MIGRASI PENGEMBANGAN APLIKASI ANDROID (TRAVEL AJA) DARI REACT NATIVE KE KOTLIN NATIVE MENGGUNAKAN METODE SCRUM

Disusun Oleh:

N a m a	: Yafi Hudatama Wibowo
NIM	: 19523224

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

MIGRASI PENGEMBANGAN APLIKASI ANDROID (TRAVEL AJA) DARI REACT NATIVE KE KOTLIN NATIVE MENGGUNAKAN METODE SCRUM

TUGAS AKHIR

Disusun Oleh:

N a m a : Yafi Hudatama Wibowo

NIM : 19523224

Yogyakarta, 7 Juli 2023

Pembimbing,

Galang Prihadi Mahardhika, S.Kom., M.Kom.

HALAMAN PENGESAHAN DOSEN PENGUJI

MIGRASI PENGEMBANGAN APLIKASI ANDROID (TRAVEL AJA) DARI REACT NATIVE KE KOTLIN NATIVE MENGGUNAKAN METODE SCRUM

TUGAS AKHIR

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

Yogyakarta, 10 Oktober 2023

Tim Penguji

Galang Prihadi Mahardhika, S.Kom., M.Kom. .

Anggota 1

Kurniawan Dwi Irianto, S.T., M.S.C.

Anggota 2

Moh. Idris, S.Kom., M.Kom.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Yafi Hudatama Wibowo

NIM : 19523224

Tugas akhir dengan judul:

MIGRASI PENGEMBANGAN APLIKASI ANDROID (TRAVEL AJA) DARI REACT NATIVE KE KOTLIN NATIVE MENGGUNAKAN METODE SCRUM

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, 7 Juli 2023

(Yafi Hudatama Wibowo)

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 Tri Eni Kawuri dan Bapak Wahono 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 adek saya Gatra serta Mas Bayu, Mas Adam, Mas Beny, Mas Wawan yang selalu memberikan bimbingan, nasehat, dorongan, dan semangat yang diberikan kepada saya selaku adik terakhir selama menempuh pendidikan di Universitas Islam Indonesia.

Dosen Pembimbing,

Terima kasih sebanyak-banyaknya untuk dosen pembimbing saya, Bapak Galang Prihardi Mahardhika, S.KOM, M.KOM 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 Guntur, Refo, Rio, Rashid, Rama, Wahyu, Alif dan Vasant 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

“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” (Steve Ballmer)

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 "**MIGRASI PENGEMBANGAN APLIKASI ANDROID (TRAVEL AJA) DARI REACT NATIVE KE KOTLIN NATIVE MENGGUNAKAN METODOLOGI SCRUM**". Serta shalawat dan salamdiucapkan 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. Telkom 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 magangdengan 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 Jurusan Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.. selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Galang Prihardi Mahardhika, S.KOM, M.KOM selaku Dosen Pembimbing yang telah bersedia membimbing dan mengarahkan penyusunan laporan ini.
6. Ibu Pramesti Puji Lestari, Squad Leader Travel & Tourism Direktorat Digital Business (DDB) PT. Telkom Indonesia Tbk selaku pembimbing yang telah menerima penulis dengan baik.
7. Mas Tiara Agung dan tim pengembang di Travel & Touris projek TravelAja mobile apps, Divisi Direktoral Digital Bussines (DDB) PT. Telkom Indonesia

Tbk.

8. Segenap karyawan dan karyawati PT. Telkom Indonesia Tbk yang telah tulus menerima, memberi pengarahan dan membantu penulis selama melaksanakan magang di perusahaan tersebut.

Laporan Tugas Akhir ini telah dibuat dengan usaha terbaik, tetapi masih jauh dari kata sempurna. Penulis memerlukan 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.

Yogyakarta, 7 Juli 2023

(Yafi Hudatama Wibowo)

SARI

PT. Telkom Indonesia sebagai perusahaan telekomunikasi terkemuka di Indonesia berkomitmen untuk terus berinovasi dalam menyediakan solusi teknologi yang bermanfaat bagi masyarakat. Dalam upaya tersebut, PT. Telkom Indonesia telah mengembangkan aplikasi *Travel* Aja sebagai salah satu produk inovatif yang bertujuan untuk memenuhi kebutuhan para *traveler* dalam merencanakan dan mengatur perjalanan mereka. Seiring dengan peningkatan permintaan akan pengembangan aplikasi lintas *platform*. Dulu aplikasi *Travel* Aja dikembangkan oleh developer vendor menggunakan *react native*. Namun, dengan munculnya Kotlin *native*, para pengembang sekarang memiliki alternatif lain. Developer PT.Telkom Indonesia mengevaluasi manfaat dan tantangan migrasi dari *react native* ke Kotlin *native* dengan membandingkan faktor-faktor seperti kinerja, *reusable code*, efisiensi pengembangan, dan pengalaman pengguna secara keseluruhan. Metode pengembangan aplikasi Travel Aja menggunakan pendekatan *agile,* dengan fokus pada penggunaan metode *scrum*. Selama proses pengembangan, pemahaman tentang kebutuhan pengguna dan tantangan dalam merencanakan perjalanan dipelajari secara mendalam. Aplikasi ini menawarkan fitur-fitur seperti berbagi cerita dan pengalaman, memberikan rekomendasi destinasi, serta memfasilitasi interaksi dan koneksi antara *traveler*. Melalui analisis komprehensif dan studi kasus migrasi aplikasi, laporan ini memberikan wawasan tentang keuntungan dan keterbatasan kotlin *native* serta dampaknya terhadap proses pengembangan.

Kata kunci: migrasi, Travel Aja, pengembangan aplikasi, *react native*, kotlin *native*, dan efektivitas.

GLOSARIUM

Compile	proses untuk mengubah berkas kode program dengan berkas lain yang terkait menjadi berkas yang siap untuk dieksekusi oleh sistem operasi secara langsung.
Debug	langkah untuk menelusuri kesalahan kode program. Developer Seseorang yang bertugas mengembangkan suatu sistem.
Retrofit	*Library* untuk mempermudah pertukaran data melalui *REST API*.
Room	*Library* dari Google untuk mempermudah menggunakan SQLite.
Scrum	Kerangka kerja yang digunakan untuk pengembangan suatu produk.
User interface	Tampilan antarmuka pengguna

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pariwisata merupakan sektor yang memiliki potensi besar dalam meningkatkan pertumbuhan ekonomi suatu negara. Di Indonesia, sektor pariwisata telah menjadi salah satu sektor andalan dalam menghasilkan pendapatan dan menciptakan lapangan kerja (Rama, 2020). Namun, salah satu masalah yang dihadapi oleh sektor pariwisata terkait pemanfaatan teknologi adalah kurangnya aksesibilitas informasi wisata yang akurat dan terkini. Banyak destinasi wisata di Indonesia yang masih belum memiliki sistem informasi yang memadai sehingga sulit bagi wisatawan untuk mendapatkan informasi yang diperlukan seperti harga tiket, fasilitas, dan aktivitas yang tersedia di destinasi tersebut (Avinda et al., 2016). Selain itu, kurangnya pemanfaatan teknologi dalam sektor pariwisata juga berdampak pada kurangnya promosi yang efektif. Promosi merupakan salah satu faktor penting dalam menarik minat wisatawan untuk mengunjungi suatu destinasi. Namun, banyak destinasi wisata di Indonesia yang masih mengandalkan promosi konvensional seperti brosur, spanduk, dan media cetak, sedangkan potensi pemanfaatan platform digital dan sosial media untuk memperluas jangkauan promosi belum dimanfaatkan secara optimal (Wahyudi et al., 2022).

PT. Telkom Indonesia berinisiatif untuk memberikan solusi dengan mengembangkan sebuah aplikasi yakni “Travel Aja” dengan tujuan untuk berinovasi di sektor pariwisata. Aplikasi ini ditempatkan di dalam *project* TNT *(Tourisme And Travel)* untuk memenuhi kebutuhan informasi dan pengalaman pengguna dalam menjelajahi tempat-tempat wisata di Indonesia. Dengan menggunakan aplikasi Travel Aja, pengguna dapat dengan mudah menemukan informasi terkini mengenai destinasi wisata, melihat ulasan dan rating dari pengguna lain, merencanakan liburan dengan budget yang dapat disesuaikan oleh pengguna, mengatur jadwal perjalanan, dan banyak lagi.

Aplikasi yang berbasis mobile ini sebelumnya dikembangkan menggunakan teknologi *react native* dengan beberapa pertimbangan seperti, menghemat biaya, menghemat waktu pengembangan aplikasi serta hanya diperlukan satu jenis developer karena *react native* dapat digunakan dibeberapa sistem operasi sekaligus.

