dahulu (*suspending function*). Apabila proses yang terjadi pada *await* telah selesai, baru dilanjutkan untuk mengeksekusi kode berikutnya yang ada di dalam *asynchronous*. Kode program LinkAja *inquiry service* dapat dilihat pada Gambar 3.21.

```
Future linkAjaInquiryService(LinkAjaInquiryParameterPost value) async {
  print("RUNNING LINK AJA INQUIRY SERVICE");
  try {
    final response = await dio.post(
      UriApi.linkAjaInquiryUrl,
      options: Options(
        headers: {
          'Content-Type': 'application/json',
        },
        sendTimeout: 15,
      ),
      data: {
        "kode_mitra": value.kode_mitra,
        "h_amount": value.h_amount,
        "h_msisdn": value.h_msisdn,

//.. dan data lain

      },
    );
    log.info("REQUEST = ${response.requestOptions.data}");
    log.info("RESPONSE SERVICE LINK AJA INQUIRY = ${response.statusCode}");
    log.info("DATA : ${response.data}");
    if (response.statusCode == 200) {
      return compute(linkAjaInquiryModelFromJson,
json.encode(response.data));
    }
  } on DioError catch (e) {
    log.warning("ERROR SERVICE LINK AJA INQUIRY =
${e.response.statusCode}");
  }
}

class LinkAjaInquiryParameterPost {
  String kode_mitra;
  String h_amount;
  String h_msisdn;
  //.. dan data lain

  LinkAjaInquiryParameterPost({
    this.kode_mitra,
    this.h_amount,
    this.h_msisdn,
    //.. dan data lain
  });}
```

Gambar 3.21 Kode program LinkAja *inquiry service*

6. LinkAja *url list inquiry.*

Pada halaman *url list* terdapat alamat *UriApi* untuk setiap fitur. Setiap *url* berfungsi untuk mengidentifikasi lokasi *resource* pada *service* dengan fungsinya masing-masing. Fitur *top up* LinkAja memiliki lima bagian yaitu *linkAjaInquiryUrl*, *linkAjaPaymentUrl*, *linkAjaLastTransactionUrl*, *linkAjaSetFavBillingUrl*, dan *linkAjaGetFavBillingUrl*. *linkAjaInquiryUrl* alamat untuk inquiry. *linkAjaPaymentUrl* merupakan alamat untuk *payment*. *linkAjaLastTransactionUrl* alamat saat melakukan riwayat transaksi yang dilakukan. *linkAjaSetFavBillingUrl* alamat saat meng*set* transaksi sebagai *user* favorit dan *linkAjaGetFavBillingUrl*

sebagai alamat saat menampilkan daftar *user* favorit. Kode program LinkAja *url list inquiry* dapat dilihat pada Gambar 3.22.

```
/*----------------------LINK AJA API-----------------------------------*/

static const String linkAjaInquiryUrl = "/linkAjaInquiry";

static const String linkAjaPaymentUrl = "/linkAjaPayment";

static const String linkAjaLastTransactionUrl = "/getTransaksiTerakhir";

static const String linkAjaSetFavBillingUrl = "/setFavoritBilling";

static const String linkAjaGetFavBillingUrl = "/getFavoritBillingList";
```

Gambar 3.22 Kode program LinkAja *url list inquiry*

7. LinkAja *inquiry* model.

Data *request* akan dicocokan oleh *LinkAjaInquiryModel*. Setelah itu *service* akan memberikan data *response* ke *bloc*. Data tersebut akan di *parsing* dari bentuk *String* menjadi bentuk JSON dengan bantuan *Map<String, dynamic>json*. Kegiatan *parsing* tersebut dilakukan agar data yang akan di*generate* menjadi model oleh *service* menjadi lebih cepat dan aman untuk berbagai pemrograman. Method *fromJson()* dalam pemrograman Dart disebut dengan *NamedConstructor*. *Named Constructor* berfungsi untuk mengubah data JSON ke dalam bentuk *class* model. Penggunaan *type dynamic* pada *value Map* digunakan untuk memastikan tipe data yang terdapat pada JSON. Data *response* dibuat berdasarkan format *response inquiry* yang ada pada dokumen spesifikasi API. Kode program LinkAja *inquiry* model dapat dilihat pada Gambar 3.23.

```
LinkAjaInquiryModel linkAjaInquiryModelFromJson(String str)=>
    LinkAjaInquiryModel.fromJson(json.decode(str));

String linkAjaInquiryModelToJson(LinkAjaInquiryModel data){
  return json.encode(data.toJson());
}
      class LinkAjaInquiryModel{
        bool error;
        String message;
        String reason;
        String hCustomerName;
        int totalAmount;

//.. dan data lain

        LinkAjaInquiryModel({
          this.error,
          this.message,
          this.reason,
          this.hCustomerName,
          this.totalAmount,
```

```
  //.. dan data lain
      });

factory LinkAjaInquiryModel.fromJson(
    Map<String, dynamic> json) =>
    LinkAjaInquiryModel(
      error:  json["error"] == null ? null : json ["error"],
      message: json["message"] == null ? null : json ["message"],
      reason: json["reason"] == null ? null : json["reason"],
      hCustomerName: json["h_customerName"] == null ? null : json
["h_customerName"],
      totalAmount: json["total_amount"] == null ? null :
json["total_amount"],

      //.. dan data lain
    );

Map<String, dynamic> toJson() => {
  "error" : error == null ? null : error,
  "message" : message == null ? null : message,
  "reason" : reason == null ? null : reason,
  "h_customerName" : hCustomerName == null ? null : hCustomerName,
  "total_amount" : totalAmount == null ? null : totalAmount,

    //.. dan data lain
};
```

Gambar 3.23 Kode program LinkAja *inquiry* model

8. LinkAja *bloc inquiry.*

Data *response* dari *service* akan disimpan dalam sebuah model bernama *linkAjaInquiryModel*. Data *response* yang dikirim oleh *service* akan ditampung lagi oleh *bloc*. Kemudian LinkAja *bloc* bertugas mengecek apakah *response* tersebut *failed*, *error*, ataupun *success*. Apabila *response* tersebut *success* maka akan menuju *state* bernama *LinkAjaInquirySukses()*. Kode program LinkAja *bloc inquiry* dapat dilihat pada Gambar 3.24.

```
@override
  Stream<LinkAjaState> mapEventToState(
      LinkAjaEvent event) async* {

    if (event is FetchLinkAjaInquiry) {
      try {
        yield LinkAjaLoading();

        LinkAjaInquiryModel value = await linkAjaService
            .linkAjaInquiryService(event.value);

        if (value != null && value.error == false) {
          log.info("BLOC: SUKSES GET DATA INQUIRY LINK AJA");
          yield LinkAjaDisposeLoading();
          yield LinkAjaInquirySukses(value);
        }…}
```

Gambar 3.24 Kode program LinkAja *bloc inquiry*

9. LinkAja *inquiry state.*

Berikut merupakan *state* untuk setiap *response* dari *service*. *Class LinkAjaInitial()* digunakan untuk saat data awal dimulai. *LinkAjaLoading()* digunakan saat proses menunggu. *LinkAjaDisposeLoading()* digunakan saat *view* dibuang atau selesai. *LinkAjaInquirySukses()* digunakan saat data *inquiry* berhasil dimuat. *LinkAjaInquiryFailed()* digunakaan saat data *inquiry* gagal untuk dimuat dan *LinkAjaInquiryError()* digunakan saat terjadi kesalahan pada data *inquiry* akan dimuat. Kode program LinkAja *inquiry state* dapat dilihat pada Gambar 3.25.

```
class LinkAjaState extends Equatable {
  @override
  List<Object> get props => [];
}

class LinkAjaIntial extends LinkAjaState {}

class LinkAjaLoading extends LinkAjaState {}

class LinkAjaDisposeLoading extends LinkAjaState {}

class LinkAjaInquirySukses extends LinkAjaState {
  final _data;

  LinkAjaInquirySukses(this._data);
  LinkAjaInquiryModel get value => _data;

  @override
  List<Object> get props => [_data];
}
```

Gambar 3.25 Kode program LinkAja *inquiry state*

10. LinkAja *top up* nominal *view.*

Setelah dicek berdasarkan *state* nya, *bloc* akan mengirim *state* ke *view*. Apabila berstatus *success* maka *state* yang dikirim bernama *LinkAjaInquirySukses()*. Data tersebut akan ditampilkan pada halaman selanjutnya yaitu *LinkAjaInquiryPage()*. Data yang muncul pada *LinkAjaInquiryPage()* berupa nominal, nomor transaksi, nama pelanggan, *set* simpan favorit dan data model nya. Kode program *state inquiry success* dapat dilihat pada Gambar 3.26.

```
body: BlocListener<LinkAjaBloc, LinkAjaState>(
  listener: (context, state) {
        if (state is LinkAjaInquirySukses) {
      Future.delayed(Duration(seconds: 2), () {
        pushToPage(
          pageBuilder: BlocProvider(
            create: (context) =>
                LinkAjaBloc(LinkAjaService()),
            child: LinkAjaInquiryPage(
              nominalCtrl: nominalCtrl.text.toString().replaceAll(',', ''),
              idPelanggan: noHpCtrl.text,
              namaPelanggan: widget.isFromFavorit ? widget.namaPelanggan :
```

```
namaPelangganCtrl.text,
              isSimpanDaftarFavorit: isSaveFavorit,
              dataModel: state.value,
            ),),);});}},
```

Gambar 3.26 Kode program *state inquiry success*

Halaman top up nominal terdapat pada fitur *top up* LinkAja, DANA, dan TapCash. Ketiga fitur tersebut memiliki tampilan *widget* yang sama. Perbedaan nya terletak pada jenis data yang akan tampil. LinkAja dan DANA menampilkan nomor *handphone* sedangkan TapCash menampilkan nomor kartu. Struktur *frontend* halaman ini terbagi menjadi beberapa *widget* yaitu *BuildContext(), appBar(), infoNomorHp(), formNominal(), checkboxCustom(), formNamaPelanggan(), buttonLanjutEnable(), buttonLanjutDisable(), widgetPilihanNominal(),* dan *buttonNominal()*. Struktur *widget* halaman *top up* nominal dapat dilihat pada Gambar 3.27.

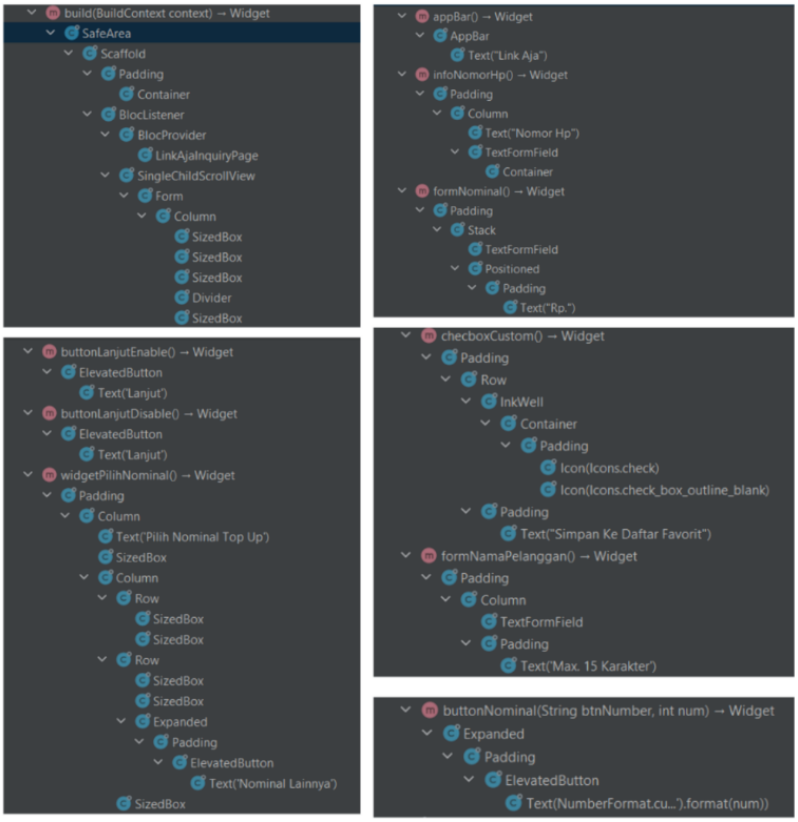
Widget BuildContext terdapat *class BlocListener()* untuk menampilkan *state* dan *BlocProvider()* untuk mengarahkan *bloc* ke halaman konfirmasi yaitu *LinkAjaInquiryPage()*. Di dalam *class SingleChildScrollView* terdapat *class Form()* yang tersusun atas beberapa *class Column()*. *Class Column()* ini berfungsi membungkus beberapa widget di dalamnya yaitu *widget infoNomorHp(), checkboxCustom(), formNamaPelanggan(),* dan *widgetPilihanNominal()*. Setiap *widget* dibungkus dalam sebuah *class* bernama *SizedBox()*. *Class Divider()* berfungsi memisahkan *class formNamaPelanggan()* dengan class *widgetPilihanNominal(). Widget appBar()* tersusun atas *class AppBar()* dan *Text()*. Pada *class AppBar()* terdapat properti *elevation*, *backgroundColor*, *iconTheme,* dan *title* untuk mengatur *style* penulisan judul fitur. *Class Text()* digunakan untuk menulis judul fitur seperti "Link Aja".

Widget infoNomorHp() tersusun dari *class Column()*. *Column* tersebut berisi *class Text()*. Teks pada *form* ditulis berdasarkan fitur yang dibuat. Fitur *top up* LinkAja dan DANA menggunakan judul "Nomor Hp". Fitur pembayaran tiket menggunakan judul "Kode Bayar" sedangkan fitur *top up* TapCash menggunakan "Nomor Kartu". *Class TextFormField()* berfungsi sebagai tempat untuk menampilkan data yang dikirim dari halaman *input*. Pada *class* ini juga terdapat *icon image* untuk menampilkan *icon* dari setiap fitur.

Widget formNominal() terbuat dari c*lass Stack()*. *Class Stack()* berfungsi agar dua buah *widget* saling menumpuk. Widget yang menumpuk yaitu *button dan TextFormFiled.*

Form akan muncul ketika pengguna mengklik button “Nominal Lainnya”. Perubahan kondisi *button* dibuat dengan sebuah *property validator* dan aksi *onChanged*.

Widget checkboxCustom() tersusun dari *class Row()*. Pada *class Row()* terdapat *class Icon()* yang akan muncul saat pengguna menceklis bagian *checkbox*. *Class Text()* digunakan untuk menulis teks berjudul “Simpan Ke Daftar Favorit”. Di dalam *class Row()* terdapat *class InkWell()* dan aksi *onTap*. *Class* tersebut berfungsi sebagai tempat berjalannya *function* ketika *checkbox* di klik.



Gambar 3.27 Struktur *widget* halaman *top up* nominal

Widget formNamaPelanggan() terletak di bawah *widget checkboxCustom()*. *Widget* ini berisi *class TextFormField()* yang digunakan sebagai tempat menginputkan nama *user* favorit. Pada class tersebut terdapat *class Text()* yang berisi teks berjudul “Max. 15 Karakter”. Teks ini digunakan untuk mengingatkan pengguna bahwa nama *user* favorit yang dibuat tidak boleh melebihi 15 karakter.

Widget buttonLanjutEnable() terbuat dari *class ElevatedButton()*. Button “Lanjut” memiliki *property style* berupa *onPrimary* dan *primary*. *Primary* merupakan warna yang digunakan ketika *button* aktif yaitu *orange*. Sebaliknya *onPrimary* merupakan warna *button* ketika pasif yaitu putih.

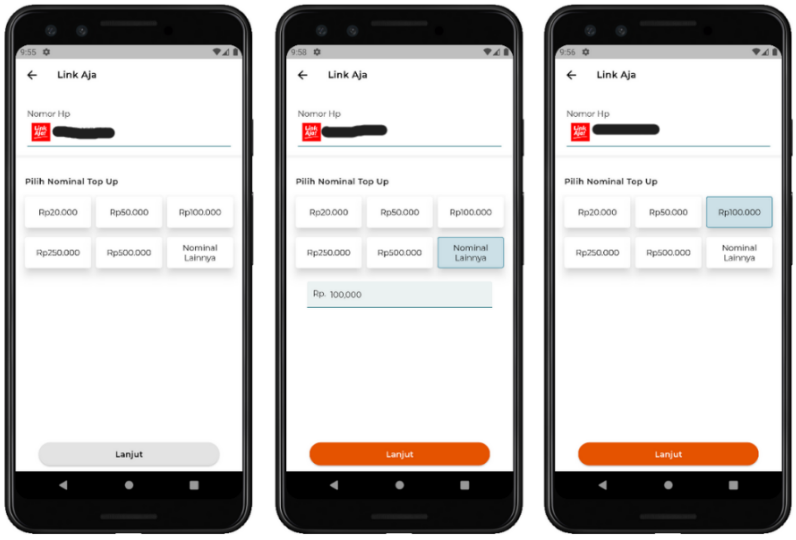
Pada *class ElevatedButton()* terdapat aksi *onPressed* yang berfungsi untuk mengarahkan menuju halaman konfirmasi transaksi. Selain itu *class ElevatedButton()* merupakan tempat *function getInquiryLinkAja()* berada.

Widget buttonLanjutDisable() memiliki *style* yang sama dengan *widget buttonLanjutEnable()*. Perbedaan *widget* ini terletak pada warna *primary button* yaitu putih. Aksi *onPressed* pada *class ElevatedButton()* bertugas menampilkan status *log* info bahwa *button disable*. Kondisi tersebut menyebabkan *button* tidak dapat diklik "Lanjut" apabila tidak terisi nominal *top up* nya.

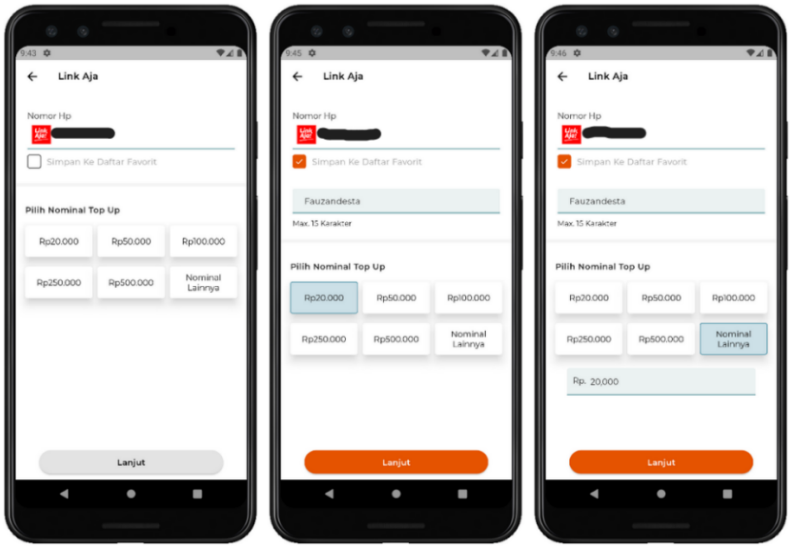
Widget widgetPilihNominal() tersusun dari *class Column()*. Di dalam *class Column()* terdapat *class Text()* bertuliskan "Pilih Nominal Top Up" dan beberapa *class SizedBox()* yang tersusun di dalam *class Row()*. *Class Row()* berisi *widget buttonNominal()*. *Widget buttonNominal()* terbuat dari *class ElevatedButton()*. Pada class ElevatedButton() terdapat aksi *onPressed* dan *number format*. Format tersebut berfungsi untuk memastikan nominal yang dipilih memiliki *value* yang sama antara *button* pilihan nominal dengan input nominal pada *class TextFormField()*.

Pada halaman *top up* nominal pengguna bisa menggunakan dua metode yaitu dengan mengklik *button* "Buat Input Baru" atau dengan memilih salah satu nama *user* favorit. Apabila pengguna menggunakan *button* "Buat Input Baru" maka akan diarahkan ke halaman input nomor transaksi terlebih dahulu. Sebaliknya apabila pengguna mengklik salah satu nama *user* favorit maka akan diarahkan langsung ke halaman *top up* nominal. Pada halaman *top up* nominal dengan *user* favorit tidak terdapat checkbox untuk mengset *user* favorit. Pada halaman *top up* nominal pengguna bisa memilih nominal *top up* yang akan dikirim menuju nomor transaksi yang dipilih. Pada bagian atas halaman tertera nomor transaksi yang dituju.

Pilihan nominal *top up* terdapat dua cara yaitu melalui pilihan *button* nominal yang tersedia seperti Rp 20.000, Rp 50.000, Rp 100.000, Rp 250.000, dan Rp 500.000. Apabila pengguna ingin memasukan jumlah nominal *top up* sendiri maka bisa mengklik *button* "Nominal Lainnya". Pada saat pengguna mengklik *button* "Nominal Lainnya" maka akan muncul *pop up* form untuk mengisi jumlah nominal *top up*. Pada halaman ini pengguna juga bisa mengklik *checkbox* untuk meng*set user input* baru sebagai *user* favorit. Saat mengklik *checkbox* maka akan muncul *pop up* untuk menampilkan *form*.



Gambar 3.28 *Top up* nominal dengan *user* favorit LinkAja



Gambar 3.29 *Top up* nominal dengan *set* favorit

Form tersebut digunakan untuk mengisi nama pengguna dengan batas maksimal 15 karakter. Pengisian nominal *top up* juga berpengaruh terhadap *status enable button* "Lanjut". Jika sudah berhasil memilih nominal yang diinginkan *button* "Lanjut" akan aktif dan berwarna *orange*. Sebaliknya jika nominal belum diisi maka *button* "Lanjut" akan berstatus *disable* dan berwarna abu-abu. *Button* tersebut akan mengarahkan ke halaman konfirmasi transaksi. *Top up* nominal dengan *user* favorit LinkAja dapat dilihat pada Gambar 3.28. *Top up* nominal dengan *set* favorit LinkAja dapat dilihat pada Gambar 3.29.

g. Halaman konfirmasi transaksi.

1. *Widget button* “Lanjut”.

Berikut merupakan *widget buttonNavigasi()* dari *button* “Lanjut” pada halaman konfirmasi. *Button* tersebut terbentuk dari *class Inkwell()*. *Button “Lanjut”* akan mengirim data dari *LinkAjaInquiryModel* dan *password* yang diinputkan dengan property *onTap()*. Data yang diinputkan saat pengguna melakukan konfirmasi akan dimasukan ke dalam *function onPaymentLinkAja()*. Kode program *button* “Lanjut” konfirmasi dapat dilihat pada Gambar 3.30.

```
InkWell(
  onTap: () {
    if (_formKey.currentState.validate()) {
      onPaymentLinkAja();
    }
  },
  child: Container(
    width: screenWidth(context) / 2.5,
    height: 40,
    decoration: BoxDecoration(
        borderRadius: BorderRadius.all(
          Radius.circular(10),
        ),
        color: HexColor("E55300")),
    child: Center(
      child: Text(
        'Lanjut',
        style: BniTheme.textButtonWhite,
      ),),),),
```

Gambar 3.30 Kode program *button* “Lanjut” konfirmasi

2. *Function onPaymentLinkAja().*

Berikut merupakan *function onPaymentLinkAja ()*. Pada *function* tersebut terdapat *LinkAjaBloc()* yang berfungsi untuk meneruskan *event* ke *service*. Selain itu terdapat *event* yang dikirim berupa *LinkAjaPaymentEvent()*. *Function LinkAjaPaymentParameterPost()* berfungsi sebagai *data class* yang dikirim ketika *request* ke REST*ful* API. Data tersebut dibuat berdasarkan dari format *request payment* yang ada pada dokumen spesifikasi API. *Variable* pin_transaksi diambil berdasarkan *variable* saat memasukan nomor transaksi yaitu pinCtrl.text. Kode program *function onPaymentLinkAja()* dapat dilihat pada Gambar 3.31.

```
final linkAjaPaymentBloc = BlocProvider.of<LinkAjaBloc>(context);

linkAjaPaymentBloc.add(
    LinkAjaPaymentEvent(LinkAjaPaymentParameterPost(
      kodeMitra: kodeMitra,
      hMsisdn: widget.dataModel.hMsisdn,
      hAmount: widget.nominalCtrl ,
```

```
      totalAmount: widget.dataModel.totalAmount,
      pin_transaksi: pinCtrl.text,

      //.. dan data lain
      )),);}
```

Gambar 3.31 Kode program *function onPaymentLinkAja()*

3. LinkAja *bloc.*

LinkAjaBloc() merupakan *bloc* yang bertugas mengirim *event request* ke *service*. Nama *event* yang dikirim *bloc* berupa *LinkAjaEvent()*. Kode program LinkAja *bloc* dapat dilihat pada Gambar 3.32.

```
class LinkAjaBloc extends Bloc<LinkAjaEvent, LinkAjaState> {
  LinkAjaBloc(this.linkAjaService)
      : super(LinkAjaIntial());
```

Gambar 3.32 Kode program LinkAja *bloc*

4. LinkAja *payment event.*

Pada LinkAja *event* terdapat nama *event* yang dikirim ke *service* yaitu *LinkAjaPaymentEvent()*. Selain itu terdapat *service LinkAjaPaymentParameterPost()*. *LinkAjaPaymentParameterPost()* merupakan *value* atau data yang dikirim saat meng*hit* ke REST*ful* API. Kode program LinkAja *payment event* dapat dilihat pada Gambar 3.33.

```
class LinkAjaPaymentEvent extends LinkAjaEvent {
  final LinkAjaPaymentParameterPost value;
  LinkAjaPaymentEvent(this.value);

  @override
  List<Object> get props => [value];
}
```

Gambar 3.33 Kode program LinkAja *payment event*

5. LinkAja *payment service.*

LinkAja *service* merupakan data yang dikirim ketika *bloc* melakukan *request* ke service. Pada *service* tersebut terdapat *UriApi* berupa *linkAjaPaymentUrl* yang berfungsi sebagai alamat *route* untuk mengirim *response* dari *service*. *Request* yang diterima oleh *service* nantinya akan dicocokan oleh *linkAjaPaymentModelFromJson*. Jika *request* sesuai dengan data yang ada pada model maka *service* akan memberi *response* dengan *statusCode* 200. Selain itu *service* akan memberikan *response* berupa data model dari *linkAjaPaymentModelFromJson*. Kode program LinkAja *payment service* dapat dilihat pada Gambar 3.34.

```
Future linkAjaPaymentService(LinkAjaPaymentParameterPost value) async {
  print(" RUNNING LINK AJA PAYMENT SERVICE");
  try {
    final response = await dio.post(UriApi.linkAjaPaymentUrl,
      options: Options(
        headers: {
          'Content-Type': 'application/json',
        },
        sendTimeout: 15,
      ),
      data: {
        "kode_mitra": value.kodeMitra,
        "h_msisdn": value.hMsisdn,
        "h_amount": value.hAmount,
        "total_amount": value.totalAmount,
        "pin_transaksi": value.pin_transaksi,
 //.. dan data lain
      },
    );
    log.info("REQUEST SERVICE GET HERE : ${response.requestOptions.data}");
    log.info("RESPONSE SERVICE GET CODE : ${response.statusCode}");
    log.info("DATA RESPONSE GET : ${response.data}");
    print("RESPONSE PAYMENT LINK AJA HERE");
    if (response.statusCode == 200) {
      if (response.data["error"] != null ||
          response.data["error"] == true ||
          response.data["errorNum"] != null) {
        return Left(errorPaymentModelFromJson(json.encode(response.data)));
      } else {
        return
Right(linkAjaPaymentModelFromJson(json.encode(response.data)));
      }
    }
  } on DioError catch (e) {
    log.warning("ERROR LINK AJA PAYMENT SERVICE =
${e.response.statusCode}");}}
```

Gambar 3.34 Kode program LinkAja *payment service*

6. LinkAja *url list payment.*

Berikut merupakan url list payment. Kode program *url list payment* dapat dilihat pada Gambar 3.35.

```
static const String linkAjaPaymentUrl = "/linkAjaPayment";
```

Gambar 3.35 Kode program *url list payment*

7. LinkAja *payment* model.

Data *request* akan dicocokan oleh *LinkAjaPaymentModel*. Setelah itu *service* akan memberikan data *response* ke *bloc*. Data tersebut akan di *parsing* dari bentuk *String* menjadi bentuk JSON dengan bantuan *Map<String, dynamic>json*. Data *response* dibuat berdasarkan format *response payment* yang ada pada dokumen spesifikasi API. Kode program LinkAja *payment* model dapat dilihat pada Gambar 3.36.

```
LinkAjaPaymentModel linkAjaPaymentModelFromJson(String str) =>
    LinkAjaPaymentModel.fromJson(json.decode(str));

String linkAjaPaymentModelToJson(LinkAjaPaymentModel data) =>
    json.encode(data.toJson());

class LinkAjaPaymentModel {
  LinkAjaPaymentModel({
    this.trxType,
    this.status,
    this.data,
    this.result,
    this.customerData,

});

  String trxType;
  String status;
  Data data;
  Result result;
  CustomerData customerData;

  factory LinkAjaPaymentModel.fromJson(Map<String, dynamic> json) =>
      LinkAjaPaymentModel(
        trxType: json["trxType"] == null ? null : json["trxType"],
        status: json["status"] == null ? null : json["status"],
        data: json["data"] == null ? null : Data.fromJson(json["data"]),
        result: json["result"] == null ? null :
Result.fromJson(json["result"]),
        customerData: json["CustomerData"] == null ? null :
CustomerData.fromJson(json["CustomerData"]),
      );

  Map<String, dynamic> toJson() => {
        "trxType": trxType == null ? null : trxType,
        "status": status == null ? null : status,
        "data": data == null ? null : data.toJson(),
        "result": result == null ? null : result.toJson(),
        "CustomerData": customerData == null ? null :
customerData.toJson(),
      };}

class Data {…}

class Result {…}

class CustomerData {..}
```

Gambar 3.36 Kode program LinkAja *payment model*

8. LinkAja *bloc payment.*

Data *response* dari *service* akan disimpan dalam sebuah model bernama *linkAjaPaymentModel*. Data *response* yang dikirim oleh *service* akan ditampung lagi oleh *bloc*. Kemudian LinkAja *bloc* bertugas mengecek apakah *response* tersebut *failed*, *error*, ataupun *success*. Apabila *response* tersebut *success* maka akan menuju *state LinkAjaPaymentSukses()*. Kode program LinkAja *bloc payment* dapat dilihat pada Gambar 3.37.

```
if (event is LinkAjaPaymentEvent) {
  yield LinkAjaLoading();
  try {
    var response = await linkAjaService
        .linkAjaPaymentService(event.value);
    log.warning(response);

    log.warning("BLOC OK");
    yield LinkAjaDisposeLoading();
    yield response.fold(
       …
      ) else {
        print("SUKSES PAYMENT LINK AJA");
        return LinkAjaPaymentSukses(data);
      }),);
  …
```

Gambar 3.37 Kode program LinkAja *bloc payment*

9. LinkAja *payment state*.

Berikut merupakan *state* untuk setiap *response* dari *service*. *LinkAjaPaymentSukses()* digunakan saat data *payment* berhasil dimuat. *LinkAjaPaymntFailed()* digunakaan saat data *payment* gagal untuk dimuat dan *LinkAjaPaymentError()* digunakan saat terjadi kesalahan pada data *payment* akan dimuat. Kode program *payment state* dapat dilihat pada Gambar 3.38.

```
class LinkAjaPaymentSukses extends LinkAjaState {
  final _data;
  LinkAjaPaymentSukses(this._data);
  LinkAjaPaymentModel get value => _data;

  @override
  List<Object> get props => [_data];
}
```

Gambar 3.38 Kode program *payment state*

10. LinkAja konfimasi *view*.

Setelah dicek berdasarkan *state* nya, *bloc* akan mengirim *state* ke *view*. Apabila berstatus *success* maka *state* yang dikirim *LinkAjaPaymentSukses()*. Data tersebut akan ditampilkan pada halaman selanjutnya yaitu *LinkAjaStatusPage()*. Kode program *state payment success* dapat dilihat pada Gambar 3.39.

```
if (state is LinkAjaPaymentSukses) {
  print(state.value);
  Future.delayed(Duration(seconds: 2), () {
    pushReplacement(
      pageBuilder: BlocProvider<LinkAjaBloc>(
        create: (context) =>
            LinkAjaBloc(LinkAjaService()),
        child: LinkAjaStatusPage(
          dataPayment: state.value,
          nominalCtrl: widget.nominalCtrl,
          idPelanggan: widget.idPelanggan,
          namaPelanggan: widget.namaPelanggan,
```

```
          isSimpanDaftarFavorit: widget.isSimpanDaftarFavorit,
          reffNum: widget.dataModel.reffNum,
        ),),);});}
```

Gambar 3.39 Kode program *state payment success*

Halaman konfirmasi transaksi terdapat pada fitur *top up* LinkAja, DANA, TapCash dan pembayaran tiket KAI. Ketiga fitur tersebut memiliki tampilan *widget* yang sama. Perbedaan nya terletak pada jumlah data yang akan tampil saat dikonfirmasi. Data pada fitur pembayaran tiket KAI memiliki jumlah yang lebih banyak dibandingkan dengan fitur *top up*. Hal ini dikarenakan data pada fitur pembayaran tiket KAI terdapat informasi detail mengenai data pemesanan kereta nya. Halaman ini tersusun dari *widget BuildContext(), containerDataTransaksi(), containerDataPayment(), buttonNavigasi(), sessionExpiredWidget(),* dan *pushReplacement()*. Halaman ini menggunakan *class SafeArea()* dan *Scaffold()* untuk bagian *context* nya. Stuktur *widget* halaman konfirmasi transaksi dapat dilihat pada Gambar 3.40.

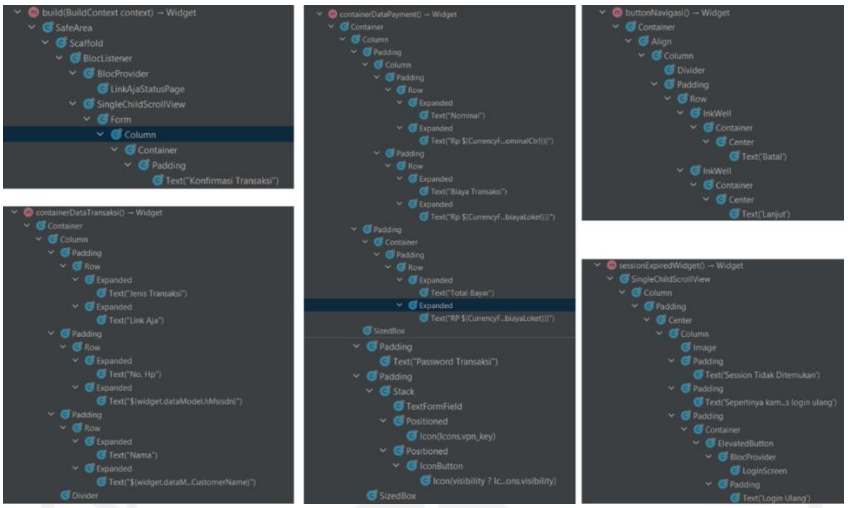
Pada *widget BuildContext()* terdapat *class BlocListener()* dan *Bloc Provider()* untuk menampilkan *state context.* Selain itu *bloc* tersebut bertugas mengarahkan pengiriman data yang dibuat dari *inquiry* ke halaman bukti transaksi. *Class SingleChildScrollView()* digunakan untuk membungkus *widget containerDataTransaksi()* dan *containerDataPayment()*. Widget tersebut tersusun dalam *class Column()*. Pada bagian atas *class Container()* terdapat *class Text()* berjudul "Konfirmasi Transaksi" yang diberi jarak oleh *class Padding()*.

Widget containerDataTransaksi() berisi data transaksi yang dilakukan. Data transaksi tersusun dalam *class Row()* dan diberi *class Expanded()* untuk memperluas *row* yang dibuat. Setiap data memiliki ukuran letak yang sama. Hal ini dikarenakan penggunaan property *flex* dalam *class Expanded()*.

Widget containerDataPayment() berisi data nominal yang harus dibayarkan. Data nominal tersusun dalam *class Row()* dan *Expanded()*. Pada bagian ini terdapat *class Stack()*. Stack digunakan untuk menumpuk *class TextFormField()* tempat mengisi *password* transaksi. Selain itu terdapat *property validator* yang berguna untuk memastikan bahwa *password* transaksi tidak *null*.

Widget buttonNavigasi() digunakan untuk menaruh *button* "Lanjut" dan "Batal". Kedua *button* tersebut menggunakan *class Inkwell()* dan aksi *onTap()* yang bertugas mengarahkan halaman ketika salah satu *button* diklik. *Widget sessionExpiredWidget()* berguna untuk memastikan nasabah bahwa *session* berada dalam *state* yang benar.