Namun, disaat pengembangan aplikasi yang mulai kompleks kinerja dan hasil aplikasi cukup rendah. Hal tersebut berbeda dengan aplikasi yang dikembangkan dengan bahasa pemograman khusus yang sesuai dengan sistem operasinya. Sama halnya dengan kasus aplikasi

Travel Aja ini, developer menemukan banyak masalah *Debugging* dan kompabilitas. Maka, tim developer memutuskan untuk melakukan migrasi ke *kotlin native* dengan berbagai alasan yang dapat divalidasi. Sejalan dengan aplikasi yang semakin memiliki banyak fitur dan tampilan, sehingga termasuk aplikasi yang kompleks. Travel Aja memiliki banyak developer untuk mengembangkan aplikasi Android yang mereka miliki, serta terdapat juga developer magang yang berasal dari berbagai universitas di Indonesia. Jika dalam pengembangannya tidak menerapkan *design pattern* dan arsitektur, maka kode akan menjadi tidak konsisten dan sulit untuk dibaca dan dipahami oleh developer lain. Pembagian pekerjaan kepada developer juga menjadi sulit, sehingga kemungkinan terjadinya konflik saat pengembangan menjadi besar. Developer akan kesulitan untuk melakukan pembaruan kode apabila terjadi perubahan pada proses bisnis, serta akan mengalami kesulitan untuk melakukan pengujian dan perawatan pada kode. Dengan demikian, penting untuk menerapkan suatu arsitektur dan *design pattern* MVVM(*Model View View Model*) pada pengerjaan aplikasi Android.

1.2 Ruang lingkup

Program magang berlangsung selama enam bulan di Telkom Indonesia. Posisi penulis selama magang ditempatkan di Divisi DBT (*Digital Business & Technology*) sebagai Android developer, yang termasuk salah satu divisi di Direktorat *Digital Business* PT. Telkom Indonesia. Didampingi oleh Ibu Pramesti sebagai supervisor di tempat magang. Lalu, ada juga mas Ivan sebagai pegawai aktif PT. Telkom Indonesia dan juga senior Android developer yang membimbing penulis di tempat magangnya. Jadwal kerja yang relatif *flexibel*, membuat penulis dituntut mampu mandiri, membuat jadwal harian yang terstruktur dengan baik dan cermat. Penulis juga dibebaskan bekerja darima napun, selama penulis nyaman dengan kondisi serta pekerjaan yang berjalan dengan semestinya. Tugas Akhir ini akan berfokus pada proyek migrasi. Proyek migrasi adalah pengembangan aplikasi Travel Aja menggunakan SDK Android Studio dan bahasa pemrograman Kotlin untuk menggantikan aplikasi Travel Aja yang sudah ada dan dikembangan menggunakan *react native* oleh vendor. Pembahasan akan berfokus kepada penerapan *design pattern Model View View Model* (MVVM) dan *Clean Acrhitecture* pada pengembangan fitur rekomendasi rencana liburan pada proyek migrasi aplikasi Travel Aja. Penulis berfokus pada pengembangan fitur rekomendasi rencana liburan, fitur ini memiliki 4 bagian *page* yang terdiri dari *page* rencana liburan, *page* isi *field* rencana liburan, *page* cari lokasi liburan, dan *page* hasil rekomendasi rencana liburan. Selama

pengembangan fitur tersebut, penulis memiliki *guideline* penulisan standar code pemograman sebagai developer pengembangan aplikasi android di PT. Telkom Indonesia.

1.3 Tujuan

Berdasarkan latar belakang di atas, laporan magang ini ditulis untuk mencapai tujuan sebagai berikut.

a) Menjelaskan cara penerapan *design pattern Model View View Model* (MVVM) dalam pengembangan aplikasi Travel Aja.
b) Menjelaskan cara penerapan *Clean Architecture* dalam pengembangan aplikasi Travel Aja.
c) Menjelaskan manfaat migrasi aplikasi Travel Aja dengan menggunakan Kotlin *native*

1.4 Manfaat

Berdasarkan tujuan diatas, laporan akhir ini diharapkan dapat digunakan sebagai:

a) Dokumen cara penerapan *Clean Architecture* pada pengembangan aplikasi mobile menggunakan *Android Studio* dan bahasan pemograman Kotlin
b) Dokumen manfaat migrasi teknologi pada pengembangan aplikasi mobile menggunakan *Android Studio* dan bahasan pemograman Kotlin

1.5 Sistematika Penulisan

Adapun tugas akhir ini memiliki sistematika penulisan sebagai berikut.

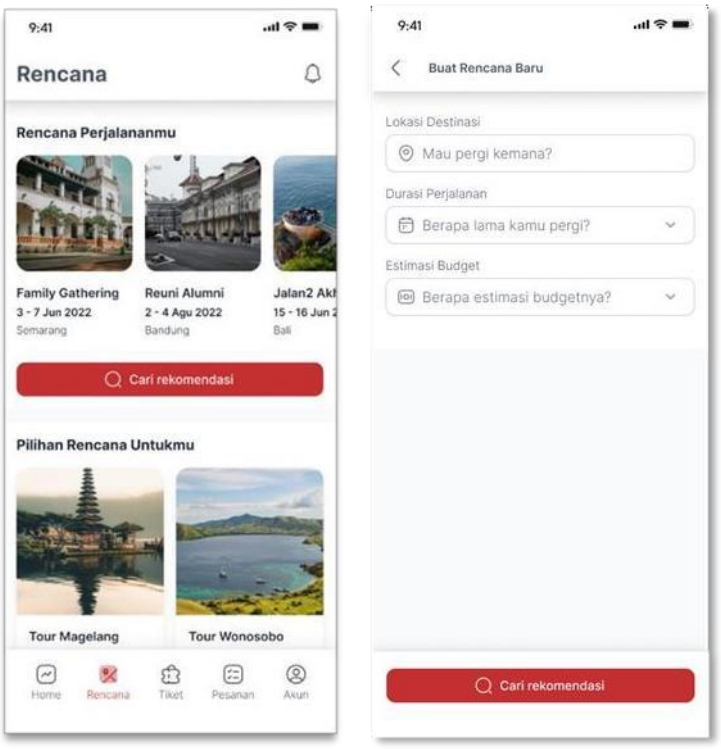
a) BAB I membahas tentang latar belakang, ruang lingkup magang, tujuan, dan manfaat penulisan dari Penerapan *Design Pattern* MVVM dan *Clean Architecture*.
b) BAB II membahas tentang dasar-dasar teori yang berkaitan dengan migrasi aplikasi, design pattern MVVM dan *Clean Architecture*.
c) BAB III menjelaskan tentang pelaksanaan magang, mulai dari *on-boarding* sampai dengan penerapan dari teori-teori yang telah dijelaskan pada bab sebelumnya.
d) BAB IV menjelaskan refleksi pembelajaran magang baik dari sisi teknis dan nonteknis.
e) BAB V membahas tentang kesimpulan manfaat yang telah diperoleh dari pelaksanaan magang, serta saran dan kritik dari penulis berdasarkan pengalaman yang penulis rasakan selama mengikuti magang.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Aplikasi Travel Aja

Travel Aja merupakan aplikasi berbasis android yang memiliki target pengguna yakni wisatawan dari berbagai negara. Aplikasi yang memiliki konsep memberikan fasilitas-fasilitas untuk mencari tempat wisata terbaik sesuai dengan diinginkan oleh pengguna. Calon pengguna akan dimudahkan dengan berbagai fitur seperti pencarian destinasi wisata, rekomendasi tempat wisata sesuai *budget*, dan rencana liburan yang disesuaikan dengan *budget* pengguna. Aplikasi ini dilengkapi juga fitur *sharing* foto dan video yang bisa dilihat sesama pengguna aplikasi.

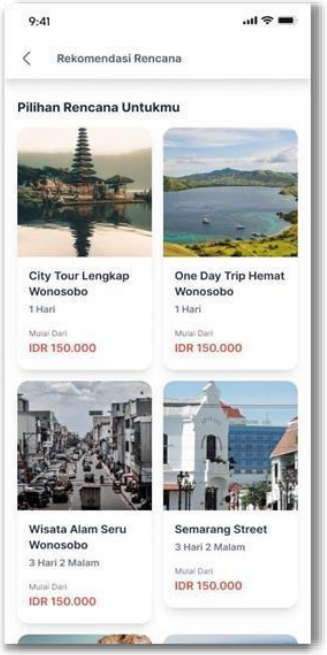
Pada gambar Gambar 2.1 merupakan tampilan fitur rekomendasi rencana liburan di aplikasi Travel Aja. Fitur ini akan bermanfaat bagi pengguna yang belum tahu lokasi wisata yang ingin dikunjungi.