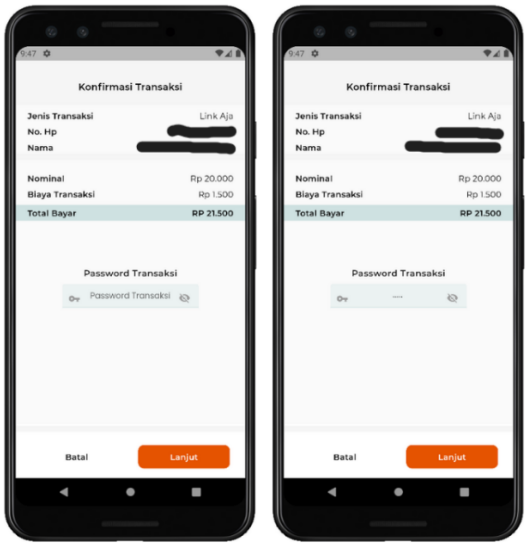
Apabila *session* tidak ditemukan maka nasabah akan diberi informasi *log* untuk melakukan *login* ulang.



Gambar 3.40 Stuktur *widget* halaman konfirmasi transaksi

Widget pushReplacement() digunakan sebagai *navigator widget* ketika proses konfirmasi yang dilakukan berhasil maka akan mengarahkan ke halaman terakhir yaitu *LinkAjaStatusPage()*. *Class SlideTransition()* digunakan untuk memberikan efek *animation* ketika berpindah ke halaman selanjutnya.

Pada halaman konfirmasi pengguna dapat melihat informasi konfirmasi *top up* yang dilakukan yaitu data *payment* dan data transaksi. Detail data transaksi berupa jenis transaksi, nomor *handphone* dan nama pemilik nomor *handphone*.



Gambar 3.41 Halaman konfirmasi transaksi LinkAja

Data *payment* berupa nominal biaya yang harus dibayar yaitu total penjumlahan antara biaya administrasi dan nominal asli *top up*.

Pada halaman ini juga terdapat bagian untuk mengisi *password* transaksi dan apabila telah selesai mengisi *password* pengguna bisa mengklik *button* “Lanjut”. Apabila proses transaksi *top up* yang dilakukan berhasil dan *password* transaksi yang diinputkan benar maka akan mengarah ke halaman akhir dari proses yaitu bukti transaksi. Pengguna dapat membatalkan *top up* yang dilakukan bisa mengklik *button* “Batal” untuk kembali ke halaman *top up* nominal. Halaman konfirmasi transaksi LinkAja dapat dilihat pada Gambar 3.41.

h. Halaman bukti transaksi.

Halaman bukti transaksi bertugas untuk menampilkan seluruh data *response* berupa *linkAjaPaymentModel()*. Data tersebut akan tampil pada *widget detailInformasi()*. Data yang muncul pada *LinkAjaStatusPage()* berupa data *payment*, nominal, nomor transaksi, nama pelanggan, *set* simpan favorit dan data model nya. Kode program *response* bukti transaksi dapat dilihat pada Gambar 3.42.

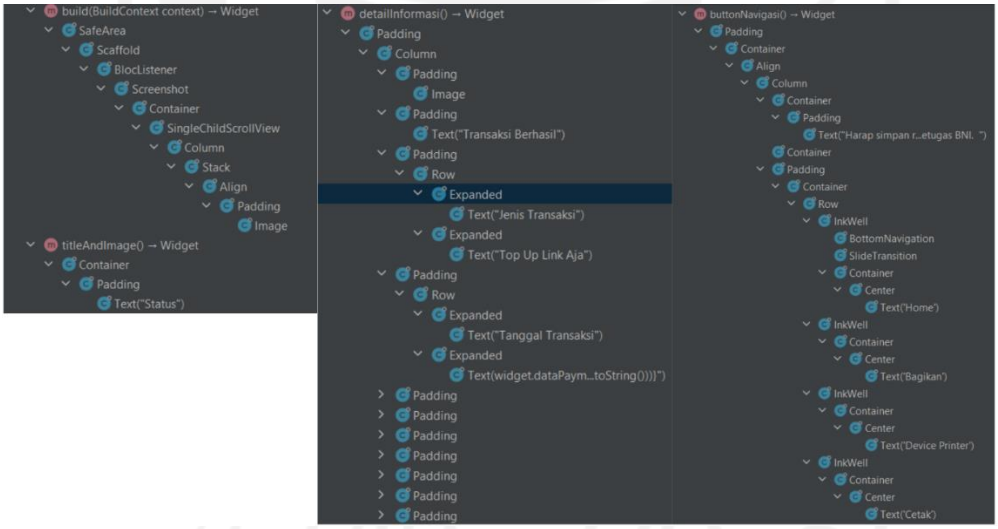
```
Widget detailInformasi() {
  return Padding(
    padding: const EdgeInsets.only(top: 15, left: 20, right: 20),
    child: Column(
      children: [
        Padding(
          padding: const EdgeInsets.only(top: 20),
          child: Image.asset(
            “assets/images/agen_46_logo.png”,
            width: 150,
          ),
        ),

Padding (…)

Padding (…)

Padding (…)

Padding (…)

Padding(
  padding: const EdgeInsets.only(top: 10),
  child: Row(
    children: [
      Expanded(
        flex: 2,
        child: Text(
          “No. Hp”,
          style: BniTheme.text14blackSemiBold,
        ),),
      Expanded(
        flex: 2,
        child: Text(
          “${widget.dataPayment.data.hMsisdn}”,
          style: BniTheme.text14black,
          textAlign: TextAlign.right,
        ),),],),),
```

```
Padding (…)…
```

Gambar 3.42 Kode program *response* bukti transaksi

Halaman bukti transaksi terdapat pada fitur *top up* LinkAja, DANA, TapCash dan pembayaran tiket KAI. Ketiga fitur tersebut memiliki tampilan *widget* yang sama. Perbedaan nya terletak pada jumlah data yang akan tampil pada bukti transaksi. Data pada fitur pembayaran tiket KAI memiliki jumlah yang lebih banyak dibandingkan dengan fitur *top up*. Halaman ini terdiri dari *widget BuildContext(), titleAndImage(), detailInformasi(),* dan *buttonNavigasi()*. Halaman ini menggunakan *class SafeArea()* dan *Scaffold()* untuk menampilkan *context* didalamnya. Struktur *widget* halaman bukti transaksi dapat dilihat pada Gambar 3.43.

Widget BuildContext() tersusun dari *class BlocListener()* yang berfungsi menampilkan status penyimpanan *user* favorit. Selain itu widget ini juga berfungsi untuk membungkus *widget* lainnya seperti *buttonNavigasi(), titleAndImage(),* dan *detailInformasi()*. Tampilan *background* di *widget detailInformasi()* menggunakan *class Stack()* untuk meletakkan *class Imageassets()* sebagai *watermark* dari logo bank BNI.

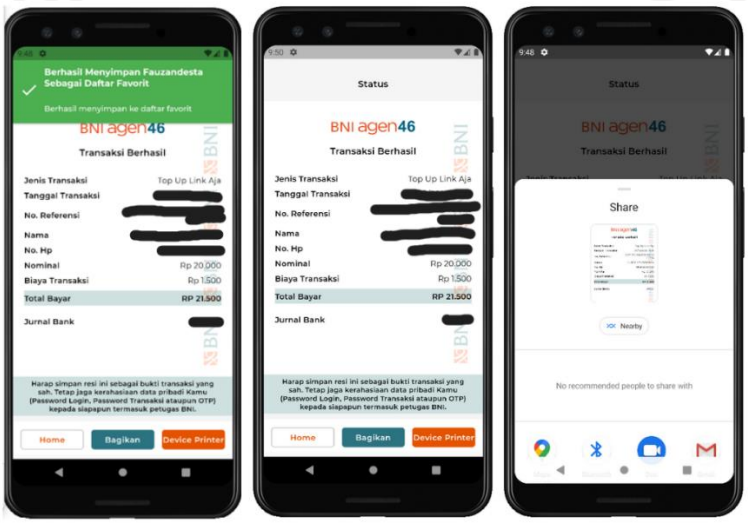


Gambar 3.43 Struktur *widget* halaman bukti transaksi

Widget titleAndImage() berfungsi untuk meletakan penulisan judul dari status transaksi. *Style* penulisan judul tersebut menggunakan *BniTheme. Widget detailInformasi()* tersusun dari properti *Imageasset()* untuk logo BNI Agen46, *class Text()* untuk judul status transaksi, dan *class Column()* untuk informasi detail transaksi yang dilakukan. Informasi detail transaksi berada dalam sebuah *class Row(). Class* tersebut diberi jarak antar data dengan *class Expanded()*.

Widget buttonNavigasi() tersusun dari *class Text()* sebagai tempat untuk menuliskan perintah penyimpanan resi transaksi. Selain itu terdapat *class ButtonNavigation()* pada bagian bawahnya. *Class ButtonNavigation()* terdiri dari tiga bagian yaitu "Bagikan", "Cetak" dan "*Home*". Setiap bagian navigasi menggunakan *class InkWell()* yang berfungsi untuk mengarahkan ke *event* yang dituju.

Halaman bukti transaksi menampilkan beberapa informasi detail terkait *top up* yang dilakukan yaitu data transaksi dan data *payment*. Berikut beberapa informasi yang tertera yaitu: jenis transaksi, tanggal transaksi, nomor referensi, nama, nomor hp, nominal, biaya transaksi, total bayar, dan jurnal bank.

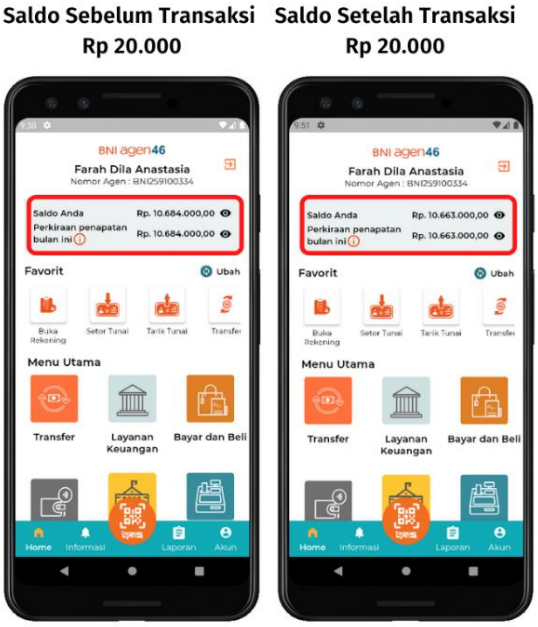


Gambar 3.44 Halaman bukti transaksi LinkAja

Pada halaman ini juga menampilkan status *log* aktivitas jika pengguna menyimpan data baru sebagai *user* favorit. Ada tiga aktivitas yang dapat diarahkan oleh *button* yaitu membagi bukti transaksi, mencetak bukti transaksi dengan *device* lain, dan kembali ke halaman utama aplikasi. Halaman bukti transaksi LinkAja dapat dilihat pada Gambar 3.44.

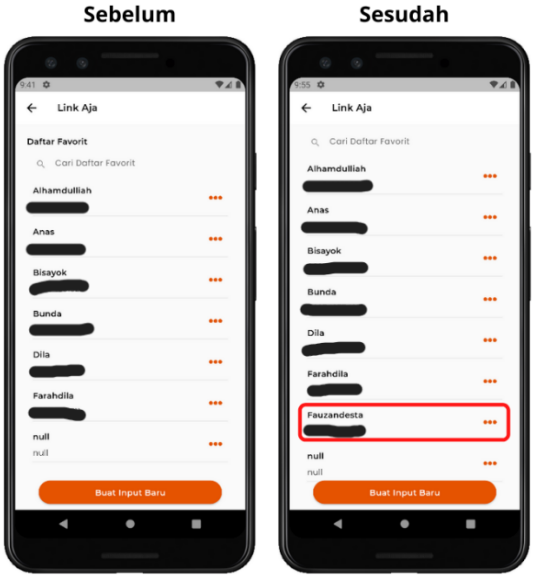
i. Pengecekan pengurangan saldo LinkAja.

Pada halaman *home* pengguna dapat mengecek saldo yang tersisa . Pengecekan tersebut dapat digunakan untuk memastikan proses integrasi API antara *backend* dan *frontend*. Hal tersebut terbukti berhasil setelah melakukan *top up*. Pada informasi bagian atas terdapat pengurangan saldo setelah melakukan *top up* sebesar Rp 20.000. Saldo awal sebesar Rp 10.684.000 berkurang menjadi Rp 10.663.000. Pengecekan pengurangan saldo LinkAja dapat dilihat pada Gambar 3.45.



Gambar 3.45 Pengecekan pengurangan saldo LinkAja

j. Pengecekan daftar *user* favorit LinkAja.



Gambar 3.46 Pengecekan daftar *user* favorit LinkAja

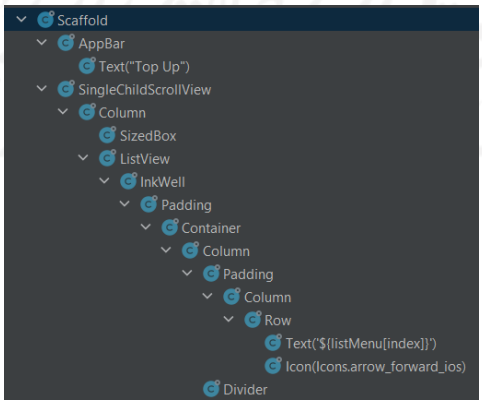
Pengguna dapat mengecek daftar *user* favorit pada halaman uatama LinkAja. Pengecekan dilakukan untuk memastikan keberhasilan aplikasi saat pengguna mengeset *user* baru sebagai *user* favorit. Setelah melaukan transaksi dengan mengeset *user* favorit terjadi penambahan data pada halaman utama LinkAja. Hal tersebut membuktikan bahwa penambahan *user* favorit berhasil dilakukan. Pengecekan daftar *user* favorit LinkAja dapat dilihat pada Gambar 3.46.

Fitur pembayaran tiket KAI

Halaman daftar menu terdapat pada fitur pembayaran tiket KAI dan *top up* uang elektronik TapCash. Keduanya memiliki persamaan dalam implementasi *widget*. Sebelum pengguna masuk ke halaman daftar favorit, aplikasi akan menampilkan *list* menu terlebih dahulu. Secara *frontend* halaman ini terdiri dari widget *AppBar()* dan *widget SingleChildScrollView()*. Gambar struktur *widget* halaman daftar menu dapat dilihat pada Gambar 3.47. Kedua *widget* tersebut berada dalam sebuah *widget* utama yaitu *widget Scaffold()*.

Widget AppBar() tersusun dari *class Text()*. *Class* inilah yang digunakan sebagai *title* berjudul "*Top Up*" untuk fitur *top up* TapCash dan "Kereta Api Indonesia (KAI)" untuk fitur pembayaran tiket. Pada bagian *AppBar* diberi *property backgroundColor* berwarna putih. Selain itu terdapat *elevation* sebesar 0. *Elevation* berfungsi untuk mengatur ukuran bayangan pada *AppBar* tersebut. Pada *class Text()* diberi *property style BniTheme* dengan warna *blackSemiBold*.

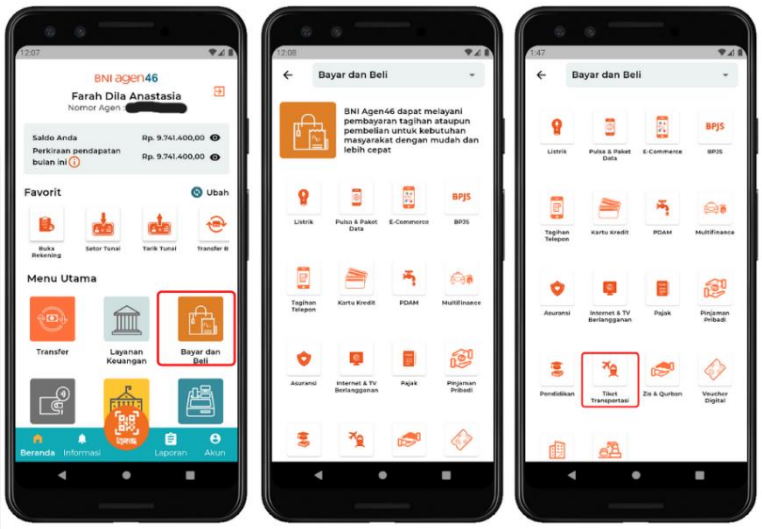
Widget SingleChildScrollView() terletak pada bagian *body*. *Widget* tersebut memiliki turunan *child* yaitu *Column()*. *Class Column()* ini bertugas untuk membungkus *class ListView.builder()*. *ListView* tersebut terbungkus dalam sebuah *class* bernama *SizedBox()*. Pada *class ListView* terdapat *property itemCount* yang menyimpan pilihan menu lainnya. *Property* lain pada *ListView* seperti *shrinkWrap* dan *NeverScrollableScrollPhysics()* berfungsi untuk memastikan agar *list* menu tidak dapat di*scroll* terkecuali terdapat penambahan. Pengaturan nagivasi untuk setiap menu dibuat dengan menambahkan *class InkWell()* pada *ListView.builder()*. Selain itu menu akan diberi aksi *onTap()* untuk mengarahkan menu sesuai dangan indexnya.



Gambar 3.47 Struktur *widget* halaman daftar menu

Setiap menu terdapat *class Padding()* yang berfungsi untuk memberi jarak. *Class Row()* digunakan untuk menaruh *class Text()* dan *class Icon()*. *Class Divider()* digunakan untuk memberikan efek garis pada antar menu dengan ketebalan yang diinginkan.

a. Halaman utama tiket transportasi tiket KAI.



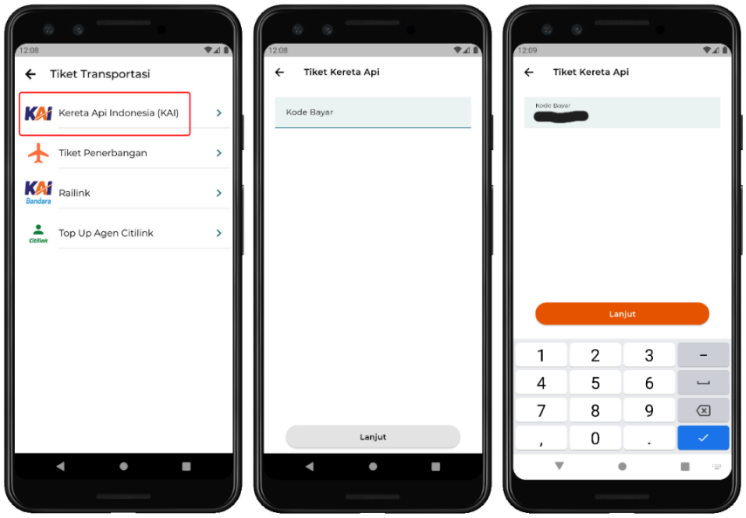
Gambar 3.48 Halaman utama tiket transportasi tiket KAI

Pembayaran tagihan dilakukan dengan cara memilih menu “Bayar dan Beli” pada halaman home. Menu tersebut akan mengarahkan akan mengarahkan ke halaman Bayar dan Beli yang berisi berbagai layanan pembayaran tagihan seperti listrik, e-commerce, BPJS, telepon, pajak, asuransi, pendidikan dan lainnya. Pada halaman ini pengguna dapat mengklik menu “Tiket Transportasi”. Menu “Tiket Transportasi” tersebut akan mengarahkan ke halaman Tiket Transportasi. Halaman utama tiket transportasi tiket KAI dapat dilihat pada Gambar 3.48.

b. Halaman *input* kode bayar tiket KAI.