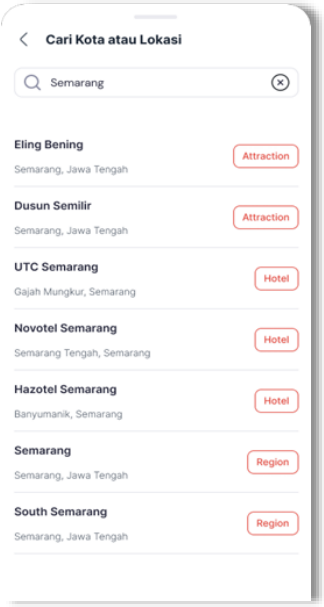
Gambar 2.1 Fitur Rencana Rekomendasi Liburan

Pada fitur ini pengguna dapat memilih lokasi yang ingin dikunjungi, Kemudian mengisi field durasi perjalanan dan estimasi *budget* pengguna. Lalu, aplikasi dapat memberikan saran wisata liburan yang sesuai.

Ketika pengguna melakukan klik pada *field* lokasi destinasi, maka pengguna akan dibawa ke halaman cari lokasi yang dapat dilihat pada Pada Gambar 2.3 halaman cari lokasi, pengguna dapat memilih satu lokasi tujuan destinasi wisata. Kemudian, bentuk hasil pencarian rekomendasi liburan pada



Gambar 2.2 Halaman Hasil Rekomendasi Rencana Liburan



Gambar 2.3 Halaman Pencarian Lokasi Liburan

2.2 Android

Menurut (Sihite, 2022) android adalah sebuah sistem operasi yang dikembangkan oleh Google untuk perangkat *mobile*, seperti *smartphone* dan tablet. Sistem operasi ini berbasis pada *kernel Linux* dan dirancang khusus untuk memungkinkan pengembangan dan penggunaan aplikasi yang beragam. Pada dasarnya, Android merupakan *platform* yang memungkinkan pengembang untuk membuat dan mengelola aplikasi yang dapat dijalankan di perangkat *mobile*. Android menyediakan lingkungan yang lengkap untuk pengembangan aplikasi, termasuk bahasa pemrograman, kerangka kerja (*framework*), dan alat pengembangan yang dibutuhkan.

Salah satu bahasa pemrograman yang umum digunakan dalam pengembangan aplikasi Android adalah Kotlin. Koltin merupakan bahasa pemrograman yang populer dan memiliki ekosistem yang kuat dalam pengembangan perangkat lunak. Penulis menggunakan bahasa kotlin untuk menulis kode program yang menjalankan logika dan fungsi-fungsi dari aplikasi Android.

Dalam pengembangan aplikasi Android, penggunaan bahasa pemrograman tersebut digunakan untuk membuat logika bisnis, interaksi dengan pengguna, akses ke sumber daya perangkat, dan integrasi dengan layanan atau API eksternal. Pengembang juga menggunakan kerangka kerja (*framework*) Android, seperti Android *SDK* (*Software Development Kit*), untuk mengakses fitur dan komponen sistem Android, seperti kamera, GPS, sensor, dan lainnya.

Dengan menggunakan bahasa pemrograman dan kerangka kerja yang sesuai, pengembang dapat merancang, membangun, dan menguji aplikasi Android dengan memperhatikan prinsip desain antarmuka pengguna yang baik dan performa yang optimal. Aplikasi Android yang dikembangkan dapat diunggah ke *Google Play Store* atau distribusikan melalui sumber lain untuk diunduh dan digunakan oleh pengguna Android di seluruh dunia.

Dengan adanya *platform* Android, pengguna dapat menikmati berbagai macam aplikasi yang beragam dan inovatif di perangkat mobile mereka. Android telah menjadi salah satu sistem operasi yang dominan dalam industri mobile, memberikan kesempatan bagi pengembang untuk menghadirkan pengalaman pengguna yang kaya dan memperluas kemungkinan dalam dunia teknologi mobile.

2.2.1 Android Studio

Android Studio adalah sebuah lingkungan pengembangan terintegrasi (*Integrated Development Environment/IDE*) yang digunakan untuk membuat aplikasi Android. IDE ini

dirancang khusus untuk memudahkan pengembangan, pengujian, dan *debugging* aplikasi Android. Dalam pengembangan aplikasi Android, Android Studio menyediakan berbagai fitur dan alat yang diperlukan oleh pengembang. IDE ini memiliki editor kode yang kuat dengan fitur penyorotan sintaks, saran kode, dan pemformatan otomatis. Pengembang dapat menulis kode program menggunakan bahasa pemrograman Java atau Kotlin. Android Studio juga mendukung pembangunan antarmuka pengguna (*User Interface/UI*) melalui tata letak (*layout*) visual yang intuitif. Pengembang dapat merancang tampilan aplikasi dengan *drag-and-drop*, menyesuaikan tata letak, dan mengatur elemen-elemen antarmuka.

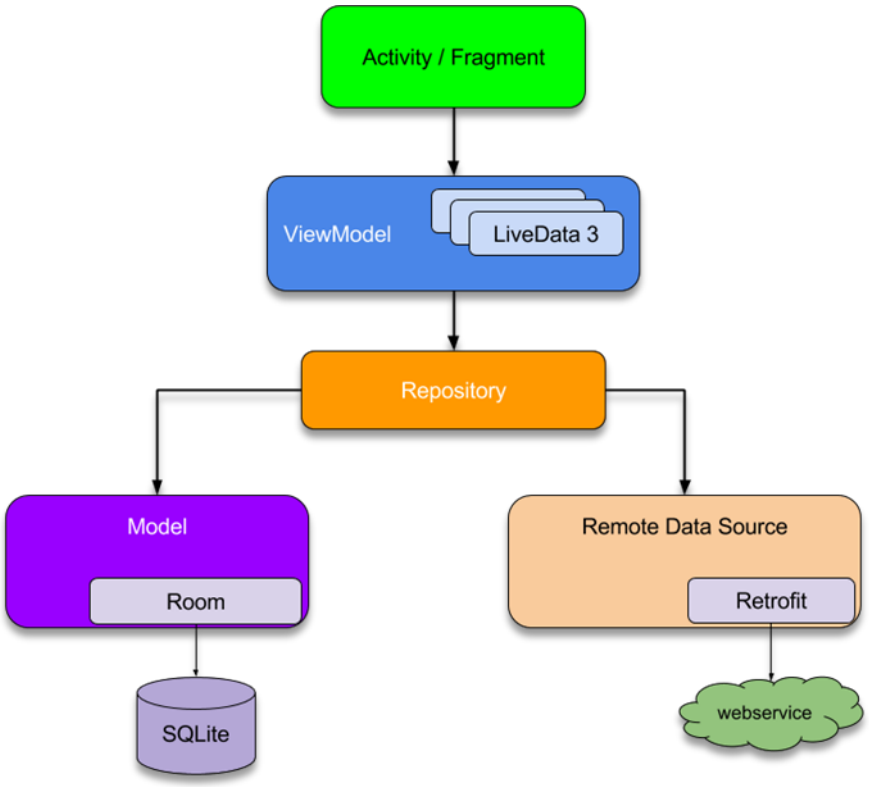
Selain itu, Android Studio menyediakan alat bantu untuk mendebug dan memantau aplikasi secara real-time. Pengembang dapat melacak log (*logcat*), menganalisis performa aplikasi, dan mengidentifikasi masalah yang mungkin terjadi. Android Studio juga terintegrasi dengan Android SDK (*Software Development Kit*) yang berisi berbagai alat pengembangan, pustaka, dan komponen yang dibutuhkan dalam membangun aplikasi Android. Pengembang dapat mengelola dependensi, mengatur versi SDK, dan mengakses berbagai sumber daya pengembangan yang disediakan oleh Google. Dengan Android Studio, pengembang dapat dengan mudah membuat, menguji, dan mempublikasikan aplikasi Android dengan lebih efisien. IDE ini memberikan lingkungan yang lengkap dan terstruktur untuk memfasilitasi pengembangan aplikasi Android yang profesional dan berkualitas tinggi.

2.2.2 Kotlin Native

Menurut (Fajri, 2022)Kotlin *native* adalah sebuah ekstensi dari bahasa pemrograman Kotlin yang memungkinkan pengembangan aplikasi lintas *platform*, termasuk di luar lingkungan *Java Virtual Machine* (JVM) seperti Android. Dalam pengembangan dengan Kotlin *Native*, kode sumber Kotlin dikompilasi menjadi kode mesin langsung yang dapat dijalankan tanpa perlu melalui jembatan komunikasi. Kotlin *native* memungkinkan pengembang untuk menggunakanKotlin dalam pengembangan aplikasi di berbagai *platform* seperti *iOS, macOS, Linux,* dan lainnya. Dengan menggunakan Kotlin *native*, pengembang dapat berbagi kode logika bisnis antara *platform* secara efisien, mengurangi upaya pengembangan yang harus dilakukan secaraterpisah untuk setiap *platform*.