Pada halaman tiket transportasi terdiri dari empat jenis transportasi yaitu Kereta Api Indonesia (KAI), Tiket Penerbangan, *Railink*, dan *Top Up* Agen Citilink. Pengguna dapat mengklik menu “Kereta Api Indonesia (KAI)”. Menu tersebut akan mengarahkan ke halaman untuk memasukan kode pembayaran tiket KAI. Pengguna dapat menginputkan kode pembayaran dengan mengklik bagian *form*. Ketika pengguna mengklik area *form* kode bayar, secara otomatis aplikasi akan menampilkan *keyboard* bertipe *number*.

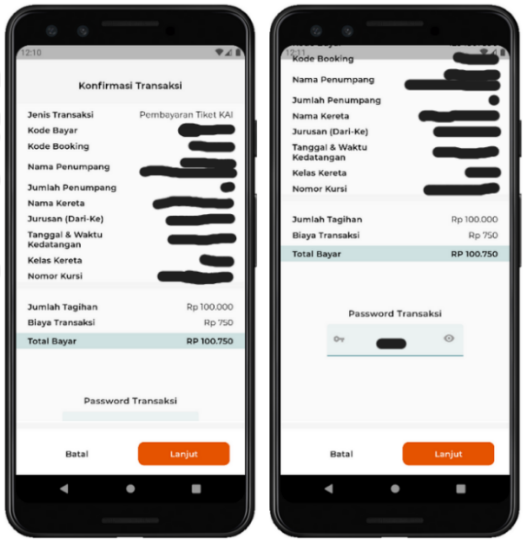
Setelah berhasil mengisi kode pembayaran pengguna dapat mengklik *button* "Lanjut" yang aktif dan berwarna *orange*. *Button* "Lanjut" tersebut akan mengarahkan ke halaman konfirmasi transaksi. Halaman *input* kode bayar tiket KAI dapat dilihat pada Gambar 3.49.



Gambar 3.49 Halaman *input* kode bayar tiket KAI

c. Halaman konfirmasi transaksi tiket KAI.

Pada halaman konfirmasi transaksi terdapat informasi pemesanan kereta api seperti jenis transaksi, kode bayar, kode *booking*, nama penumpang, jumlah penumpang, nama kereta, jurusan (dari-ke), tanggal & waktu kedatangan, kelas kereta, dan nomor kursi. Selain itu terdapat informasi biaya pemesanan seperti jumlah tagihan, biaya transaksi, dan total bayar.

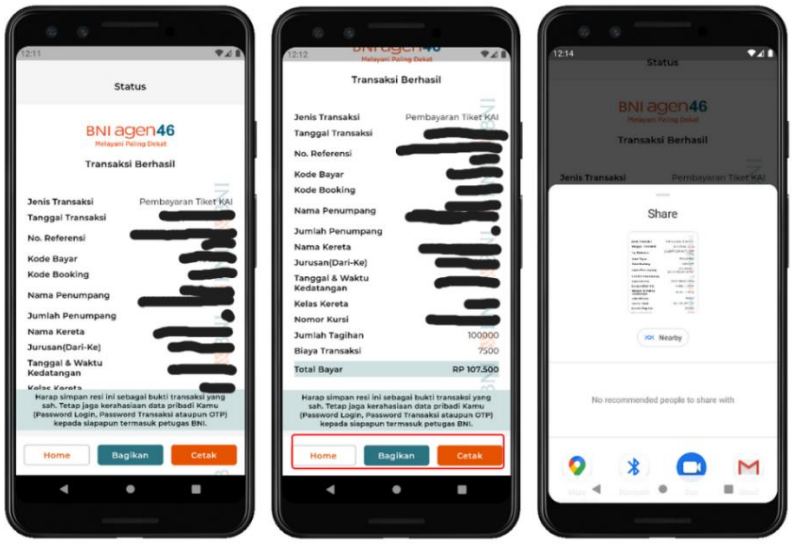


Gambar 3.50 Halaman konfirmasi transaksi tiket KAI

Pada bagian bawah informasi biaya terdapat form untuk mengisi *password* transaksi milik pengguna. Setelah itu pengguna dapat mengklik *button* "Lanjut" untuk menyelesaikan proses pembayaran. Apabila *password* yang diisikan tervalidasi benar maka pengguna akan diarahkan ke halaman bukti transaksi. Sebaliknya jika *password* yang dimasukan salah maka akan muncul *log* peringatan. Pengguna juga dapat membatalkan proses konfirmasi pembayaran dengan mengklik *button* "*Cancel*". *Button* tersebut akan mengembalikan tampilan menuju halaman *input* kode bayar. Halaman konfirmasi transaksi tiket KAI dapat dilihat pada Gambar 3.50.

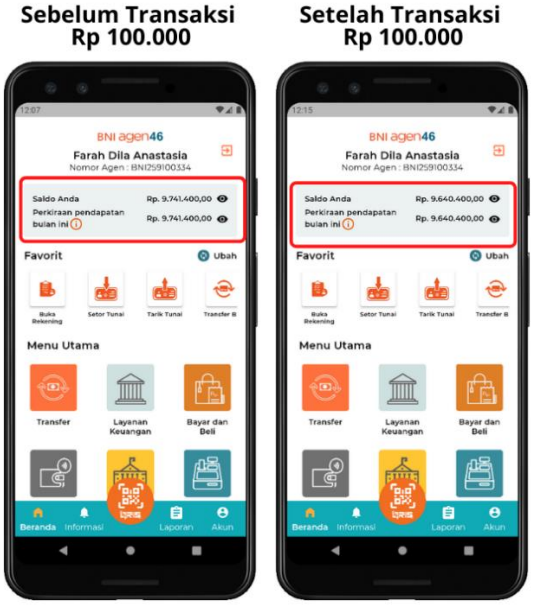
d. Halaman bukti transaksi tiket KAI.

Halaman ini terdapat informasi secara detail mengenai transaksi pembayaran yang telah dilakukan. Informasi tersebut terdiri dari: jenis transaksi, tanggal transaksi, nomor referensi, kode bayar, kode *booking*, nama penumpang, jumlah penumpang, nama kereta, jurusan (dari-ke), tanggal & waktu kedatangan, kelas kereta, nomor kursi, jumlah tagihan, biaya transaksi, dan total bayar. Pada bagian bawah halaman ini terdapat tiga *button*. *Button* "Cetak" digunakan untuk mencetak bukti transaksi. Pencetakan bukti transaksi dapat dilakukan dengan cara menghubungkan *device* lain. *Button* "Bagikan" digunakan untuk membagikan bukti transaksi dengan berbagai metode seperti *email*, *bluetooth*, dan lainnya. Selain itu *button* "*Home*" digunakan untuk mengembalikan tampilan menuju halaman utama. Halaman bukti transaksi tiket KAI dapat dilihat pada Gambar 3.51.



Gambar 3.51 Halaman bukti transaksi tiket KAI

e. Pengecekan pengurangan saldo tiket KAI.



Gambar 3.52 Pengecekan pengurangan saldo tiket KAI

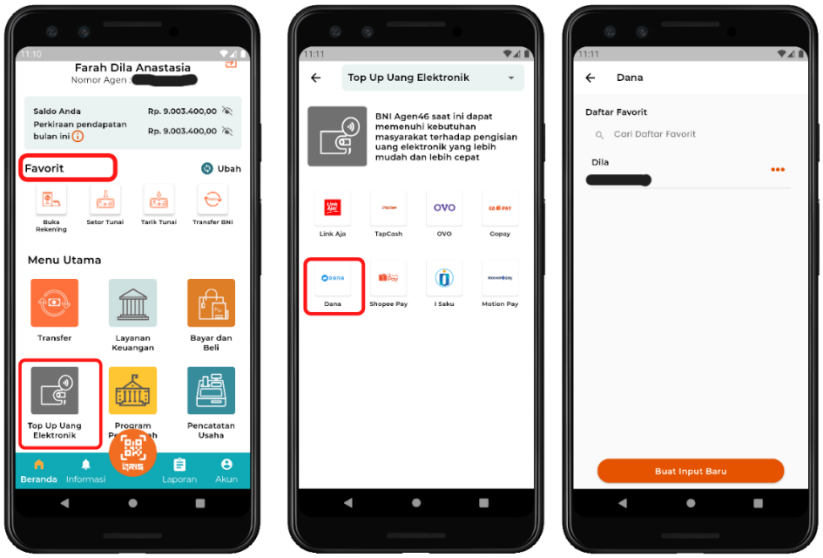
Pada halaman tersebut dapat terlihat bahwa terjadi pengurangan saldo. Hal ini membuktikan bahwa proses integrasi API yang terjadi pada fitur pembayaran tiket KAI berhasil dilakukan. Bagian informasi saldo tertera bahwa terdapat pengurangan saldo setelah melakukan pembayaran tagihan sebesar Rp 100.000. Saldo awal sebesar Rp 9.741.400 menjadi Rp 9.640.400. Pengecekan pengurangan saldo tiket KAI dapat dilihat pada Gambar 3.52.

Fitur *top up* DANA

a. Halaman utama *top up* DANA.

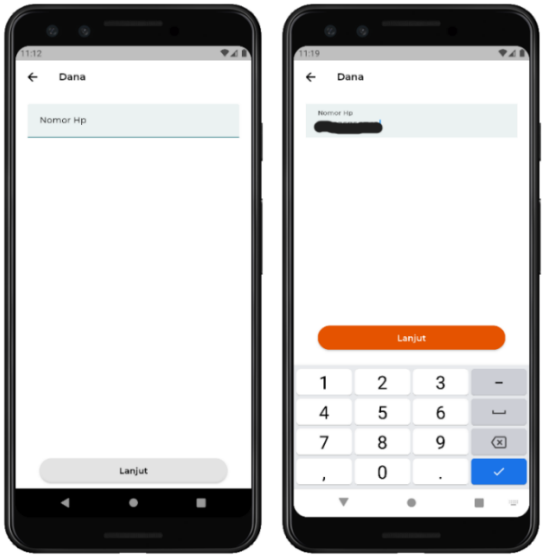
Pengguna dapat melakukan *top up* DANA dengan cara mengklik menu "*Top Up* Uang Elektronik" pada halaman *home*. Menu tersebut berada pada menu utama atau yang ada pada daftar menu favorit pada bagian atas. Ketika pengguna mengklik menu "*Top Up* Uang Elektronik" maka akan diarahkan ke dalam halaman *top up* uang elektronik uang elektronik yang berisi pilihan menu metode *top*. Kemudian pengguna bisa mengklik logo bertuliskan metode "DANA". Menu "DANA" akan mengarahkan ke halaman utama "DANA". Pada halaman utama DANA terdapat *list* daftar favorit *user* dan nomor transaksi yang telah diset sebagai *user* favorit.

Pengguna bisa menggunakan salah satu *list* user favorit untuk melakukan transaksi dan menggunakan *button* “Buat Input Baru” untuk melakukan transaksi dengan nomor yang belum pernah digunakan sebelumnya. Halaman utama *top up* DANA dapat dilihat pada Gambar 3.53.



Gambar 3.53 Halaman utama *top up* DANA

b. Halaman *input* nomor transaksi DANA.



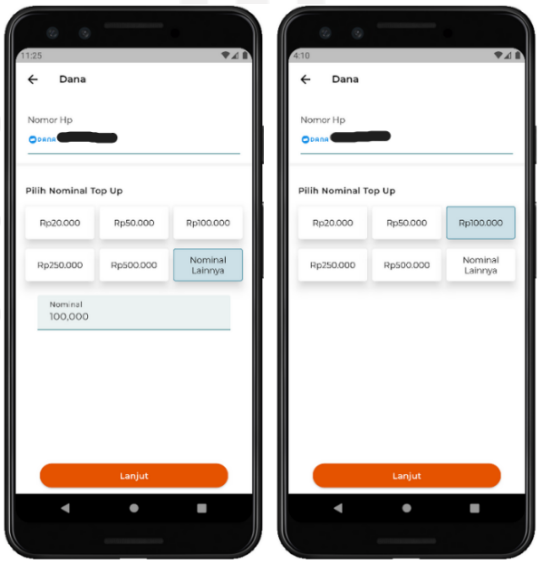
Gambar 3.54 Halaman *input* nomor transaksi DANA

Pada halaman ini pengguna bisa mengisi nomor transaksi baru yang belum pernah terdaftar. Pengguna dapat mengklik *form* pada bagian atas untuk mengisi nomor yang dituju.

Saat pengguna mengklik area *form* tersebut, halaman akan langsung secara otomatis menampilkan *keyboard* bertipe *number*. *Button* "Lanjut" pada bagian bawah akan bergantung pada pengisian *form*. Apabila *form* belum terisi maka *button* "Lanjut" akan berstatus *disable* namun jika *form* sudah terisi maka *button* "Lanjut" akan berstatus *enable* dan berwarna *orange*. *Button* "Lanjut" akan mengarahkan ke halaman selanjutnya yaitu halaman *top up* nominal. Halaman *input* nomor transaksi DANA dapat dilihat pada Gambar 3.54.

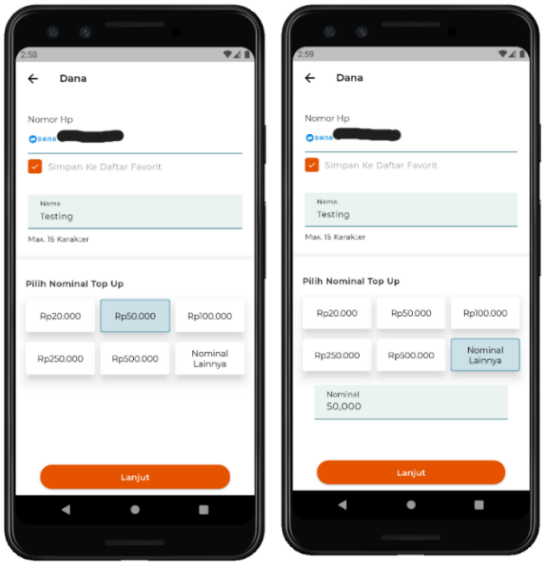
c. Halaman *top up* nominal DANA.

Pada halaman *top up* nominal pengguna dapat menggunakan dua metode untuk melakukan transaksi. Metode pertama yaitu dengan mengklik salah satu daftar favorit yang akan langsung mengarahkan ke halaman *top up* nominal. Metode lain yaitu dengan mengklik *button* "Buat Input Baru" terlebih dahulu lalu lanjut ke halaman *top up* nominal. Pebedaan antara kedua metode tersebut berada di *checkbox* untuk meng*set* sebagai *user* favorit. Pada halaman *top up* nominal dengan *user* favorit tidak terdapat *checkbox* tersebut. Sebaliknya jika pengguna melakukan transaksi dengan *user* baru maka saat mengklik *checkbox* tersebut akan muncul *pop up* berupa *form* untuk mengisi nama *user* favorit. *Form* tersebut memiliki batas pengisian yaitu 15 karakter. Pada bagian atas *form* tersebut akan tertera informasi nomor *handphone* yang dikirim dari halaman sebelumnya yaitu halaman input nomor transaksi.



Gambar 3.55 *Top up* nominal dengan *user* favorit

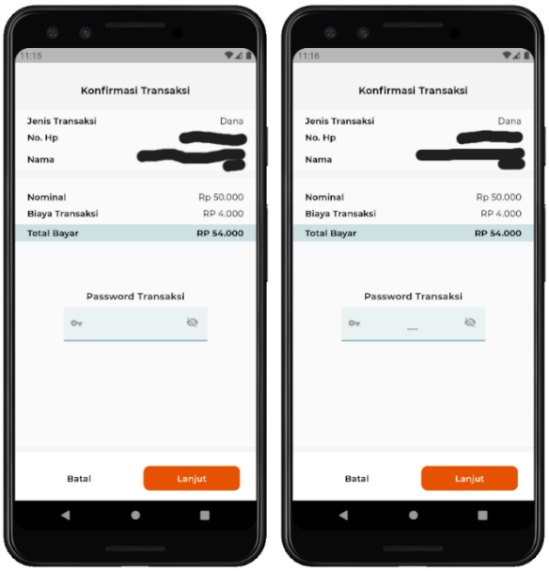
Pada bagian bawah terdapat pilihan *button* nominal yang dapat langsung digunakan seperti Rp 20.000, Rp 50.000, Rp 100.000, Rp 250.000, dan Rp 500.000. Jika pengguna ingin memilih nominal yang tidak terdapat pada pilihan *button* nominal, pengguna dapat mengklik *button* "Nominal Lainnya". Saat *button* "Nominal Lainnya" diklik akan muncul *pop up form* untuk mengisi jumlah nominal yang diinginkan. Pengisian nominal *top up* berpengaruh terhadap *status enable button* "Lanjut". Jika sudah berhasil memilih nominal yang diinginkan *button* "Lanjut" akan aktif dan berwarna *orange*. Sebaliknya jika nominal belum diisi maka *button* "Lanjut" akan berstatus *disable* dan berwarna abu-abu. *Button* tersebut akan mengarahkan ke halaman konfirmasi transaksi. *Top up* nominal dengan *user* favorit dapat dilihat pada Gambar 3.55. *Top up* nominal dengan *set* favorit dapat dilihat pada Gambar 3.56.



Gambar 3.56 *Top up* nominal dengan *set* favorit

d. Halaman konfirmasi transaksi DANA.

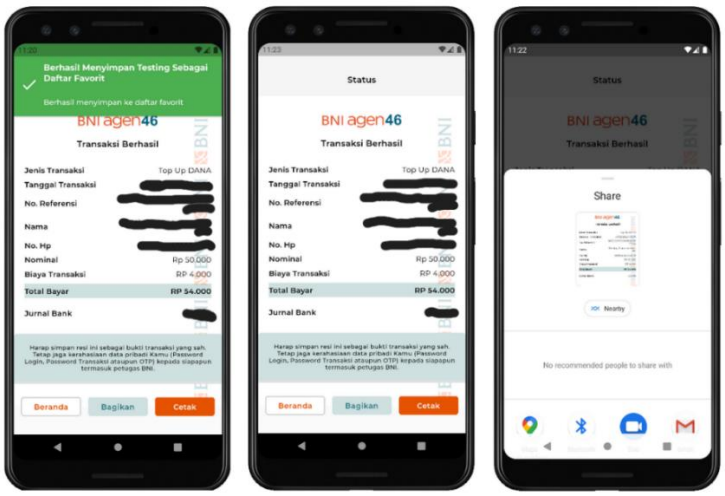
Pada halaman konfirmasi transaksi pengguna akan diperlihatkan informasi mengenai transaksi yang akan dilakukan. Informasi tersebut berisi jenis transaksi, nomor *handphone*, dan nama pemilik nomor *handphone*. Selain itu pada halaman ini juga tertera data biaya transaksi yang akan dilakukan yaitu nominal *top up*, biaya transaksi, dan total bayar. Pada halaman ini pengguna harus mengkonfirmasi *top up* dengan mengisi *password*. Setelah itu pengguna dapat menyelesaikan konfirmasi dengan mengklik *button* "Lanjut". Apabila *password* yang dimasukan tervalidasi benar maka aplikasi akan menampilkan halaman seanjutnya yaitu halaman bukti transaksi.



Gambar 3.57 Halaman konfirmasi transaksi DANA

Sebaliknya jika *password* yang dimasukan salah maka akan muncul *log* peringatan. Pengguna bisa membatalkan konfirmasi dengan mengklik *button* "Cancel". Button tersebut akan mengarahkan pengguna ke halaman sebelumnya yaitu halaman *top up* nominal. Halaman konfirmasi transaksi DANA dapat dilihat pada Gambar 3.57.

e. Halaman bukti transaksi DANA.



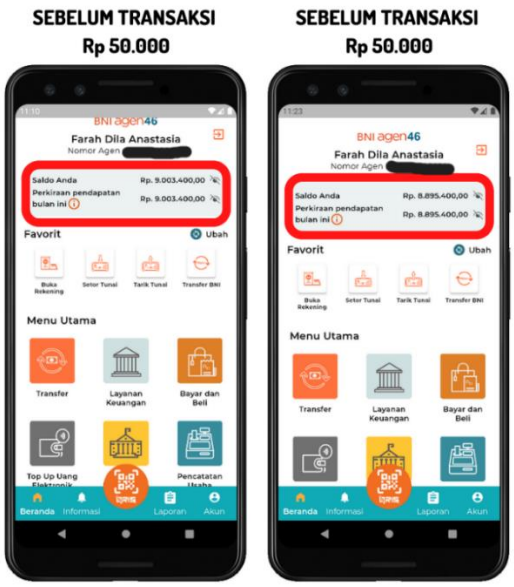
Gambar 3.58 Halaman bukti transaksi DANA

Halaman ini merupakan halaman terakhir saat pengguna berhasil melakukan *top up*. Pada bagian atas halaman ini akan muncul *pop up* berwarna hijau yang memberitahukan bahwa nama yang telah diset sebagai *user* favorit berhasil disimpan.

Pada halaman ini juga tertera informasi detail transaksi yang telah dilakukan yaitu: jenis transaksi, tanggal transaksi, nomor referensi, nama, nomor *handphone*, nominal, biaya transaksi, total bayar, jurnal bank. Selain itu pada bagian bawah terdapat tiga *button* sebagai navigasi. *Button* "Cetak" digunakan untuk mencetak bukti transaksi. *Button* "Cetak" dapat digunakan dengan menghubungkan dengan *device* lain. *Button* "Bagikan" digunakan untuk membagikan bukti transaksi dengan berbagai metode seperti *email*, *bluetooth*, dan lainnya. *Button* "Beranda" digunakan untuk kembali menuju halaman *home*. Halaman bukti transaksi DANA dapat dilihat pada Gambar 3.58.

f. Pengecekan pengurangan saldo DANA.

Pada halaman *home* pengguna dapat mengecek saldo yang tersisa. Pengecekan dilakukan untuk memastikan bahwa proses integrasi API yang terjadi pada fitur tersebut berhasil. Terjadi perbedaan antara sebelum dan sesudah melakukan transaksi top up. Bagian informasi saldo tertera bahwa terdapat pengurangan saldo setelah melakukan transaksi sebesar Rp 54.000. Saldo awal sebesar Rp 9.003.400 berkurang menjadi Rp 8.895.400. Pengecekan pengurangan saldo DANA dapat dilihat pada Gambar 3.59.

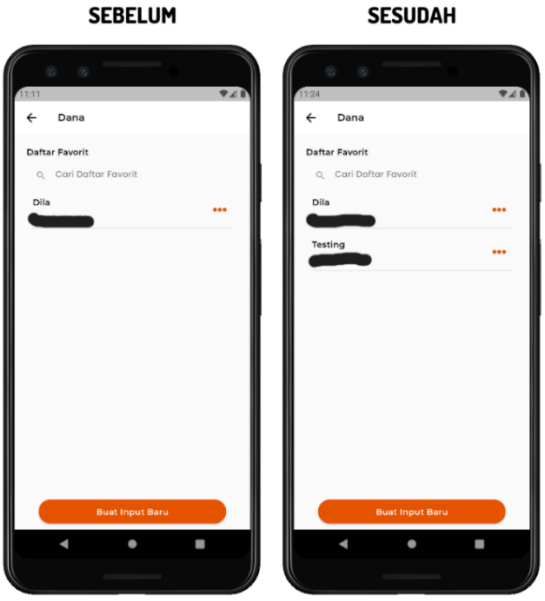


Gambar 3.59 Pengecekan pengurangan saldo DANA

g. Pengecekan daftar *user* favorit DANA.

Pengecekan daftar favorit dilakukan pada halamn utama DANA. Pengecekan dilakuakan untuk memastikan apakah penambahan *user* baru sebagai *user* favorit telah berhasil.

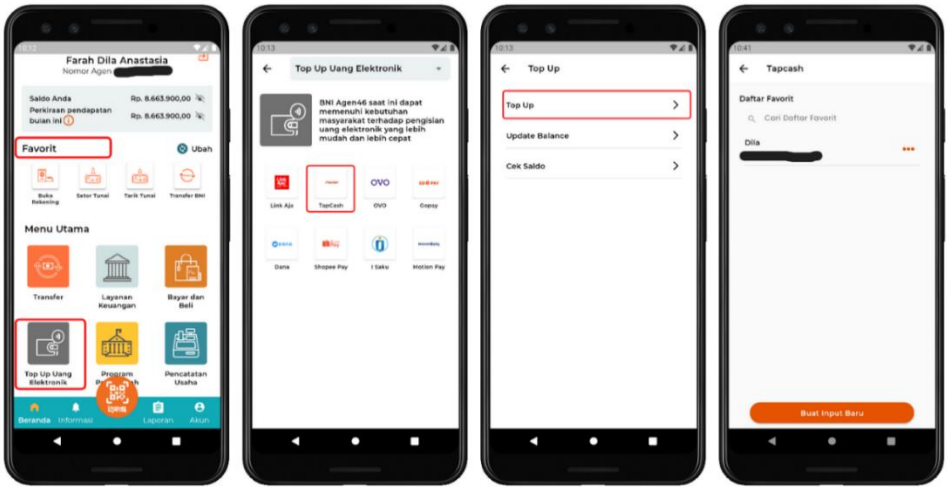
Pada gambar membuktikan bahwa terjadi penambahan daftar *user* favorit pada halaman utama DANA setelah melakukan transaksi *top up*. Hal tersebut membuktikan bahwa penambahan *user* favorit berhasil dilakukan. Pengecekan daftar *user* favorit DANA dapat dilihat pada Gambar 3.60.



Gambar 3.60 Pengecekan daftar *user* favorit DANA

Fitur *top up* TapCash

a. Halaman utama *top up* TapCash.

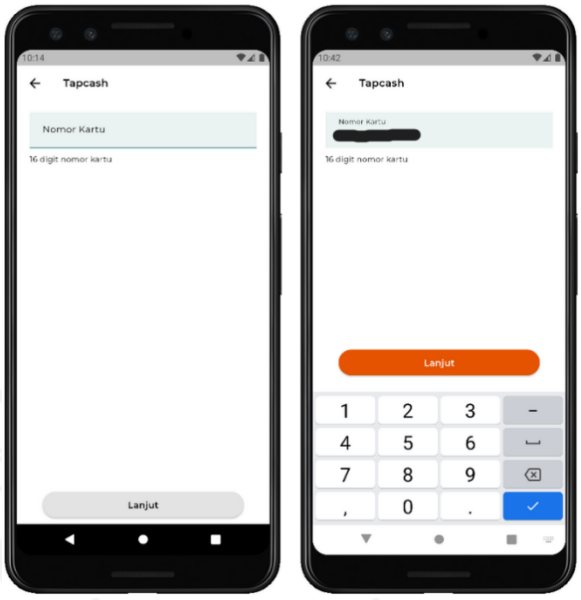


Gambar 3.61 Halaman utama *top up* TapCash

Pengguna harus mengklik menu "*Top Up* Uang Elektronik". Menu tersebut berada pada menu utama atau yang ada pada daftar menu favorit pada bagian atas.

Ketika pengguna mengklik menu "*Top Up* Uang Elektronik" maka akan diarahkan ke dalam halaman *top up* uang elektronik uang elektronik yang berisi pilihan menu metode *top up*. Kemudian pengguna bisa mengklik logo bertuliskan metode "TapCash". Menu "TaCash" akan mengarahkan ke halaman utama *Top Up*. Pada halaman *Top Up* terdapat tiga pilihan menu lagi yaitu *Top Up, Update Balance*, dan Cek Saldo. Pengguna dapat mengklik menu "Top Up" untuk melakukan transaksi *top up* TapCash secara manual. Setelah mengklik menu tersbut pengguna akan diarahkan ke halaman utama TapCash. Pada halaman utama TapCash terdapat *list* daftar favorit *user* dan nomor transaksi yang telah diset sebagai *user* favorit. Pengguna bisa menggunakan salah satu *list* user favorit untuk melakukan transaksi dan menggunakan *button* "Buat Input Baru" untuk melakukan transaksi dengan nomor yang belum pernah digunakan sebelumnya. Halaman utama *top up* TapCash dapat dilihat pada Gambar 3.61.

b. Halaman *input* nomor kartu TapCash.



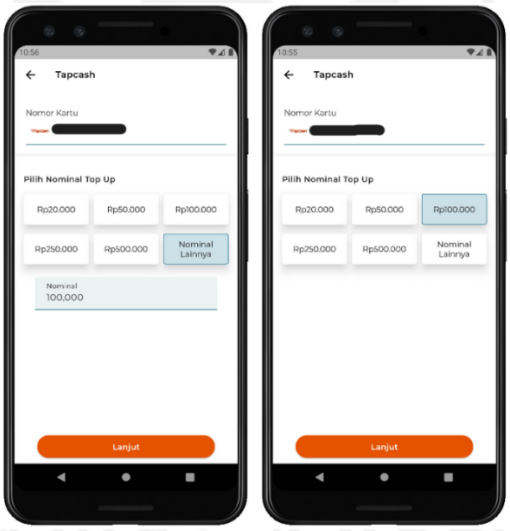
Gambar 3.62 Halaman *input* nomor kartu TapCash

Pada halaman ini pengguna bisa mengisi nomor kartu baru yang belum pernah terdaftar pada daftar *user* favorit. Pengguna dapat mengklik *form* pada bagian atas untuk mengisi nomor kartu yang dituju. Saat pengguna mengklik area *form* tersebut, halaman akan langsung secara otomatis menampilkan *keyboard* bertipe *number*. *Button* "Lanjut" pada bagian bawah akan bergantung pada pengisian *form*. *Form* nomor kartu harus diisi dengan 16 karakter.

Apabila *form* belum terisi maka *button* "Lanjut" akan berstatus *disable* namun jika *form* sudah terisi maka *button* "Lanjut" akan berstatus *enable* dan berwarna *orange*. *Button* "Lanjut" akan mengarahkan ke halaman selanjutnya yaitu halaman *top up* nominal. Halaman *input* nomor kartu TapCash dapat dilihat pada Gambar 3.62.

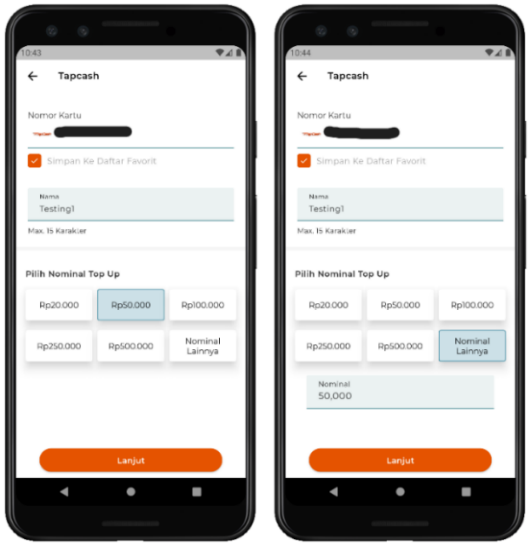
c. Halaman *top up* nominal TapCash.

Pada halaman *top up* nominal memiliki tampilan yang sama seperti halaman *top up* nominal pada fitur DANA dan LinkAja. Pengguna dapat melakukan *top up* dengan dua metode yaitu dengan memilih *user* favorit maupun dengan menginputkan *user* baru. Apabila menggunakan daftar *user* favorit maka pengguna akan langsung diarahkan ke halaman *top up* nominal. Sebaliknya jika pengguna ingin menggunakan *user* baru maka harus mengklik *button* "Buat Input Baru". Pada halaman ini *input user* baru pengguna juga dapat mengset *user* baru sebagai *user* favorit. *Checkbox* untuk meng*set user* favorit akan menampilkan *pop up form* untuk mengisi nama user favorit.



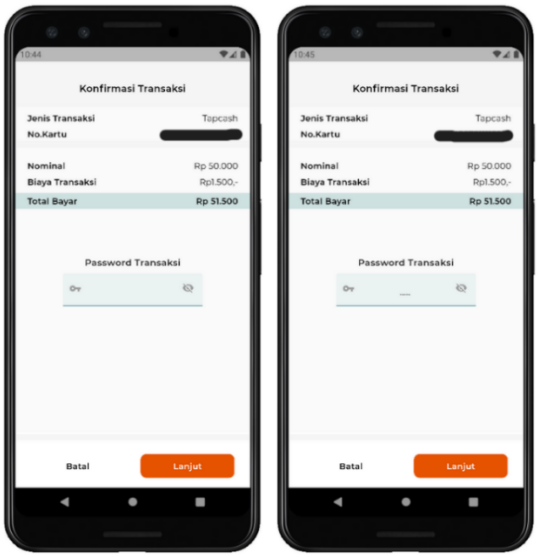
Gambar 3.63 *Top up* nominal dengan *user* favorit

Pilihan *button* pada halaman ini juga memiliki persamaan dengan *button* nominal yang ada pada halaman *top up* nominal DANA. Pilihan *button* terdiri dari *button* nominal angka dan *button* "Nominal Lainnya" untuk mengisi nominal sesuai yang diinginkan. Apabila telah memilih nominal *top up* maka pengguna dapat langsung mengklik *button* "Lanjut". *Button* tersebut akan mengarahkan ke halaman konfirmasi transaksi. *Top up* nominal dengan *user* favorit dapat dilihat pada Gambar 3.63. *Top up* nominal dengan *set* favorit dapat dilihat pada Gambar 3.64.



Gambar 3.64 *Top up* nominal dengan *set* favorit

d. Halaman konfirmasi transaksi TapCash.



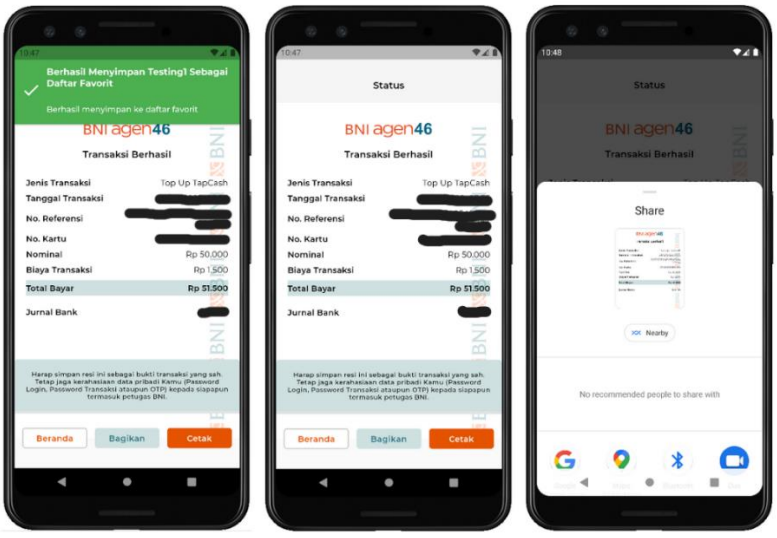
Gambar 3.65 Halaman konfirmasi transaksi TapCash

Setelah melakukan *top up* penguna dapat mengkonfirmasi transaksi yang dilakukan. Halaman konfirmasi transaksi memiliki tampilan yang sama dengan fitur *top up* uang elektronik DANA dan LinkAja. Pada halaman ini akan disajikan informasi data transaksi dan data *payment*. Data transaksi terdiri dari jenis transaksi dan nomor kartu. Data payment terdiri dari nominal *top up*, biaya transaksi dan total bayar. Pada bagian bawah terdapat *form* yang digunakan untuk mengisi *password*. Setelah mengisi *password* pengguna dapat menyelesaikannya dengan mengklik *button* “Lanjut”.

Button “Cancel” digunakan untuk membatalkan transaksi yang dilakukan. *Button* tersebut akan mengembalikan pengguna menuju halaman *top up* nominal. Apabila *password* yang diisikan tervalidasi benar maka transaksi akan berhasil dan menampilkan bukti transaksi pada halaman selanjutnya. Sebaliknya apabila *password* yang dimasukan salah maka akan muncul *log* peringatan. Halaman konfirmasi transaksi TapCash dapat dilihat pada Gambar 3.65.

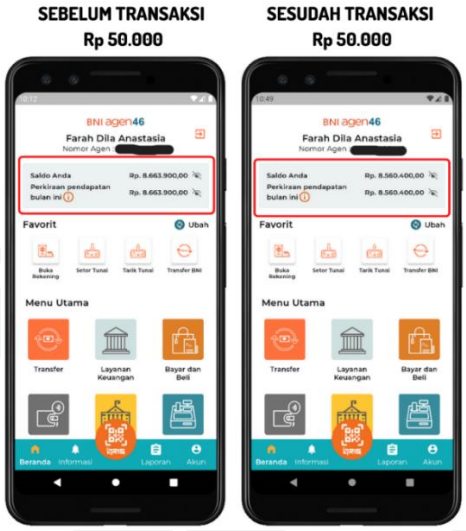
e. Halaman bukti transaksi TapCash.