2.2.3 Android Architecture Components

Menurut (Setiawan et al., 2021) *android architecture components* adalah sekumpulan komponen yang disediakan oleh Android *Jetpack* untuk membantu pengembangan aplikasi Android yang lebih terstruktur, mudah diuji, dan mudah diatur. Komponen-komponen ini dirancang untuk memisahkan tugas- tugas yang berbeda dalam aplikasi, mempromosikan arsitektur yang baik, dan menyediakan solusi umum untuk tantangan pengembangan yang umum dihadapi oleh para pengembang. Berikut Gambar 2.4 merupakan gambaran dari Android *Architecture Components.*



Gambar 2.4 Android *Architecture Components*

Sumber: (https://proandroiddev.com/)

Komponen utama dalam Android *Architecture Components* adalah:

a) *LiveData* adalah objek observasi yang dapat mengirimkan perubahan data secara responsif. Komponen ini dapat digunakan untuk menghubungkan data dari sumber seperti *database* atau jaringan dengan antarmuka pengguna. *LiveData* secara otomatis mengelola siklus hidup dan pembaruan data, sehingga memudahkan pengembang dalam mengelola perubahan data secara reaktif.

b) *ViewModel* membantu dalam memisahkan logika bisnis dari antarmuka pengguna. Komponen ini bertanggung jawab untuk menyimpan dan mengelola data yang diperlukan oleh tampilan atau antarmuka pengguna.
c) *Room* adalah komponen yang menyediakan lapisan abstraksi untuk mengakses dan berinteraksi dengan database SQLite dalam aplikasi Android. Komponen ini menyederhanakan proses penggunaan database dengan menyediakan anotasi yang mudah digunakan untuk mendefinisikan skema *database*, serta metode-metode yang nyaman untuk mengakses data.
d) *Paging* Komponen membantu dalam mengelola data yang terlalu besar untuk ditampilkan secara keseluruhan di layar. Dengan menggunakan *Paging*, pengembang dapat membagi data menjadi bagian-bagian kecil yang dapat dimuat secara bertahap saat pengguna menggulir layar. Hal ini meningkatkan kinerja aplikasi dan efisiensi penggunaan memori.
e) *Repository* adalah sebuah komponen dalam pengembangan perangkat lunak yang bertanggung jawab untuk mengatur akses dan interaksi antara aplikasi dengan sumber data eksternal, seperti *database, API*, atau penyimpanan lokal. Tujuan utama dari penggunaan *Repository* adalah untuk memisahkan logika bisnis dari akses data, sehingga aplikasi menjadi lebih modular, mudah diuji, dan fleksibel terhadap perubahan.
f) *Room* adalah salah satu komponen dari Android *Architecture Components* yang menyediakan *layer* abstraksi di atas SQLite untuk menyederhanakan dan mempermudah akses dan pengelolaan basis data lokal pada aplikasi Android.

Menurut (Oleh & Rahman, 2022), Gambar 2.4 menunjukkan alur bagaimana sebuah *Activity* atau *Fragment* melakukan permintaan *data* kepada *ViewModel*. Lalu, *ViewModel* meneruskan permintaan tersebut ke *Repository*, aplikasi dapat mengetahui apakah data berasal dari *local* atau *network*. Setelah *Repository* mendapatkan data, Kemudian data dikirimkan ke *ViewModel*. Ketika terjadi perubahan *data*, View akan otomatis melakukan pembaharuan tampilan berdasarkan data yang diterima. flow bagaimana sebuah *Activity/Fragment* melakukan permintaan data kepada *ViewModel*.

2.2.4 Model View View Model

Menurut (Bui, 2020)*Model View ViewModel (MVVM)* pattern adalah sebuah pola desain arsitektur perangkat lunak yang digunakan dalam pengembangan aplikasi berbasis antarmuka

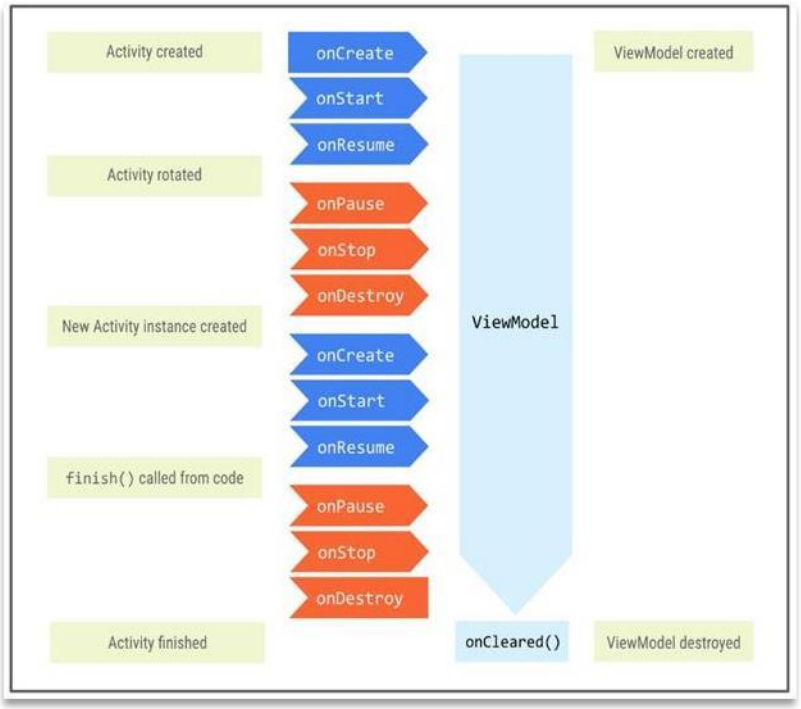
pengguna (UI), terutama dalam konteks aplikasi *desktop* dan mobile. Pola ini diperkenalkan oleh John Gossman pada tahun 2005, dan merupakan evolusi dari pola *Presentation Model* (PM) yang diperkenalkan oleh Martin Fowler pada tahun 2004.

MVVM terdiri dari tiga komponen utama:

a) *Model* mewakili data dan logika bisnis dari aplikasi. *Model* berfungsi untuk mengelola dan menyediakan data yang diperlukan oleh antarmuka pengguna.

b) *View* menampilkan antarmuka pengguna kepada pengguna. *View* berinteraksi dengan *ViewModel* untuk mendapatkan data yang akan ditampilkan, serta merespons input pengguna.

c) *ViewModel* bertindak sebagai penghubung antara *Model* dan *View*. *ViewModel* berfungsi untuk mengatur pemrosesan data, logika bisnis, serta memberikan *data* yang diperlukan oleh *View*. *ViewModel* juga berperan dalam menjaga pemisahan antara tampilan dan logika bisnis aplikasi.

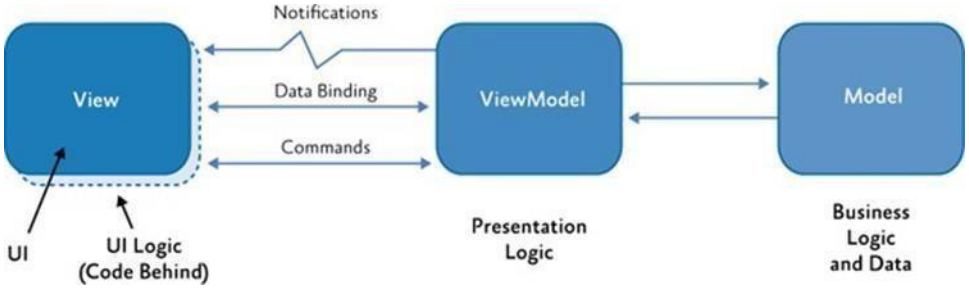


Gambar 2.5 Siklus hidup *View Model*

Sumber:(https://developer.android.com/topic/libraries/architecture/viewmodel?hl=id)

Model View View Model (MVVM) mempromosikan pemisahan yang jelas antara tampilan (*View*) dan logika bisnis (*ViewModel*). Ini memungkinkan pengembang untuk lebih mudah memperbarui dan menguji bagian-bagian yang berbeda dari aplikasi secara terpisah. Selain itu, dengan menggunakan *Model View View Model* (MVVM), pengembang dapat memanfaatkan

binding data yang kuat antara *View* dan *ViewModel*, yang memungkinkan perubahan pada data langsung tercermin pada tampilan, dan sebaliknya.



Gambar 2.6 Ilustrasi interaksi MVVM

Sumber:(https://syafdia.medium.com/mengenal-arsitektur-model-view-viewmodel-mvvm-di-android)

Komponen *View* berinteraksi dengan komponen *ViewModel* melalui *databinding*, *commands*, dan *change notification events* dengan perantara *live data* sebagai *observable data holder* yang dapat diobservasi oleh komponen *View*. Komponen *ViewModel* mengamati dan mengkoordinasikan pembaruan ke *Model*, melakukan konversi, memvalidasi, dan menggabungkan data yang akan ditampilkan di komponen *View*.