Halaman ini memiliki persamaan dengan halaman bukti transaksi pada fitur *top up* uang elektronik DANA dan LinkAja. Pada bagian atas akan muncul *log* status penyimpanan nama *user* baru sebagai *user* favorit berwarna hijau. Pada halaman ini akan menampilkan detail transaksi yang telah dilakukan yaitu: jenis transaksi, tanggal transaksi, nomor referensi, nomor kartu, nominal, biaya transaksi, total bayar, dan jurnal bank. Pada bagian bawah halaman ini juga terdapat tiga button yaitu *button* “Cetak”, “Bagikan”, dan “Beranda”. Button “Cetak” digunakan untuk mencetak bukti transaksi. Pengguna dapat mencetaknya dengan cara menghubungkan aplikasi tersebut dengan *device* lain. *Button* “Bagikan” digunakan untuk membagikan bukti transaksi dengan berbagai metode seperti *email*, *bluetooth*, dan lainnya. Terakhir yaitu *button* “Beranda” berfungsi sebagai navigasi untuk kembali menuju halaman *home*. Halaman bukti transaksi TapCash dapat dilihat pada Gambar 3.66.



Gambar 3.66 Halaman bukti transaksi TapCash

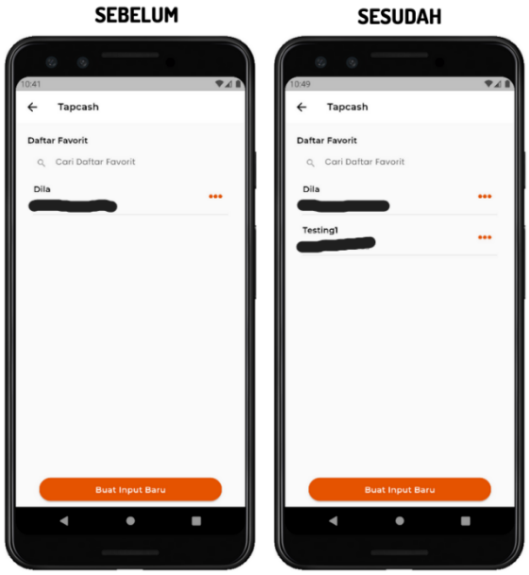
f. Pengecekan pengurangan saldo TapCash.



Gambar 3.67 Pengecekan pengurangan saldo TapCash

Pada halaman *home* pengguna dapat mengecek informasi saldo yang tersisa. Pengurangan saldo membuktikan bahwa proses integrasi API yang terjadi antara *frontend* dengan *backend* berhasil. Pada gamabar membuktikan bahwa terjadi perbedaan antara sebelum dan sesusah melakukan transaksi *top up*. Setelah melakukan transaksi sebesar Rp 50.000 terjadi pengurangan saldo. Saldo awal Rp 8.663.400 berkurang menjadi Rp 8.560.400. Pengecekan pengurangan saldo TapCash dapat dilihat pada Gambar 3.67.

g. Pengecekan daftar *user* favorit TapCash.



Gambar 3.68 Pengecekan daftar *user* favorit TapCash

Setelah berhasil melakukan transaksi pengguna dapat mengecek penambahan *user* favorit pada halaman utama TapCash. Pengecekan dilakukan untuk memastikan bahwa penyimpanan *user* baru sebagai *user* favorit berhasil. Pada gambar membuktikan terjadi perbedaan antara sebelum dan sesudah melakukan transaksi *top up*. Hal ini membuktikan bahwa terjadi penambahan *user* baru setelah melakukan transaksi. Pengecekan daftar *user* favorit TapCash dapat dilihat pada Gambar 3.68.

3.1.5 Pengendaliaan dan Evaluasi Proyek

Pada tahap ini juga dilakukan pengujian terhadap fitur yang telah dibuat serta melakukan *review* hasil kepada *product owner*. Tahap ini juga digunakan untuk mendapatkan saran serta *improvement* yang diperlukan terhadap fitur yang dibuat agar dapat berjalan dengan baik. Pengujian yang dilakukan adalah dengan metode *blackbox testing*. Pengujian dilakukan untuk mengetahui apakah fugsionalitas dalam fitur tersebut sudah berjalan dengan baik. Pengujian *top up* dilakukan menggunakan nomor tujuan transaksi tersendiri yang telah di*setting* oleh bagian *backend engineer*. Evaluasi yang dilakukan terdiri meeting dengan mentor, ketua tim proyek dan *senior engineer*.

Testing

Pada tahap ini dilakukan pengujian dengan *blackbox testing* manual dan evaluasi fitur berupa *review* yang disampaikan kepada *product owner* sebagai ketua tim proyek. *Blackbox testing* dilakukan untuk menguji fungsionalitas setiap bagian dari fitur yang dibuat dan dapat diimplementasikan dengan baik. Pengujian dilakukan dengan menggunakan simulator *device*. Simulator *device* ini hanya bisa dijalankan pada satu device saja baik melalui *handphone* maupun dengan *emulator* di *Android Studio*. *Device* simulator yang digunakan yaitu versi Android 11.0. Apabila ingin pindah *device* maka harus minta izin *request* kepada tim *simulator*. Selama pengujan berlangsung engineer diberikan simulasi akun berupa *username*, *password*, dan data saldo *user* dari tim *backend engineer* untuk mencoba apakah integrasi API yang dibuat sudah berjalan. Pada pengujian ini *frontend engineer* harus memastikan bahwa setiap *widget* yang dibuat harus sesuai dengan desain UI/UX dari tim *designer*. Selain itu pengiriman data dari *backend* menuju *frontend* juga harus dapat tampil pada tampilan *user interface* dari fitur yang dibuat. Hal ini dapat terbukti berhasil jika terdapat pengurangan saldo *user* setelah dan sebelum bertransaksi.

Pada tahap ini *frontend engineer* hanya melakukan pengujian secara fungsionalitas dari data *input* hingga *output* halaman bukti transaksi. Pengujian performa aplikasi dengan Flutter akan diuji lebih lanjut oleh tim QA. Hasil pengujian fungsionalitas fitur dapat dilihat pada Tabel 3.4.

Tabel 3.4 Pengujian fungsionalitas fitur

No	Fitur yang diuji	Fungsionalitas yang diuji	Hasil yang diharapkan	Hasil pengujian	Status
1	*Top up* uang elektronik (LinkAja, DANA, TapCash)	Menu "*Top Up* Uang Elektronik"	Masuk ke halaman *top up* uang elektronik	Masuk ke halaman *top up* uang elektronik	Berhasil
		Menu *top up* LinkAja/DANA /TapCash	Masuk ke halaman *top up*	Masuk ke halaman *top up*	Berhasil
		Halaman daftar favorit	Masuk ke halaman daftar favorit	Masuk ke halaman daftar favorit	Berhasil
			Menampilkan daftar *user* favorit	Menampilkan daftar *user* favorit	Berhasil
			Mencari salah satu daftar favorit	Mencari salah satu daftar favorit	Berhasil
			Memilih salah satu data *user* favorit untuk *top up*	Memilih salah satu data *user* favorit untuk *top up*	Berhasil
			Membuat data *user* baru dengan mengklik *button* "Buat Input Baru"	Membuat data *user* baru dengan mengklik *button* "Buat Input Baru"	Berhasil
			Mengarahkan *button* "Buat Input Baru" ke halaman input *user* baru	Mengarahkan *button* "Buat Input Baru" ke halaman input *user* baru	Berhasil
		Halaman input user baru	Menampilkan halaman *user input* baru	Menampilkan halaman *user input* baru	Berhasil
			Mengisi nomor transaksi yang dituju	Mengisi nomor transaksi yang dituju	Berhasil

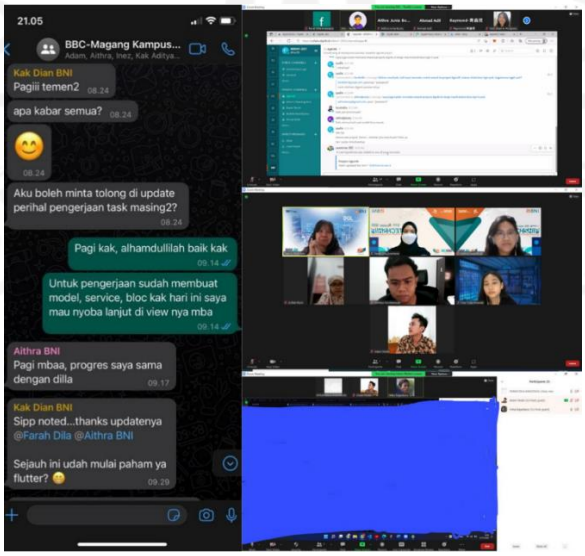
			Mengklik *button* Lanjut untuk menuju halaman *top up* nominal	Mengklik *button* Lanjut untuk menuju halaman *top up* nominal	Berhasil
		Halaman *top up* nominal	Menampilkan halaman *top up* nominal	Menampilkan halaman *top up* nominal	Berhasil
			Menampilkan nomor transaksi yang dituju	Menampilkan nomor transaksi yang dituju	Berhasil
			Meng-set *user* baru sebagai *user* favorit dengan mengklik *checkbox* dan mengisi *form* nama penggunanya	Meng-set *user* baru sebagai *user* favorit dengan mengklik *checkbox* dan mengisi *form* nama penggunanya	Berhasil
			Memilih nominal *top up* dengan pilihan *button*	Memilih nominal *top up* dengan pilihan *button*	Berhasil
			Memilih nominal *top up* dengan mengisi *form*	Memilih nominal *top up* dengan mengisi *form*	Berhasil
			Mengklik *button* “Lanjut” untuk menuju halaman konfirmasi	Mengklik *button* “Lanjut” untuk menuju halaman konfirmasi	Berhasil
		Halaman konfirmasi transaksi	Menampilkan halaman konfirmasi transaksi	Menampilkan halaman konfirmasi transaksi	Berhasil
			Menampilkan data transaksi & data *payment*	Menampilkan data transaksi & data *payment*	Berhasil
			Mengisi *password* transaksi	Mengisi *password* transaksi	Berhasil

			Mengklik *button* "Lanjut" untuk menuju halaman bukti transaksi	Mengklik *button* "Lanjut" untuk menuju halaman bukti transaksi	Berhasil
			Mengklik *button* "Batal" untuk membatalkan konfirmasi transaksi dan 88aragra menuju halaman *top up* nominal	Mengklik *button* "Batal" untuk membatalkan konfirmasi transaksi dan 88aragra menuju halaman *top up* nominal	Berhasil
		Halaman bukti transaksi	Menampilkan halaman bukti transaksi	Menampilkan halaman bukti transaksi	Berhasil
			Menampilkan keseluruhan detail data transaksi	Menampilkan keseluruhan detail data transaksi	Berhasil
			Menampilkan *log* status penyimpanan *user* favorit	Menampilkan *log* status penyimpanan *user* favorit	Berhasil
			Mengklik *button* "Cetak" untuk mencetak bukti transaksi dengan *device* lain	Mengklik *button* "Cetak" untuk mencetak bukti transaksi dengan *device* lain	Berhasil
			Mengklik *button* "Bagikan" untuk membagikan bukti transaksi dalam berbagai format *file*	Mengklik *button* "Bagikan" untuk membagikan bukti transaksi dalam berbagai format *file*	Berhasil
			Mengklik *button* "*Home*" untuk kembali ke halaman utama aplikasi	Mengklik *button* "*Home*" untuk kembali ke halaman utama aplikasi	Berhasil

2	Pembayaran tiket KAI	Menu “Bayar dan Beli”	Masuk ke halaman bayar dan beli	Masuk ke halaman bayar dan beli	Berhasil
		Menu “Tiket Transportasi”	Masuk ke halaman tiket transportasi	Masuk ke halaman tiket transportasi	Berhasil
		Menu “Kereta Api Indonesia (KAI)”	Masuk ke halaman input kode bayar KAI	Masuk ke halaman input kode bayar KAI	Berhasil
		Halaman *input* kode bayar	Menampilkan halaman *input* kode bayar	Menampilkan halaman *input* kode bayar	Berhasil
			Mengisi kode pembayaran	Mengisi kode pembayaran	Berhasil
		Halaman konfirmasi transaksi	Menampilkan halaman konfirmasi transaksi	Menampilkan halaman konfirmasi transaksi	Berhasil
			Menampilkan data transaksi & data *payment*	Menampilkan data transaksi & data *payment*	Berhasil
			Mengisi *password* transaksi	Mengisi *password* transaksi	Berhasil
			Mengklik *button* “Lanjut” untuk menuju halaman bukti transaksi	Mengklik *button* “Lanjut” untuk menuju halaman bukti transaksi	Berhasil
			Mengklik *button* “Batal” untuk membatalkan konfirmasi transaksi dan kembali menuju halaman *input* kode bayar	Mengklik *button* “Batal” untuk membatalkan konfirmasi transaksi dan kembali menuju halaman *input* kode bayar	Berhasil
		Halaman bukti transaksi	Menampilkan halaman bukti transaksi	Menampilkan halaman bukti transaksi	Berhasil

			Menampilkan keseluruhan detail data transaksi	Menampilkan keseluruhan detail data transaksi	Berhasil
			Mengklik *button* "Cetak" untuk mencetak bukti transaksi dengan *device* lain	Mengklik *button* "Cetak" untuk mencetak bukti transaksi dengan *device* lain	Berhasil
			Mengklik *button* "Bagikan" untuk membagikan bukti transaksi dalam berbagai format *file*	Mengklik *button* "Bagikan" untuk membagikan bukti transaksi dalam berbagai format *file*	Berhasil
			Mengklik *button* "*Home*" untuk kembali ke halaman utama aplikasi	Mengklik *button* "*Home*" untuk kembali ke halaman utama aplikasi	Berhasil

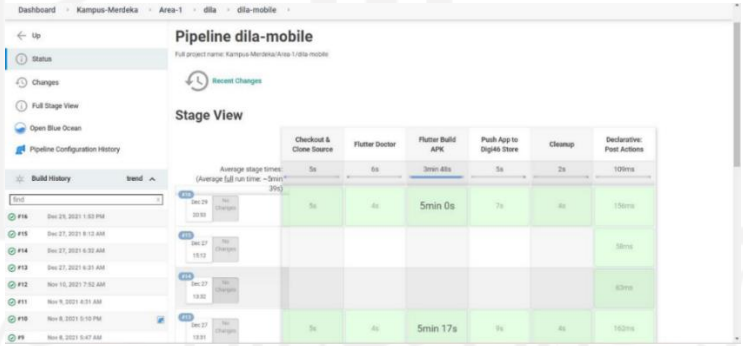
Melakukan *meeting*



Gambar 3.69 Pelaksanaan *meeting*

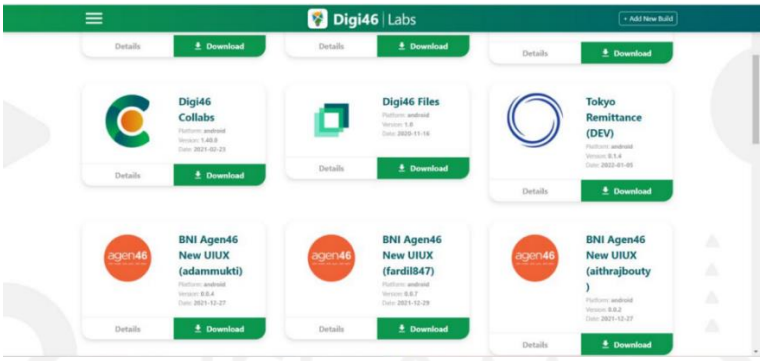
Meeting yang dilakukan terbagi menjadi tiga yaitu dengan mentor, ketua tim proyek, dan *senior engineer*. Ketiga *meeting* tersebut dilakukan secara *online* melalui *platform* Zoom. Meeting yang dilakukan oleh mentor dilakukan secara rutin setiap satu bulan sekali. *Meeting* tersebut membahas mengenai konsultasi kendala teknis dan solusi yang diberikan mentor selama proses pembuatan fitur. Selain itu juga membahas *progress* yang telah dicapai. Saran dan masukan mengenai fitur yang telah dibuat juga disampakan oleh mentor. *Meeting* dengan ketua tim proyek dilakukan secara rutin tiap seminggu sekali. *Meeting* tersebut membahas tentang *progress* pengembangan dan pembagian *backlog* yang diberikan. Penyampaian *progress* pembuatan fitur dilakukan melalui *chatting* dengan *platform* WhatsApp. Tiap tiga hari sekali, ketua tim proyek akan mengecek sejauh mana dalam melakukan pengembangan. Pada *meeting* tersebut juga dilakukan *review* dan saran perbaikan apabila terdapat kekurangan dari fitur yang dibuat. *Meeting* dengan *senior engineer* dilakukan apabila terjadi kendala dalam pembuatan integrasi dengan pihak *backend*. Hal ini dikarenakan segala aktivita magang dilakukan secara *online* sehingga *senior engineer* berperan sebagai penghubung antara *frontend* dengan *backend*. Pelaksanaan *meeting* dapat dilihat pada Gambar 3.69.

Deployment



Gambar 3.70 Jenkins pengembangan fitur

Deployment dilakukan dengan menggunakan *tools automation deployment* yaitu Jenkins. Jenkins menyediakan layanan integrasi *custom* bagi pengembangan perangkat lunak. Penggunaan Jenkins dapat mempercepat *Software Development Lifecycle* (SDLC) bagi perusahaan. Pengembang diberi akses untuk melihat *tracking progress deployment* yang telah dilakukan sehingga dapat mengecek langsung apabila terjadi terjadi kegagalan *build* di *stage* tertentu. Tampilan Jenkins pengembangan fitur dapat dilihat pada Gambar 3.70.



Gambar 3.71 *Deployment* pengembangan fitur

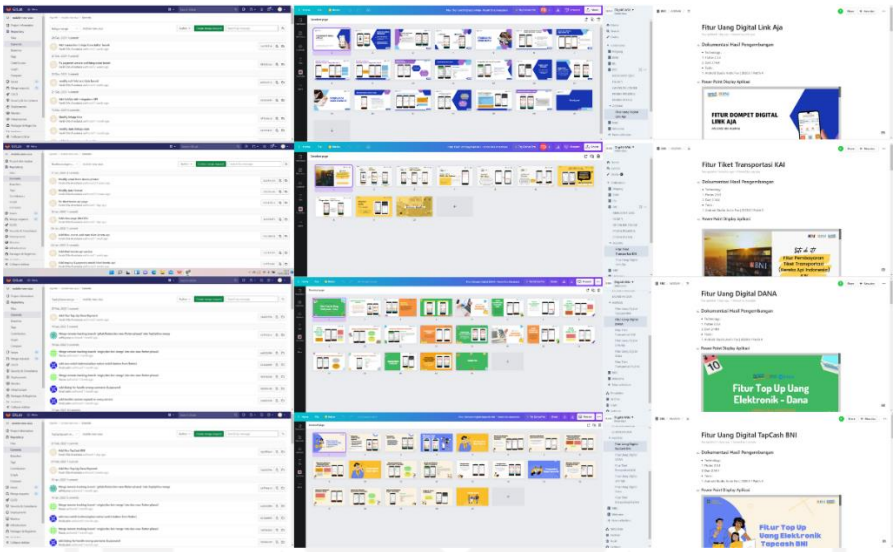
Jenkins *pipeline* sudah dibuat sebelumnya sehingga pengembang cukup menambahkan *tag* pada *branch git* yang dikembangkan kemudian akan otomatis ter *deploy* pada Jenkins. Hasil *deployment* dari aplikasi yang telah di*build* akan langsung terbentuk ke dalam *tools* bernama DigiLabs. *Tools* ini merupakan buatan milik internal BNI yang berisi kumpulan aplikasi-aplikasi yang telah dibuat sehingga memudahkan *product owner* atau *scrum master* jika ingin meng*install* dan mencoba langsung fitur aplikasi yang telah dikembangkan. Tampilan hasil *deployment* pengembangan fitur aplikasi dapat dilihat pada Gambar 3.71.

3.1.6 Penutupan Proyek

Penutupan proses pengembangan *frontend* aplikasi *m-banking* Agen46 masih belum dilakukan. Dengan masa magang yang telah selesai dan masih ada beberapa kekurangan dari segi tampilan tidak terlalu berdampak besar pada alur keseluruhan proses pengembangan. Proyek pengembangan *frontend m-banking* Agen46 pada fitur yang lainnya akan dilanjutkan oleh karyawan BBC yang berkerja di PT. Bank Negara Indonesia Tbk. Pada tahap penutupan juga berkewajiban untuk membuat dokumentasi seluruh fitur yang telah dibuat.

Documentation

Pembuatan dokumentasi wajib dilakukan setelah selesai membuat fitur. Dokumentasi terbagi menjadi tiga yaitu *push code* dengan *Gitlab*, pembuatan *powerpoint* dan dokumentasi pengembangan pada *tools* Digi46 Wiki. Kegiatan dokumentasi *code* kedalam *Gitlab* dilakukan secara rutin jika melakukan perubahan pada file proyeknya. Status aktivitas *push* ke dalam Gitlab secara otomatis terlacak oleh *tools* Digi46 *Collabs*.



Gambar 3.72 Pembuatan dokumentasi

Setiap fitur memiliki *branch git* masing-masing. Selain itu setiap pembuatan fitur baru juga akan diberi file *git* proyek yang baru. *Powerpoint* tersebut berisi alur kerja dari fitur yang dibuat. Pada *powerpoint* tersebut juga menampilkan hasil *output* penggunaan fitur. Dokumentasi penegembangan dibuat pada *tools* Digi46 Wiki. *Tools* ini merupakan aplikasi milik PT. Bank Negara Indonesia Tbk yang digunakan sebagai media dokumentasi peserta magang. Dalam dokumentasi tersebut berisi penjelasan mengenai teknologi yang digunakan, lampiran *powerpoint*, *Gitlab* file proyek, dokumentasi speksifikasi API, lampiran desain UI/UX, dan penjelasan alur integrasi API. Pembuatan dokumentasi berguna sebagai panduan pembelajaran peserta magang lain yang nantinya melanjutkan proyek tersebut. Pembuatan dokumentasi pengembangan dapat dilihat pada Gambar 3.72.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Selama enam bulan mengikuti magang terdapat beberapa teori yang berkaitan dengan implementasi yang terjadi di lapangan. Teori tersebut digunakan pada saat pengerjaan proyek. Adapun beberapa aspek teori yang sama dengan implementasi diantaranya:

4.1.1 Penggunaan *BLoC Pattern*

Implementasi *BLoC Pattern* pada proyek *m-banking* Agen46 terletak pada pembuatan integrasi API. Hal tersebut dilihat berdasarkan pemisahan antara folder *business logic* dengan *user interface* pada struktur file proyeknya. *BLoC Pattern* membantu untuk mengubah setiap *event* yang dikirim oleh *widget* dan mengubah data *response* yang didapat dalam sebuah *state*. Setiap *response* dan *request* pada fitur yang dibuat, baik pada fitur *top up* maupun pembayaran tiket dikirim sebanyak dua kali. Pengiriman pertama menuju *inqury* dan pengiriman kedua menuju *payment*. Untuk setiap format data yang dikirim sudah disesuaikan dengan spesifikasi API. Folder *bloc* pada pembuatan fitur aplikasi m-banking Agen46 terpisah menjadi tiga bagian yaitu: *event, state,* dan *bloc*. Hal ini membuat implementasi yang dibuat sesuai dengan teori.

Segala kondisi yang ada untuk setiap *event request* yang dikirim oleh *widget* dirubah menjadi sebuah *state*. *State* yang dikirim dalam bentuk *response* terbagi menjadi beberapa kondisi seperti *failed, success,* maupun *error*. Selain itu dalam *bloc* terdapat juga *state initial, loading* dan *disposeloading* yang memberikan informasi mengenai proses kondisi saat pengiriman *response*. Segala data *response* yang dikirim oleh *server* kepada *bloc* juga dirubah dalam sebuah *class* yang dinamakan dengan model. Setiap model yang terdapat pada fitur yang dibuat juga diparsing kedalam JSON dengan bantuan *MapString*. *Bloc* membantu sebagai penghubung antara folder *view* yang bertugas sebagai *user interface* dengan folder *service* yang bertugas sebagai *server* nya.

Pembuatan *route* setiap *event* pada fitur top up dan pembayaran tiket KAI sudah dibagi berdasarkan *event* ketika meng*hit* API. Aplikasi *m-banking* pada dasarnya memiliki struktur file yang sangat kompleks. Hal ini dikarenakan *m-banking* memiliki fitur yang sangat banyak. Selain itu untuk setiap fitur yang ada pada aplikasi tersebut tentunya memiliki fungsionalitas yang sangat kompleks.

Pengembangan aplikasi dengan desain *user interface* yang baru tentu membutuhkan sebuah *desain pattern* yang membantu *developer* untuk membagi sebuah *project* kompleks menjadi komponen-komponen kecil.

4.1.2 Penggunaan *widget Flutter*

Flutter menyediakan *framework* UI yang membantu membuat *user interface*. Pembuatannya dilakukan dengan menggunakan bahasa pemrograman Dart. Setiap komponen UI pada *Flutter* disebut dengan *widget*. *Widget* pada *Flutter* memiliki banyak ragam. *Widget* tersebut dapat dikustomisasi sesuai dengan kebutuhan. Setiap komponen dalam *Flutter* disimpan dalam sebuah *class*. *Widget* juga memiliki sifat *reusable widget* atau penggunaan ulang *widget*. Sifat inilah yang membantu *developer* ketika ingin memakai *widget* yang sudah ada. Hal tersebut juga membuat tampilan aplikasi menjadi lebih konsisten. Selain itu penggunaan *widget* kembali membuat lebih *clean code*. Dalam beberapa hal seperti *style* menggunakan *reusable widget* yang diberi nama *BniTheme*. Hal ini bertujuan agar segala *style* yang ada pada tampilan aplikasi memiliki keseragaman dalam *style* nya. Developer tidak perlu lagi mengatur format *style* satu persatu atau bahkan harus membuat *widget* ulang. Penggunaan *reusable widget* dilakukan dengan memanggil *class widget* dan menambahkan modifikasi sesuai dengan kebutuhan sehingga setiap perubahan dapat diimplementasikan di banyak tempat. Ada juga beberapa *widget* lain seperti *alert dialog*, *button animation*, *textfield decoration*, dan *success dialog* yang juga menggunakan prinsip *reusable widget*. Selain itu penggunaan *class* pada pengembangan fitur dominan menggunakan *stateful widget*. Hal ini dikarenakan setiap kondisi dari *state widget* memiliki sifat yang berubah-ubah atau dinamis sesuai dengan karakteristik *interface* nya.

Dalam pengembangan aplikasi *m-banking* Agen46 terbagi menjadi empat fitur yang terdiri dari tiga *merchant top up* dan satu pembayaran tiket kereta api. Dari keempat fitur tersebut memiliki beberapa kemiripan dan tampilan yang sama pada halaman favorit, halaman *input*, halaman *top up*, halaman konfirmasi dan halaman bukti transaksi. Sehingga dapat mengimplementasikan *widget* yang sama dalam beberapa fitur hal inilah yang memudahkan dalam proses pengembangan. Selain itu dengan kelebihan *widget* yang beragam membuat *user interface* yang dibuat menjadi lebih menarik dan smooth. Hal ini dapat dilihat saat menggunakan beberapa *widget animation* pada halaman fitur yang dibuat. Penambahan-penambahan berbagai *property* pada *class widget* tersebut juga membantu ketika ingin membuat sebuah *widget* berdasarkan kebutuhannya.

Widget dalam *Flutter* dapat dijadikan referensi ketika seorang *developer* ingin membangun sebuah *user interface* yang lebih menarik dengan berbagai *property* lain.

4.1.3 Format REST*ful* API

API (*Application Programming Interface*) mengimplementasikan konsep REST (*Representattional State Transfer*). API berfungsi untuk menghubungkan antara satu aplikasi dengan aplikasi lainnya yang memiliki bahasa pemrograman yang beda. Pada API terdiri beberapa komponen seperti desain URL, HTTP *verbs*, HTTP *status code*, *format response* dan lainnya. Pada pengembagan fitur aplikasi *m-banking* Agen46 *developer* menggunakan RESTful sebagai *protocol* pengiriman data antar beberapa aplikasi. Penggunaan komponen API terdiri dari beberapa seperti penggunaan format JSON pada *response* yang didapat atau modelnya, HTTP *status code* untuk setiap *response*, HTTP *verbs*, dan URL untuk setiap *event inquiry* maupun *payment.* Dokumen spesifikasi API ini telah dibuat oleh *system analyst* sebelumnya. Sebelum melakukan pembuatan fitur *developer* diberikan dokumen spesifikasi API yang sudah ditentukan formatnya. Dalam dokumen tersebut dijelaskan secara rinci mengenai format-format data yang akan digunakan ketika membuat *service*. Komponen dalam dokumen spesifikasi API terdiri dari beberapa komponen seperti HTTP *method, path*, HTTP *header, format data, request definition, url, request, response definition, success response code, response success, error response code, response error*. Komponen tersebut memiliki perbedaan antara bagian *inquiry* dengan bagian *payment*. Segala format data tesebut dijelaskan secara rinci dan lengkap beserta dengan contoh data-datanya. Selain itu dokumen spesifikasi API yang diberikan dijadikan panduan bagi pengembang untuk membuat integrasi API pada fitur. Hal ini bertujuan agar data yang ada pada bagian *backend* dapat terhubung dengan tampil pada *frontend* fitur yang telah dibuat.

Selama tahap pembuatan fitur *developer* harus menyesuaikan setiap data *response* maupun *request* antara bagian *inqury* dengan bagian *payment*. Dalam hal ini penggunaan penggunaan REST*ful* API sudah sesuai dengan implementasi yang ada namun ada beberapa hal yang masih terkendala yaitu mengenai konsistensi penulisan *request definition field* dan tipe data yang ada pada dokumen spesifikasi API. Penulisan tipe data dan *field* antara yang dibuat oleh tim *frontend* terkadang berbeda dengan yang dibuat oleh bagian *backend*. Hal ini membuat data yang di*hit* ke API saat integrasi tidak muncul sehingga tim *frontend* harus mengkonfirmasi lagi kepada pihak *backend*.

Penyebab lainnya yaitu tim *frontend* membuat tipe data berdasarkan penulisan yang dibuat oleh tim *system analyst* sedangkan tim *backend* melakukan perubahan penulisan pada dokumen spesifikasi API. Perubahan tersebut tidak di*update* terlebih dahulu pada dokumen sebelum diberikan kepada tim *frontend*. Penggunaan dokumen spesifikasi API sangat penting bagi pembuatan integrasi API sehingga konsep ini dapat dijadikan landasan bagi *developer* ketika ingin membangun sebuah aplikasi yang kompleks dan membutuhkan API. Hal ini bermanfaat agar segala kebutuhannya tercatat dengan jelas dan sinkron antara *frontend* dengan *backend*. Selain itu perlunya konsistensi penulisan tipe data pada dokumen maupun pada *coding* aplikasi agar tidak terjadi kekeliruan yang data menghambat pengembangan sistem.

4.2 Pembelajaran Magang

Banyak sekali manfaat, tantangan dan hambatan selama magang 6 bulan di PT. Bank Negara Indonesia Tbk. Hal- hal yang dialami oleh penulis selama proses kegiatan magang terdiri dari sisi teknis maupun non teknis. Dengan hal tersebut berharap ilmu, pengalaman, dan segala sesuatu yang pernah dialami selama magang dapat dijadikan pembelajaran serta bekal untuk terus belajar dan memperbaiki diri menuju masa depan yang lebih baik lagi kedepannya. Selain itu nilai-nilai yang pernah diajarkan dapat memberi manfaat bagi masyarakat dan orang lain. Selama proses pembelajaran adapun beberapa aspek yang didapatkan yaitu:

4.2.1 Manfaat

a. Teknis

1. Desain *Pattern BLoC*

   Desain pattern membantu seorang *developer* dalam memecahkan masalah yang terjadi ketika mengembangkan aplikasi. *BLoC Pattern* sangat membantu *developer* dalam melakukan pengembangan fitur dengan *user interface* yang baru. Pembagian *file* menjadi berberapa komponen membuat setiap pengujian yang dilakukan menjadi lebih mudah. Selain itu terdapat beberapa penelitian terdahulu yang sudah melakukan riset terkait performa dari penggunaan *BLoC Pattern*. Beberapa penelitian tersebut membuktikan bahwa penggunaan desain *pattern* ini memiliki performa yang baik dalam pengembangan aplikasi berbasis *mobile*. *BLoC Pattern* dapat dijadikan sebuah referensi bagi *developer android* dalam mengembangkan sebuah aplikasi degan menggunakan *Flutter* yang memiliki tingkat kompleksifitas yang tinggi.

2. Penggunaan *Flutter*

Penggunaan *framework Flutter* dapat memudahkan dalam mengembangkan aplikasi berbasis *multiplatform*. Pengembangan aplikasi *multiplatform* dapat dilakukan dengan satu basis *code* saja. Hal ini dikarenakan bahasa Dart memiliki teknologi *render* yang canggih seperti AOT- *Compile* dan LLVM. Selain itu performa *framework Flutter* tidak kalah dari aplikasi *native*. Selain itu dapat membuat *user interface* yang menarik dengan kustomisasi pada *widget Flutter*. Setiap perubahan ketika membuat aplikasi juga dipermudah dengan adanya fitur *hot-reload*. Selain itu dengan kemudahan yang dimiliki oleh *Flutter* membuat proses pengembangan menjadi lebih cepat dan meminimalisir biaya yang harus dikeluarkan.

3. *Real case* proyek

Selama proses magang berlangsung banyak mendapatkan pengetahuan teknologi baru. Pelaksanaan magang dapat terjun langsung dalam pengerjaan *real case* proyek yang ada di BNI. Selain itu dapat belajar sekaligus memecahkan masalah teknologi yang ada pada industri perbankan. Pengerjaan proyek pengembangan aplikasi merupakan kegiatan yang dilakukan setiap semester pada kegiatan perkuliahan. Pada perkuliahan diajarkan pembuatan berbagai jenis aplikasi seperti *web, mobile*, dan *desktop*. Aplikasi tersebut dibuat berdasarkan studi kasus yang terdapat di masyarakat sebagai sebuah solusi perkembangan teknologi. Sama halnya dengan pembuatan proyek ini, yang kaitannya dengan kemudahan masyarakat dalam melakukan transaksi dengan menggunakan *m-banking*. Pengerjaan proyek yang berkaitan dengan industri *mobile banking* atau sistem *payment* ini menjadi pengalaman pertama dan yang sangat berharga.

b. Non Teknis

1. *Mentorship by expert*

Dengan mengikuti magang selama 6 bulan mendapat *mentorship* langsung oleh ahli IT di bidangnya. Dengan kegiatan magang dapat mengenal beberapa *stack* teknologi yang sebelumnya belum pernah didapatkan selama masa perkuliahan. *Mentorship* saat proses *bootcamp* mengenalkan beberapa teknologi yang menjadi *trend* saat ini seperti *React JS*, *Docker*, *Java Springboot*, *Flutter* dan lainnya.

Mentor saat proses magang juga memberikan pengetahuan terkait teknologi-teknologi di industri perbankan seperti Digi46 *Collabs, Jenkins,* Digi46 *Labs*, dan lainnya yang membantu selama proses pengembangan fitur. Selain itu mendapat pengarahan langsung oleh ketua tim proyek dan *senior engineer* tentang pengembangan *frontend* dengan *Flutter* secara mendalam. Hal ini menjadi suatu bentuk pembelajaran serta bekal untuk kedepannya.

2. *Innovative* dan *Problem Solving*

   Selama proses magang ditugaskan untuk mengembangkan *frontend* dengan desain UI/UX yang telah disiapkan sebelumnya. Selain itu peserta dilatih untuk dapat berinovasi menciptakan *widget* sesuai dengan tampilan desain *user interface*. Kreatifitas dalam menciptakan sebuah *widget* diimplementasikan selama pembuatan *frontend* aplikasi. Selain itu terdapat juga sering penemuan *bug* ataupun *error* ketika membuat integrasi API. *Problem solving* diterapkan untuk memecahkan masalah dan solusi seperti *error* maupun *bug* yang ditemukan saat membuat *coding*. Hal ini bertujuan agar dapat menciptakan fitur sesuai dengan spesifikasi kebutuhannya.