2.3 Clean Archiecture

Menuru (Duy, 2017)menguraikan bahwa dalam beberapa dekade terakhir, telah ada banyak ide dan pendekatan yang dibahas dalam konteks arsitektur sistem. Beberapa di antaranya adalah:

a. *Hexagonal Architecture* atau juga dikenal sebagai *Ports* and *Adapters*. Pendekatan ini dikembangkan oleh Alistair Cockburn dan diadopsi oleh Steve Freeman dan Nat Pryce dalam buku mereka yang berjudul "*Growing Object Oriented Software with Tests*".

b. *DCI (Data, Context, Interaction*) yang dikembangkan oleh James Coplien dan Trygve Reenskaug. Pendekatan ini menekankan pentingnya pemisahan antara data (*Data*), konteks (*Context*), dan interaksi (*Interaction*) dalam desain perangkat lunak. Konsep utamanya adalah mendefinisikan peran-peran dan konteks yang terlibat dalam suatu interaksi, dan menghubungkannya dengan data yang relevan.

c. *BCE (Boundary, Control, Entity)*, diperkenalkan oleh Ivar Jacobson melalui bukunya yang berjudul "*Object Oriented Software Engineering: A Use-Case Driven Approach*". Pendekatan ini membagi perangkat lunak menjadi tiga bagian utama:

Boundary (batasan), *Control* (kontrol), dan *Entity* (entitas). *Boundary* berfungsi sebagai penghubung antara pengguna dan sistem, Control mengatur alur kontrol dan logika bisnis, sedangkan *Entity* merepresentasikan data dan objek dalam sistem.

Menurut (Fajri, 2022)masing-masing sistem tersebut menghasilkan sebuah sistem yang memiliki karakteristik sebagai berikut.

a. *Independent of Framework*, tidak tergantung pada implementasi *framework* yang digunakan.

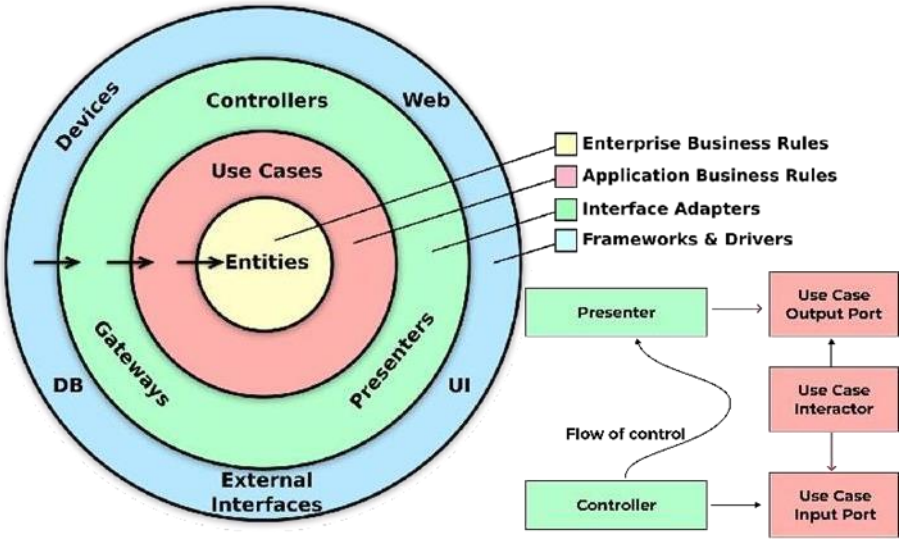
b. *Testable*, kode proses bisnis bisa dites tanpa memerlukan UI, *database*, atau *tools* lainnya.

c. *Independent of Userinterface*, UI dapat diubah dengan mudah tanpa harus mengubah keseluruhan sistem.

d. *Independent of Database*, tidak bergantung dengan *framework database* tertentu dan dapat diganti dengan mudah.

e. *Independent of External*, proses bisnis yang ada tidak perlu mengetahui apa yang ada di layer luar.

Gambar 2.7 menggambarkan varian layer dan data flow di Clean Architecture. EntitiesLayer bertanggung jawab terhadap enterprise business rules. Dekatan yang dibahas dalam konteks arsitektur sistem. Beberapa di antaranya adalah:



Gambar 2.7 *Clean Architecture*

Sumber: (androidexample365.com)

Agar arsitektur ini bekerja, *dependency rule* harus dipatuhi oleh developer dan ketergantuan hanya boleh mengarah kedalam. Kode di *layer* luar tidak boleh mengetahui kode di dalam *layer*, serta tidak boleh terpengaruh oleh perubahan yang terjadi di *layer* luar. Ketika berinteraksi antar *layer*, Developer harus menggunakan *dependency inversion principle*.

Biasanya *entities* berbentuk sebuah *object* atau sekumpulan struktur data dan fungsi. Perubahan pada aplikasi seharusnya tidak mempengaruhi entity *layer*. *Use cases layer* bertanggung jawab terhadap *application business rules*. Perubahan pada *layer* ini tidak akan mempengaruhi *entities* dan juga *layer* ini tidak boleh terpengaruh oleh perubahan pada *external layer*. *Interface adapter layer* menyediakan cara mentransformasi format data dari *entities* dan *usecases* ke format yang sesuai untuk *external agency* seperti *database* atau website. Kode pada *layer* ini seharusnya tidak tahu tentang *framework* atau *driver* yang digunakan oleh aplikasi. *Frameworks* dan *drivers layer* terdiri dari *framework* atau *tools* seperti *database*, web *framework*, dan lain-lain.

2.3.1 Entities

Menurut (Nugroho, 2010)*Entities* adalah komponen dalam arsitektur perangkat lunak yang merepresentasikan objek-objek penting dalam sistem. Mereka menggambarkan data dan perilaku yang terkait dengan domain bisnis aplikasi. *Entities* biasanya merupakan bagian inti dari sistem, dan tidak tergantung pada *layer-layer* lainnya. Mereka mewakili konsep bisnis yang spesifik dan memiliki aturan-aturan bisnis yang terkait. *Entities* berfungsi sebagai tempat penyimpanan data dan menyediakan metode untuk memanipulasi dan mengakses data tersebut. Dalam pemrograman berorientasi objek, *Entities* sering kali diimplementasikan sebagai kelas atau objek-objek yang memiliki atribut-atribut dan metode-metode yang relevan dengan *domain* bisnis yang sedang dibangun.

2.3.2 Use Case

Menurut (Kurniawan et al., n.d.) Usecase merupakan komponen dalam arsitektur perangkat lunak yang menggambarkan interaksi antara aktor atau pengguna sistem dengan entities. Usecase menjelaskan bagaimana aktor atau pengguna dapat menggunakan entities untuk mencapai tujuan bisnis atau tugas tertentu. Usecase mewakili alur logika bisnis yang spesifik dan menjembatani antara layer presentasi (seperti antarmuka pengguna) dengan layer domain (entities). Usecase biasanya terdiri dari serangkaian langkah-langkah atau aksi yang dilakukan oleh aktor atau pengguna untuk menghasilkan hasil yang diinginkan. Dalam implementasinya, usecase dapat diwujudkan sebagai kelas atau fungsi yang mengimplementasikan alur logika bisnis yang spesifik sesuai dengan kebutuhan sistem.

2.3.3 Interface Adapter

Menurut (Putra et al., n.d.)*Interface Adapter* adalah komponen dalam arsitektur perangkat lunak yang bertanggung jawab untuk menghubungkan antara *layer* aplikasi dengan *layer* eksternal, seperti *database*, jaringan, atau sistem pihak ketiga. *Interface Adapter* bertindak sebagai perantara antara *layer* aplikasi dan *layer* infrastruktur, sehingga memungkinkan komunikasi dan interaksi antara keduanya.

Interface Adapter dapat berfungsi untuk mengubah format *data* dari satu *layer* ke *layer* lainnya, menerapkan protokol komunikasi yang diperlukan, atau menangani pemetaan antara representasi data di *layer* aplikasi dengan format yang digunakan di layer eksternal.

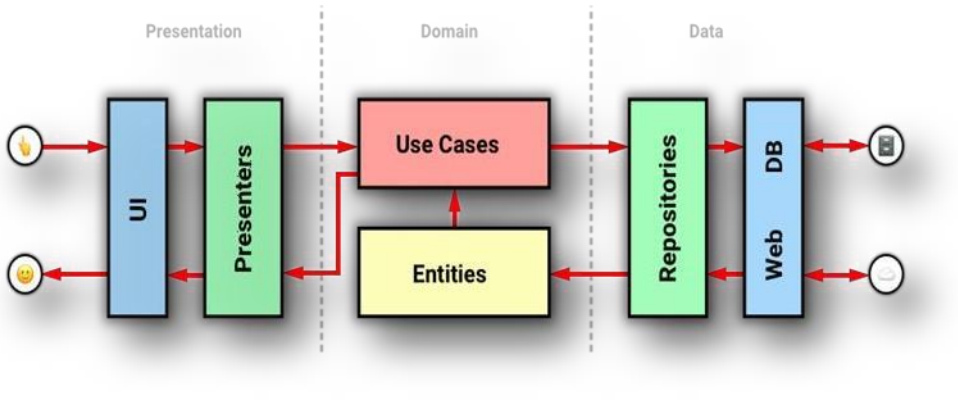
2.3.4 Framework and Drivers

Menurut (Makarenko et al., 2017)*Framework and Drivers Adapter* adalah komponen dalam arsitektur perangkat lunak yang berfungsi sebagai penghubung antara *framework* atau

library dengan *layer* aplikasi. *Framework and Drivers Adapter* membantu dalam mengintegrasikan komponen eksternal, seperti *framework* UI atau *library* eksternal, dengan aplikasi yang sedang dikembangkan. Komponen ini bertanggung jawab untuk mengatur interaksi antara *framework* atau *library* dengan kode aplikasi, sehingga memungkinkan aplikasi untuk menggunakan fitur- fitur yang disediakan oleh komponen eksternal tersebut.

2.3.5 Android Clean Architecture

Penggunaan *Clean Architecture* pada proyek Android pada umumnya dibagi menjadi tiga layer, yaitu *presentation*, *domain*, dan *data*. Pembagian *layer* tersebut dapat dilihat pada



Gambar 2.8 *Layer* Android *Clean Architecture*

Sumber: (https://ru.stackoverflow.com/)

Berikut adalah penjelasan mengenai komponen dari setiap *layer* yang terdapat pada Android *Clean Architecture* menurut (Duy, 2017)

a) Pada *Presentation Layer* terdapat *UI* dan *Presenter/ViewModel* yang mengatur tampilan berdasarkan update data terbaru. *Presentation Layer* bergantung kepada *Use Case* di *Domain Layer*.
b) Pada *Domain Layer* merupakan *layer* inti yang berkaitan dengan bisnis *model* yang terdapat *Entities*, *Use Case*, dan *Repository Interface*.
c) Hasil implementasi *Repository* dan *Data Source* yang merupakan isi dari *data layer*. *Data* bisa berasal dari *local data source* (database lokal) atau *remote data source* (*network* atau *REST API*).

2.3.6 *Dependecy Inversion Principle* dan Depedency Injection

Dependency Inversion Principle (DIP) adalah salah satu prinsip dalam desain perangkat lunak yang mengemukakan bahwa *modul-level* tinggi tidak boleh bergantung pada *modul-level*

rendah. Prinsip ini juga mengatakan bahwa detail implementasi harus bergantung pada abstraksi, bukan sebaliknya. DIP mempromosikan hubungan yang longgar antara komponen-komponen dalam sistem dengan menggunakan konsep injeksi ketergantungan.

Dalam konteks DIP, *modul-level* tinggi adalah modul yang berisi logika bisnis atau aturan bisnis inti, sedangkan modul-level rendah adalah modul yang berisi detail implementasi atau komponen teknis. Dengan menerapkan DIP, kita dapat mencapai fleksibilitas dan pemeliharaan yang lebih baik dalam sistem karena modul-level tinggi tidak bergantung pada detail implementasi yang dapat berubah.

Dikutip dari (Wibowo, n.d.), *Dependency Injection* adalah sebuah design pattern yang menerapkan prinsip *inversion of control*. Pada umumnya *dependency injection* memiliki sebuah *service* yang akan digunakan oleh *client* dan sebuah *interface* yang mendefinisikan bagaimana *client* menggunakan *service* tersebut, serta sebuah *injector* yang menginstansiasi *service* tersebut dan meng-*inject service* ke *client*.

2.4 Scrum

Menurut (Nugroho, 2010) *scrum* adalah sebuah metode pengembangan perangkat lunak yang berfokus pada pengelolaan proyek secara *adaptif* dan *kolaboratif*. Metode ini menekankan pada transparansi, inspeksi, dan adaptasi dalam proses pengembangan. *Scrum* membagi proyek menjadi iterasi pendek yang disebut "*sprint*" dan menggunakan serangkaian peran, acara, dan artefak untuk mengatur dan mengawasi pekerjaan.

Dalam *scrum*, terdapat tiga peran utama *product owner*, *scrum master*, dan Tim Pengembang. *product owner* bertanggung jawab untuk mengartikulasikan kebutuhan pengguna, merencanakan *backlog* produk, dan menentukan prioritas pekerjaan. *scrum master* bertindak sebagai fasilitator untuk memastikan kelancaran tim dan menerapkan prinsip-prinsip Scrum. Tim Pengembang adalah kelompok yang melakukan pekerjaan aktual dalam mengembangkan produk.

Selama *sprint*, tim mengadakan serangkaian acara, termasuk *sprint planning* untuk merencanakan pekerjaan, *daily scrum* untuk sinkronisasi harian, *sprint review* untuk meninjau hasil pekerjaan, dan *sprint retrospective* untuk mempelajari pelajaran dari sprint sebelumnya. Selama *sprint*, tim bekerja untuk menyelesaikan *item backlog* yang dipilih.

Scrum juga menggunakan artefak seperti *product backlog*, *sprint backlog*, dan *increment*. *product backlog* berisi daftar kebutuhan dan fitur yang harus dikerjakan, sementara *sprint backlog* adalah daftar *item* yang dipilih untuk sprint tertentu. Increment adalah versi produk

yang dapat digunakan dan memiliki penambahan fungsionalitas setelah setiap sprint. Dengan menggunakan *scrum*, tim pengembangan dapat bekerja secara kolaboratif, responsif terhadap perubahan, dan menghasilkan produk dengan cepat. Metode ini mempromosikan transparansi, akuntabilitas, dan adaptasi dalam pengembangan perangkat lunak.

2.4.1 Scrum Master

Menurut (AUTOMATA & 2021, 2021)*scrum master* adalah peran dalam metodologi scrum yang bertanggung jawab untuk memfasilitasi dan memastikan kelancaran implementasi *scrum* dalam sebuah proyek. Seorang *scrum master* bekerja sebagai pemimpin servan yang membantu tim pengembangan dan pemangku kepentingan lainnya untuk memahami dan menerapkan prinsip-prinsip serta praktik *scrum*.

Tugas utama *scrum master* meliputi menghilangkan hambatan dan mengatur lingkungan yang mendukung agar tim dapat bekerja secara efektif. Mereka bekerja untuk memastikan bahwa tim mengikuti proses *scrum* dengan benar, menjaga transparansi dalam komunikasi, dan memfasilitasi pertemuan dan acara Scrum seperti Sprint Planning, Daily Scrum, *sprint review*, dan *sprint retrospective*.

2.4.2 Development Team

Menurut (Sihite, 2022)*development team* adalah kelompok orang yang terlibat dalam pengembangan perangkat lunak menggunakan metodologi *scrum*. Tim ini terdiri dari individu-individu yang memiliki keterampilan dan kompetensi yang diperlukan untuk merancang, mengembangkan, menguji, dan mengirimkan produk yang bernilai kepada pelanggan.

Tugas utama *development team* adalah menerima dan mengerjakan item *backlog* yang dipilih selama *sprint planning*. Mereka bekerja secara kolaboratif untuk merancang, mengimplementasikan, dan menguji fitur-fitur yang diperlukan dalam pengembangan produk. Anggota tim harus memiliki keterampilan lintas-fungsional sehingga mereka dapat bekerja secara mandiri dan saling mendukung dalam mencapai tujuan *sprint*.

2.4.3 Product Owner

Menurut (Rizaldy & Dirgahayu, 2020)*product owner* adalah peran dalam metodologi *scrum* yang bertanggung jawab atas pengelolaan kebutuhan dan pengembangan produk. Seorang *product owner* memiliki pemahaman yang mendalam tentang kebutuhan pengguna, pasar, dan visi bisnis yang ingin dicapai dengan produk yang dikembangkan.

Tugas utama *product owner* adalah membuat dan mengelola *product backlog*, yaitu daftar prioritas fitur-fitur dan perubahan yang harus diimplementasikan dalam produk. Mereka

bekerja sama dengan para pemangku kepentingan dan tim pengembangan untuk mengumpulkan persyaratan, menganalisis kebutuhan pengguna, dan memastikan bahwa produk yang dikembangkan sesuai dengan visi dan strategi bisnis perusahaan.