3. *Collaboration*, *Project Management* dan *Diversity Mindset*

   Pengembangan dilakukan dengan menggunakan metodologi *Agile Scrum*. Pengembangan aplikasi ini dilakukan dengan beberapa tahapan dari mulai pendefinisian hingga penutupan proyek. Pada pelaksanaan magang terdapat beberapa peran dengan tugasnya masing-masing seperti *scrum master, product owner*, dan *team member*. Metodologi tersebut sesuai dengan yang diajarkan pada mata kuliah Manajemen Pengembangan Teknologi Informasi (MPTI) di semester 6. Tugas yang diberikan juga dibuat dalam sebuah *backlog* untuk setiap anggota pengembang (*team member*) dengan jangka waktu *sprint planning* yang telah ditentukan. Selama proses magang juga berkerja sama dengan tim yang lain dengan melakukan *meeting* secara rutin dengan mentor dan ketua tim proyek. Walaupun pelaksanaan dilaksanakan secara online, namun sering dilakukan *brainstroming* mengenai solusi yang akan dibangun dengan anggota tim, ketua tim proyek, *senior engineer* maupun mentor. Kolaborasi antar tim *backend* dan *frontend* juga sering dilakukan. Kolaborasi ini dilakukan dalam sebuah struktur perusahaan sehingga pengembang berperan dalam metodologi Agile Scrum.

Hal ini tentunya menjadi sebuah pengalaman untuk belajar cara bekerja dalam sebuah manajemen proyek yang memiliki pola pikir berbeda-beda untuk mencapai tujuan bersama sesuai dengan visi dan misi perusahaan.

4.2.2 Kendala, Hambatan dan Tantangan

a. Teknis

1. Format *type* data kurang sinkron

Selama proses pembuatan fitur juga diberikan dokumentasi spesifikasi API sebagai panduan pembuatan intergrasi API. Dokumen tersebut dibuat oleh tim sistem analis yang nantinya akan digunakan oleh tim *backend* dan *frontend*. Dalam beberapa kasus ditengah proses pembuatan terdapat penemuan format tipe data yang tidak konsisten antara yang terdapat pada dokumen spesifikasi API dengan yang ada pada *backend*. Hal ini menyebabkan pemanggilan data tidak muncul pada tampilan aplikasi ketika akan meng*hit* API dan menyebabkan beberapa *bug*. Selain itu *frontend* harus mengkonfirmasi kembali kepada pihak *backend* terkait penggunaan tipe data yang digunakan. Konfirmasi yang dilakukan sebagai peserta magang tidak bisa menghubungi tim *backend* langsung tetapi harus diwakilkan oleh *senior engineer* yang berada satu tim. Hal ini membuat beberapa kasus harus menunggu waktu yang cukup lama untuk memperbaikinya.

2. Penggunaan simulator *device* terbatas

Selama proses pengembangan diberikan satu hak akses aplikasi dengan *username, password*, dan saldo transaksi yang telah di*setting* oleh *backend*. Penggunaan *device* ini bertujuan agar dapat menguji fungsionalitas fitur beserta dengan penggunaan transaksi secara nyata. Simulator *device* dapat berjalan apabila izin akses nya telah dinyalakan oleh tim yang berwenang. Dalam pelaksanaan magang yang dilaksanakan secara *online* terdapat kendala dalam akses yang bergantung pada jam operasional kantor. Hal ini menyebabkan kendala ketika ingin melakukan perubahan atau perbaikan dalam pembuatan fitur diluar jam kerja. Perubahan dan perbaikan yang dilakukan tidak akan muncul jka izin akses simulator nya tidak berjalan. Selain itu penggunaan simulator *device* hanya dapat digunakan pada satu *device* saja baik perangkat keras *android* maupun *emulator* pada Android Studio.

Apabila *developer* ingin mengganti *device* maka harus meminta perizinan ulang untuk izin *device* yang baru. Dalam beberapa kasus juga ditemui kendala saat file proyek yang didapat tidak bisa dijalankan pada perangkat keras *android* sehingga harus melalui *emulator*. Hal ini menjadikan tantangan untuk berkerja secara optimal pada jam operasional kantor dan memastikan izin akses pada simulator *device* tetap berjalan.

3. Penyimpanan file proyek cukup besar

   Pada pembuatan fitur diberikan satu file *git* proyek Agen46. Dalam file tersebut terdapat berberapa pengembangan fitur yang sudah jadi. Setiap satu fitur yang akan dibuat nantinya disimpan dalam satu *file* proyek. Pada pembuatan fitur selanjutnya akan diberikan *file* git proyek terbaru. Hal in membuat developer harus men*download file* baru setiap saat untuk satu pembuatan fitur. Setiap satu *file* proyek flutter memakan *memory* yang cukup besar sehingga penyimpanan dalam laptop juga membutuhkan *space* yang besar. Penggunaan *code editor* dengan Android Studio juga tentunya akan memakan *memory* penyimpanan sehingga harus menambah SSD pada laptop agar proses pengembangan berjalan dengan lancar.

4. Komunitas *framework* jarang

   Framework *Flutter* resmi rilis pada tahun 2018. Penggunaan framework yang cukup baru menyebabkan kesulitan untuk mencari dokumentasi terkait *case* yang dialami selama proses pembuatan fitur. *Developer* yang menggunakan *Flutter* masih cukup jarang ditemukan. Hal ini menyebabkan kesulitan mencari solusi masalah yang terjadi ketika menemukan *bug* di internet. Selama pembuatan fitur sering dilakukan konsultasi dan koordinasi dengan *senior engineer* apabila terkena *bug* yang belum pernah ditemukan sebelumnya.

5. Data simulasi transaksi terbatas

   Proses pengujian dilakukan dengan menggunakan nomor transaksi yang diberikan oleh tim *backend*. *Developer* dapat menggunakan saldo transaksi untuk melakukan pembayaran sama seperti penggunaan *mobile banking* pada umumnya.

Nomor transaksi yang diberikan oleh *backend* merupakan nomor transaksi yang telah di*default* sehingga ketika ingin mencoba *top up* atau pembayaran hanya dapat menggunakan nomor transaksi tertentu saja yang telah diberi. Pembatasan nomor tersebut bertujuan agar data yang masuk pada bagian *backend* tidak menyebabkan data *cache* yang terlalu banyak.

b. Non Teknis

1. Koordinasi tim

Kegiatan rutin *meeting* kurang dapat terlaksana dikarenakan setiap anggota tim memiliki kegiatan serta kesibukan pada jam yang berbeda. Anggota tim lebih sering melakukan diskusi secara online melalui *chatting* aplikasi Whatsapp. Pelaksanaan magang yang dilaksanakan secara online sehingga harus memperhatikan jam kerja apabila ingin berkonsultasi maupun memberikan *progress* pekerjaan yang telah dilakukan. Hal ini menjadi tantangan agar tidak mengaggu aktivitas jam kerja kantor dari mentor maupun ketua tim proyek. Penyebab lainnya adalah belum maksimalnya penggunaan *tools* manajemen proyek. Pada awal pelaksanaan magang dikenalkan dengan Trello namun ditengah perjalanan proses magang *tools* tersebut terkendala dengan *trial* untuk *work log power-up*s.

2. Manajemen kerja

Pelaksanan magang dengan sistem *work from home* (*wfh*) dituntut untuk mengatur jam kerja nya secara mandiri. Penggunaan *tracking work* dengan Digi46 *Collabs* membuat seluruh *log* aktivitas akan terpantau secara online. Hal ini menjadi tantangan untuk menjaga komunikasi yang baik agar tidak terjadi kesalahpahaman. Selain itu sistem *wfh* yang berlangsung salama 6 bulan memicu kemungkinan munculnya rasa bosan, jenuh dan malas ditengah proses pengembangan.

3. Pelaksaaan *bootcamp*

Pelaksanaan *bootcamp* yang dilakukan selama satu bulan penuh dirasa memakan tenaga yang cukup banyak. Dalam waktu satu bulan dituntut untuk mempelajari beberapa *stack* teknologi dalam waktu yang cukup singkat. Pelaksaan *bootcamp* setiap harinya dilaksanakan secara *meeting* online memiliki waktu istirahat yang kurang. Kegiatan *live coding* juga sering dilakukan selama kegiatan *bootcamp*.

Selesai kegiatan *bootcamp* setiap harinya masih diberikan tugas rutin individu dan tugas kelompok berupa *case study* yang harus dikerjakan oleh peserta magang. Hal ini menjadi sebuah tantangan untuk menyelesaikan *bootcamp* dengan proses yang terjadi setiap harinya. Waktu, tenaga, dan pikiran dilatih untuk menjadi pribadi yang kuat menghadapi segala situasi yang sulit.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah melaksanakan magang selama 6 bulan di perusahaan PT. Bank Negara Indonesia, telah dibuat 4 fitur pada aplikasi *mobile banking* Agen46 dengan teknologi *Flutter* dan desain *pattern BLoC*. Fitur yang dikembangkan yaitu: fitur *top up* uang elektronik LinkAja, DANA, TapCash dan fitur pembayaran tiket KAI. Seluruh tugas yang diberikan ketua tim proyek dan mentor dapat dikerjakan dengan baik sehingga tercapai segala tujuan dan maanfaat dari pembuatan laporan ini. Tujuan dari pembuatan laporan ini dapat dilihat pada Bab 1.3 dan manfaat dari pembuatan laporan ini dapat dilihat pada Bab 1.4. Hasil pengembangan fitur yang telah dilakukan dapat dilihat pada Bab 3.1.4 sehingga mendapatkan hasil pengujian fitur yang memastikan bahwa fungsionalitas fitur berjalan dengan baik dapat dilihat pada Bab 3.1.5. Berdasarkan pengembangan *frontend* pada aplikasi *m-banking* Agen46 dengan teknologi *Flutter* dapat disimpulkan bahwa:

a. Pengembangan fitur *top up e-money* dan pembayarn tiket KAI pada aplikasi *m-banking* Agen46 dengan *BLoC Pattern* memiliki beberapa bagian dalam struktur file, yaitu 1) *Bloc*, 2) Model, 3) *Service*, 4) *Route*, 5) *View* yang dapat dilihat pada Bab 3.1.4.
b. Pengembangan *frontend* fitur *top up* e-money dan pembayaran tiket KAI pada aplikasi *m-banking* Agen46 memiliki beberapa bagian *view*, yaitu 1) Halaman *lis*t daftar favorit, 2) Halaman input nomor transaksi, 3) Halaman *top up* nominal 4) Halaman konfirmasi transaksi, 5) Halaman bukti transaksi yang dapat dilihat pada Bab 3.1.4.
c. Implementasi *BLoC Pattern* pada pengembangan frontend fitur *top up e-money* dan pembayaran tiket KAI dengan *framework Flutter* mempermudah *developer* dalam mengembangkan aplikasi m-banking berbasis *multiplatform* dan pengembangan dilakukan dengan menggunakan metodologi *agile scrum*.
d. Penelitian ini telah menghasilkan pengembangan *frontend* fitur *top up e-money* LinkAja, DANA, TapCash dan pembayaran tiket KAI pada *m-banking* Agen46 sesuai dengan analisis kebutuhan sistem yang diperlukan yang dapat dilihat pada Bab 3.1.4.
e. Berdasarkan hasil pengujian fungsionalitas, fitur *top up e-money* dan pembayaran tiket KAI dapat berjalan dengan baik. Dengan hasil tersebut maka diharapkan dapat membantu BNI dalam mengembangkan aplikasi *m-banking* berbasis *multiplatform*.

5.2 Saran

Terdapat beberapa saran dalam pelaksanaan magang yang akan disampaikan kepada:

a. Perusahaan magang dan pengembangan selanjutnya

Saran untuk perusahaan magang yaitu mengoptimalkan pembuatan *backlog* untuk menunjang pembuatan aplikasi yang lebih baik lagi kedepannya. Pelaksanaan meeting dengan metodologi agile scrum juga perlu peningkatan agar mengurangi kesalahpahaman antar tim. Selain itu perlunya penentuan stack teknologi pada requirement bagi calon peserta magang. Hal ini perlu dilakukan agar ketika peserta magang diberikan proyek cukup menggunakan stack teknologi yang dikerjakan saat pengerjaan proyek. Pada pelaksanaan proyek perlunya konsistensi pembuatan panduan dokumentasi seperti spesifikasi API. Kemudian perlunya pengujian lebih lanjut terkait performa dari penggunaan *BLoC Pattern* untuk mengetahui seberapa banyak pengguna yang dapat menggunakan aplikasi secara bersamaan.

b. Kampus

Perlu adanya pemberitahuan secara detail tanggal penting mengenai pengumpulan laporan, pelaksanaan diseminasi, syarat pengajuan pendadaran dan lainnya. Hal ini agar mahasiswa dapat mempersiapkan dari jauh hari. Selain itu perlunya info-info lowongan magang bagi mahasiswa yang akan mengambil jalur magang sebagai tugas akhinya. Selain itu perlunya penambahan ilmu pengetahuan pada kegiatan perkuliahan mengenai *stack* teknologi yang menjadi *trend* di industri teknologi saat ini.

c. Peserta magang selanjutnya

Bagi mahasiswa yang ingin mengambil jalur magang dan memilih PT. Bank Negara Indonesia sebagai tempat pelaksanaan magangnya untuk dapat mempersiapkan diri. Persiapan bagi peserta dari mulai *skill* yang harus dikuasai, proses seleksi yang akan dilewati, dan persiapan pemberkasan yang harus dikumpulkan. Peserta juga harus aktif mencari-cari informasi lowongan magang yang sesuai dengan posisi serta passion yang dimiliki agar ketika pelaksanaan magang peserta sudah menguasai *stack* teknologi yang digunakan saat pengerjaan proyek. Peserta juga perlu mengupgrade *skill* terkait bahasa pemrograman yang dibutuhkan di industri teknologi saat ini seperti *React Js*, *Golang*, *Flutter*, *Docker* dan lainnya. Selama proses magang berlangsung peserta juga harus komitmen untuk melaksanan tugas yang diberikan dengan baik dan semaksimal mungkin.

DAFTAR PUSTAKA

android. (2021). The Activity Lifecycle. Retrieved June 25, 2022, from Android Developers website: https://developer.android.com/guide/components/activities/activity-lifecycle#oncreate

Apple. (2020). SwiftUI. Retrieved from developer.apple.com website: https://developer.apple.com/xcode/swiftui/

Assyfa, M. Z. (2020). *Implementasi Desain Pattern BLoC dan Pattern Repository pada Aplikasi Energy Management System PT. PLN Berbasis Mobile dengan Flutter* (Thesis; p. 53). Program Studi Teknik Informatika Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta.

Bank Indonesia. (2020, December 1). Apa itu Uang Elektronik. Retrieved May 28, 2022, from www.bi.go.id website: https://www.bi.go.id/id/edukasi/Pages/Apa-itu-Uang-Elektronik.aspx

BNI. (1992a). Beranda | BNI. Retrieved May 30, 2022, from www.bni.co.id website: https://www.bni.co.id/id-id/

BNI. (1992b). BNI Agen46 | BNI. Retrieved June 1, 2022, from www.bni.co.id website: https://www.bni.co.id/id-id/ebanking/bniagen46

Boukhary, S., & Colmenares, E. (2019). A Clean Approach to Flutter Development through the Flutter Clean Architecture Package. https://doi.org/10.1109/CSCI49370.2019.00211

Bulavin, V. (2020). SwiftUI View Lifecycle. Retrieved June 26, 2022, from Yet Another Swift Blog website: https://www.vadimbulavin.com/swiftui-view-lifecycle/

Dev, P. (2022). provider | Flutter Package. Retrieved June 27, 2022, from Dart packages website: https://pub.dev/packages/provider

Flutter. (2018). Flutter Documentation. Retrieved May 29, 2022, from flutter.dev website: https://flutter.dev/docs

Hoang, L. H. (2019). *State Management Analyses of the Flutter Application* (Bachelor's Thesis; pp. 39 pages). Metropolia University of Applied Sciences Bachelor of Engineering. Retrieved from https://www.theseus.fi/bitstream/handle/10024/267674/Ly_Hoang.pdf?sequence=2&isAllowed=y

Hussain, H., Khan, K., Siddiqui, I., Farooqui, F., & Ali, Q. (2021). *Comparative Study of Android Native andFlutter App Development*. KSII The 13th International Conference on Internet (ICONI) 2021.

Ismail Abdelfattah, R., Awad, S., & Nasr, M. E. (2020). Simplified Hybrid Secure Algorithm for Mobile Banking Application. *Journal of Physics: Conference Series*, *1447*(1), 012052. https://doi.org/10.1088/1742-6596/1447/1/012052

Kusumaningrum, A., Sajati, H., & Anarianto, D. (2019). Rest and Soap Comparison on Web Service Technology for Android Based Data Services. *Conference SENATIK STT Adisutjipto Yogyakarta*, *5*. https://doi.org/10.28989/senatik.v5i0.355

Mantik, J., Rama Prayoga, R., Munawar, G., Jumiyani, R., & Syalsabila, A. (2021). Jurnal Mantik is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0). Performance Analysis of BLoC and Provider State Management Library on Flutter. *Jurnal Mantik*, *5*(3), 1591–1597.

Octabriyantiningtyas, D., Suryani, E., & Jatmiko, A. R. (2019). Modeling Customer Satisfaction with the Service Quality of E-Money in Increasing Profit of PT. Telekomunikasi Indonesia. *Procedia Computer Science*, *161*, 943–950. https://doi.org/10.1016/j.procs.2019.11.203

Puspitawati, L., & Gurning, P. (2019). Electronic payment for Micro, Small and Medium Enterprises in Developing Countries. *IOP Conference Series: Materials Science and Engineering*, *662*, 032060. https://doi.org/10.1088/1757-899x/662/3/032060

Ria, D., Nugraha, A., & Firmanto, Y. (2021). *EFEKTIVITAS KEBERHASILAN IMPLEMENTASI SISTEM PEMBAYARAN E-WALLET LINKAJA PADA PEMBELIAN TIKET KA LOKAL DI KAI ACCESS THE EFFECTIVENESS OF THE LINKAJA E-WALLET PAYMENT SYSTEM IMPLEMENTATION ON THE LOCAL TRAIN TICKET PURCHASE ON KAI ACCESS*. Retrieved from https://jimfeb.ub.ac.id/index.php/jimfeb/article/viewFile/6709/5821

Soni, A., & Ranga, V. (2019). API Features Individualizing of Web Services: REST and SOAP. *International Journal of Innovative Technology and Exploring Engineering*, *8*(9S), 664–671. https://doi.org/10.35940/ijitee.i1107.0789s19

Suzanti, I. O., Fitriani, N., Jauhari, A., & Khozaimi, A. (2020). REST API Implementation on Android Based Monitoring Application. *Journal of Physics: Conference Series*, *1569*, 022088. https://doi.org/10.1088/1742-6596/1569/2/022088

Wijayanto, B., Maryanto, E., Rahayu, S. P., & Iskandar, D. (2019). The development of REST API-based android application for Micro, Small and Medium Enterprises (MSME) in Purbalingga Regency. *Journal of Physics: Conference Series, 1367*, 012005. https://doi.org/10.1088/1742-6596/1367/1/012005

LAMPIRAN