2.4.4 Sprint

Menurut (Rizaldy & Dirgahayu, 2020)*Sprint* adalah periode waktu terbatas dan terstruktur dalam metodologi *scrum*, di mana tim pengembangan bekerja untuk menghasilkan suatu *implementasi* produk yang siap untuk dirilis. *sprint* memiliki durasi tetap, biasanya berkisar antara 1 hingga 4 minggu, dan dimulai dengan *sprint planning meeting* dan berakhir dengan *sprint review* dan *sprint retrospective*.

Sprint pada pengembangan aplikasi travel di PT. Telkom Indonesia aja berlangsung selama 4 minggu, *Sprint* bertujuan untuk menghasilkan hasil kerja yang bernilai dan dapat diuji langsung setelah selesai *Sprint*. Ini memungkinkan tim pengembangan dan pemangku kepentingan untuk melihat perkembangan produk secara berkala dan beradaptasi dengan perubahan kebutuhan yang muncul selama *Sprint*.

2.4.5 Sprint Planning

Menurut (AUTOMATA & 2021, 2021)*Sprint Planning* adalah pertemuan di awal setiap *sprint* dalam metodologi *scrum*, di manatim pengembangan bekerja sama dengan *product owner* untuk merencanakan pekerjaan yangakan dilakukan selama *sprint* tersebut. Dalam *sprint planning*, tim mengidentifikasi dan memilih item-item dari *product backlog* yang akan diimplementasikan dalam *sprint*, serta memperkirakan waktu dan usaha yang diperlukan untuk menyelesaikannya.

Hasil dari *sprint planning* adalah *sprint backlog*, yang berisi daftar *item* yang akan dikerjakan dalam *sprint*, termasuk tugas-tugas yang perlu diselesaikan dan estimasi waktu yang dibutuhkan untuk setiap tugas. *sprint planning* membantu tim pengembangan dan *product owner* untuk memiliki pemahaman yang jelas tentang tujuan dan pekerjaan yang akandilakukan dalam *sprint* tersebut.

2.4.6 Sprint Goal

Menurut (Rizaldy & Dirgahayu, 2020)*Sprint Goal* adalah tujuan yang ditetapkan untuk setiap *sprint* dalam metodologi *scrum*. Tujuan ini memberikan fokus dan arah bagi tim pengembangan dalam menyelesaikan pekerjaan yang direncanakan selama *sprint*.

Sprint Goal biasanya disusun oleh *product owner* berdasarkan kebutuhan bisnis dan prioritas yang ada. Tujuan ini dapat mencakup aspek fungsional maupun non-fungsional yang

ingin dicapai dalam *sprint* tersebut. *Sprint Goal* haruslah spesifik, terukur, mencapai hasil yang bernilai, dan memberikan arah yang jelas kepada tim pengembangan.

2.4.7 Sprint Review

Menurut (AUTOMATA & 2021, 2021)*Sprint Review* adalah kegiatan yang dilakukan pada akhir setiap *sprint* dalam metodologi *scrum*. Tujuan dari *sprint review* adalah untuk mengevaluasi hasil kerja yang telah diselesaikan oleh tim pengembangan selama *sprint* tersebut.

Pada *Sprint Review*, tim pengembangan mempresentasikan kepada stakeholders (pemangku kepentingan) hasil kerja yang telah mereka selesaikan selama *sprint*. Ini termasuk demonstrasi dari fitur-fitur atau fungsi-fungsi baru yang telah dikembangkan.

2.4.8 Sprint Retrospective

Sprint Retrospective adalah pertemuan yang dilakukan oleh tim pengembangan setelah selesainya Sprint dalam metodologi *scrum*. Tujuan dari *sprint retrospective* adalah untuk merefleksikan dan mengevaluasi proses pengembangan yang telah dilakukan selama Sprint tersebut.

Pentingnya *sprint retrospective* adalah untuk memastikan pembelajaran terus-menerus dan perbaikan berkelanjutan dalam pengembangan produk. Dengan refleksi dan evaluasi yang teratur, tim pengembangan dapat terus meningkatkan cara mereka bekerja dan menghadapi tantangan yang muncul.

Secara keseluruhan, *sprint retrospective* adalah wadah yang penting bagi tim pengembangan untuk memperbaiki proses kerja mereka, meningkatkan kolaborasi, dan mencapai tujuan pengembangan dengan lebih baik

2.4.9 Product Backlog

Product Backlog adalah daftar prioritas yang berisi semua fitur, perbaikan, dan perubahan yang diinginkan untuk produk yang sedang dikembangkan. Daftar ini berfungsi sebagai referensi utama bagi tim pengembangan dalam merencanakan dan mengatur pekerjaan selama *sprint*. Dalam *product backlog*, setiap item memiliki deskripsi yang jelas, estimasi waktu atau kompleksitas, dan prioritas yang ditentukan oleh *product owner*. *Item* yang memiliki prioritas lebih tinggi biasanya ditempatkan di atas daftar, sementara *item* dengan prioritas lebih rendah ditempatkan di bagian bawah.

Secara singkat, *product backlog* adalah daftar prioritas yang menggambarkan semua kebutuhan dan perubahan yang diinginkan untuk produk. Daftar ini membantu tim pengembangan dalam merencanakan dan melaksanakan pekerjaan dengan efektif dan efisien.

2.5 Tinjauan Pustaka

Terdapat sejumlah penelitian yang telah dilakukan terkait pengembangan aplikasi android menggunakan *Integrated Development Environment* (IDE) Android Studio. Penelitian yang dilakukan oleh (Bui, 2020) yang berjudul "*Android Applications for Student's Personal Finance*" menjelaskan tentang penerapan *Model View View Model* (MVVM) design. Penelitian yang berjudul "Pengembangan Aplikasi Berbasis Android Untuk Meningkatkan Kepatuhan Pasien Gagal Jantung Dalam Merawat Diri Di Rumah" oleh (Putra et al., n.d.) ini membahas tentang pengembangan aplikasi berbasis Android, mulai dari metode pengembangan, *database, design pattern*, dan arsitektur aplikasi. Pada penelitian ini juga menjelaskan cara implementasi *Model View View Model* (MVVM) pada pengembangan aplikasi, tetapi untuk *Clean Architecture* sendiri tidak ada penjelasan cara penerapannya pada saat pengembangan *pattern* pada pengembangan aplikasi Android menggunakan Android Studio dengan Bahasa Kotlin.

Selanjutnya, penelitian juga dilakukan oleh (Aldiansyah et al., 2021a) yang berjudul "Pengembangan Aplikasi Ngobat: Aplikasi Ketaatan Regimen Pengobatan menggunakan *Gamification* pada *Platform* Android" adalah penelitian tentang pengembangan aplikasi yang bertujuan untuk menjadi pengingat guna membantu pengguna dalam regimen pengobatan. Pada penelitian ini dibahas mengenai design *pattern Model View View Model* (MVVM) yang diterapkan dalam pengmebangan aplikasi, akan tetapi pengembangan tersebut tidak menerapkan *Clean Architecture* dalam pengembangannya.

Selanjutnya, penelitian yang dilakukan oleh (Aldiansyah et al., 2021b) yang berjudul "Pengembangan Aplikasi Ngobat: Aplikasi Ketaatan Regimen Pengobatan menggunakan *Gamification pada Platform* Android" adalah penelitian tentang pengembangan aplikasi yang bertujuan untuk menjadi pengingat guna membantu pengguna dalam regimen pengobatan. Pada penelitian ini dibahas mengenai design *pattern Model View View Model* (MVVM) yang diterapkan dalam pengmebangan aplikasi, akan tetapi pengembangan tersebut tidak menerapkan Clean Architecture dalam pengembangannya.

Penelitian yang lain juga dilakukan oleh (Oleh & Rahman, 2022)yang berjudul "Penerapan *Design Pattern* MVVM dan *Clean Architecture* Pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree)" adalah penelitian tentang studi kasus aplikasi Agree yang bertujuan

untuk mendukung aktivitas petani. Pada penelitian ini dibahas struktur package yang menggunakan *Clean Architecture*, cara penerapan *design pattern* MVVM dan *Clean Architecture* pada pengembangan aplikasi Android.

Penelitian lainnya yang dilakukan oleh (Makarenko et al., 2017)yang berjudul "*An Architectural Approach for Quality Improving of Android Applications Development which Implemented to Communication Application for Mechatronics Robot Laboratory Onaft*". Penelitian tersebut menerapkan *Clean Architecture* dan *Model View Controller design pattern*. Pada penelitian tersebut masih menerapkan *MVC design pattern* yang mana tidak sesuai dengan rekomendasi dari Google saat ini, yaitu MVVM *design pattern*.

Tabel 2.1 Arsitektur dan *Design Pattern* Aplikasi

No	**Judul Penelitian**	**Arsitektur Aplikasi**	***Design Pattern***
1	*Android Applications for Student's Personal Finance*	*Clean Architecture*	MVVM
2	*Pengembangan Aplikasi Berbasis AndroidUntuk Meningkatkan Kepatuhan Pasien Gagal Jantung Dalam Merawat Diri Di Rumah*	Tidak Diketahui	MVVM
3	*Pengembangan Aplikasi Ngobat: Aplikasi Ketaatan Regimen Pengobatan menggunakan Gamification pada Platform Android*	Tidak Diketahui	MVVM
4	*An Architectural Approach for Quality Improving of Android Applications Development which Implemented to Communication Application for Mechatronics Robot Laboratory Onaft*	*Clean Architecture*	MVC
5	*Penerapan Design Pattern MVMM Dan CleanArchitecture Pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree)*	*Clean Architecture*	MVVM

Berdasarkan Tabel 2.1 diketahui bahwa dari lima pengembangan aplikasi Android, dua diantaranya telah menerapkan arsitektur dan *design pattern*, sementara yang lainnya hanya menerapkan design pattern. Hanya satu dari lima penelitian tersebut yang menerapkan *Clean Arhitecture* dan *design pattern MVVM*.

BAB III
PELAKSANAAN MAGANG

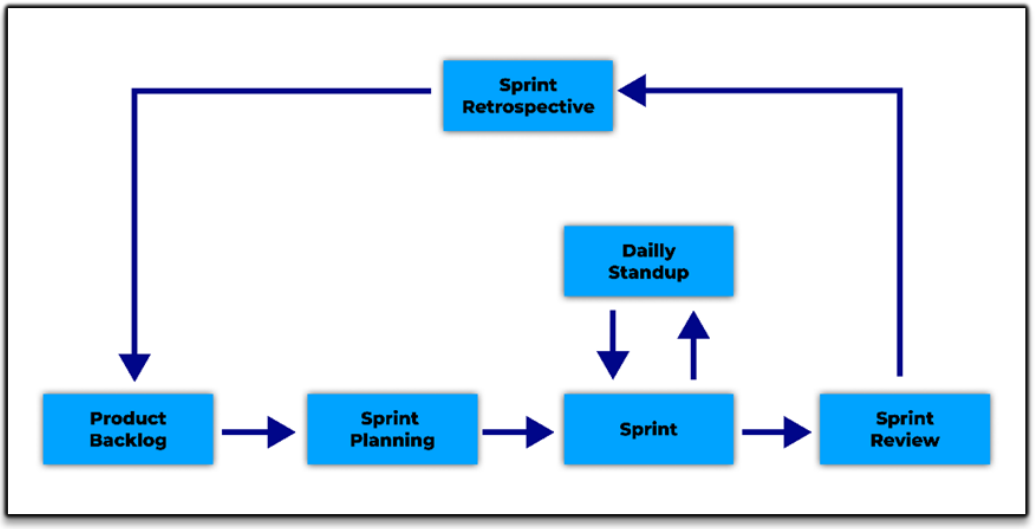
3.1 Android Developer

Program magang di Telkom Direktorat Digital Business membuka beberapa posisi, yaitu developer, *data scientist*, *designer*, *researcher*, *general*, *engineer*, dan *marketing*. Penulis berposisi sebagai Android developer yang ditempatkan di divisi *Travel n Tourisme* (TNT), Didampingi oleh ibu Pramesti sebagai supervisor di tempat magang. Lalu, ada mas Ivan sebagai pegawai aktif telkom sebagai *Senior* Android Developer yang tergabung juga di dalam *project* migrasi.

Di divisi *Travel n Tourisme* (TNT), penulis mengerjakan proyek Migrasi. Proyek migrasi adalah proyek pembuatan aplikasi Travel Aja yang saat ini sudah berubah nama menjadi Haio menggunakan Android *native* untuk menggantikan Aplikasi Travel Aja yang sebelumnya sudah dikembangkan menggunakan *react native*. Pembahasan pelaksanaan magang ini akan berfokus pada fitur Rekomendasi liburan dan share story liburan yang penulis kerjakan pada proyek Migrasi dengan menerapkan *Clean Architecrure* dan *design pattern* MVVM.

3.2 Manajemen Proyek

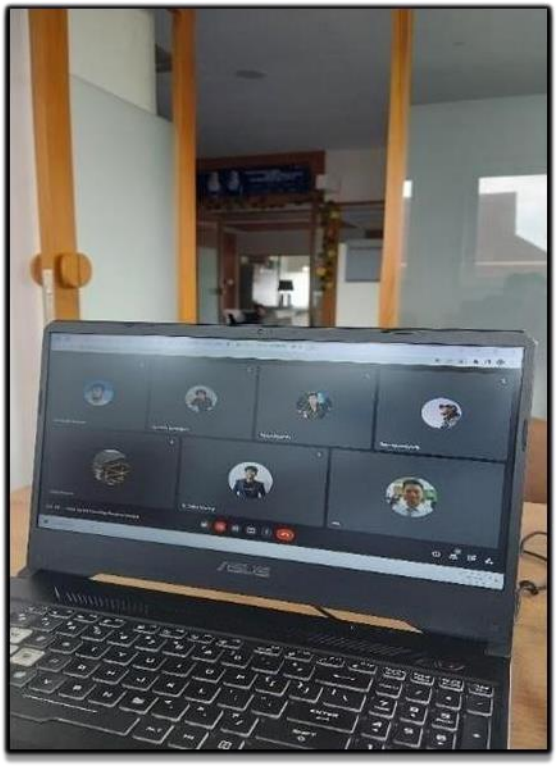
Metodologi yang digunakan dalam pengembangan aplikasi Travel Aja adalah *scrum*.
Gambar 3.1 mengilustrasikan proses pengembangan aplikasi Travel Aja.



Gambar 3.1 Tahap pengembangan aplikasi Travel Aja

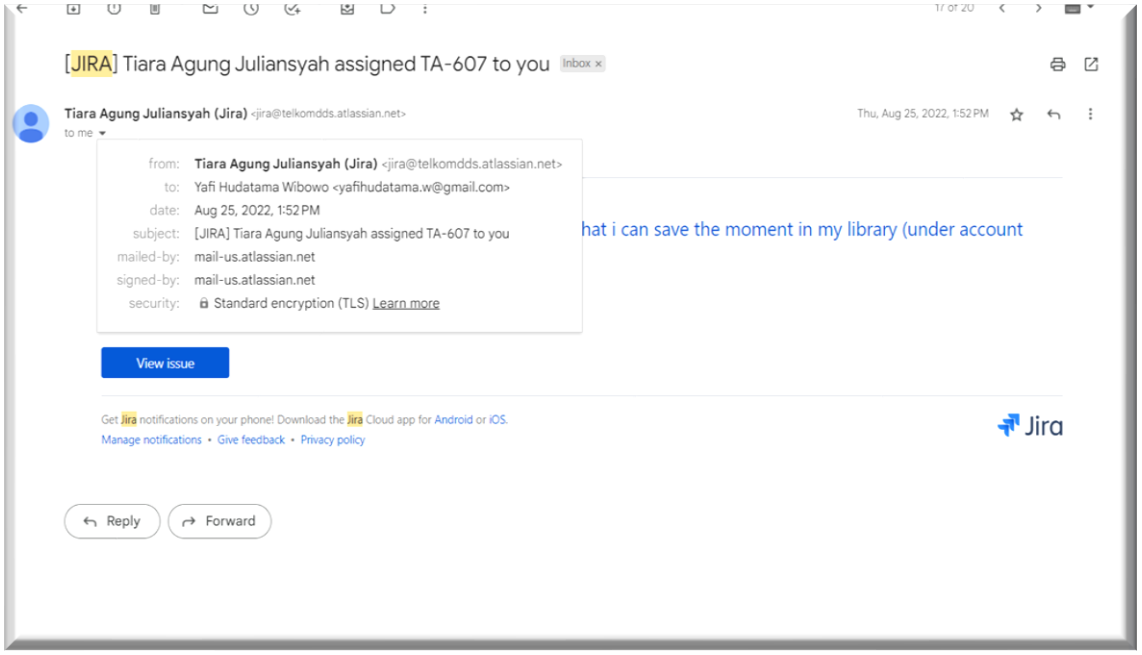
Product Backlog merupakan daftar pekerjaan yang bertujuan untuk meningkatkan kualitas produk. *Product Owner* akan melakukan pembahasan terkait *product backlog* yang akan memecah *product backlog* menjadi items yang lebih kecil dan detail. Jika *product backlog* dianggap sudah siap, maka akan dilanjutkan ke tahap *sprint planning*. Pada tahap *sprint planning product owner*

akan membahas mengapa *product backlog item* yang dibawa ke *sprint planning* penting untuk meningkatkan kualitas produk. Gambar 3.2 adalah acara *sprint planning* pada proyek Migrasi yang dilakukan bersama dengan *product owner* dan *developer*.

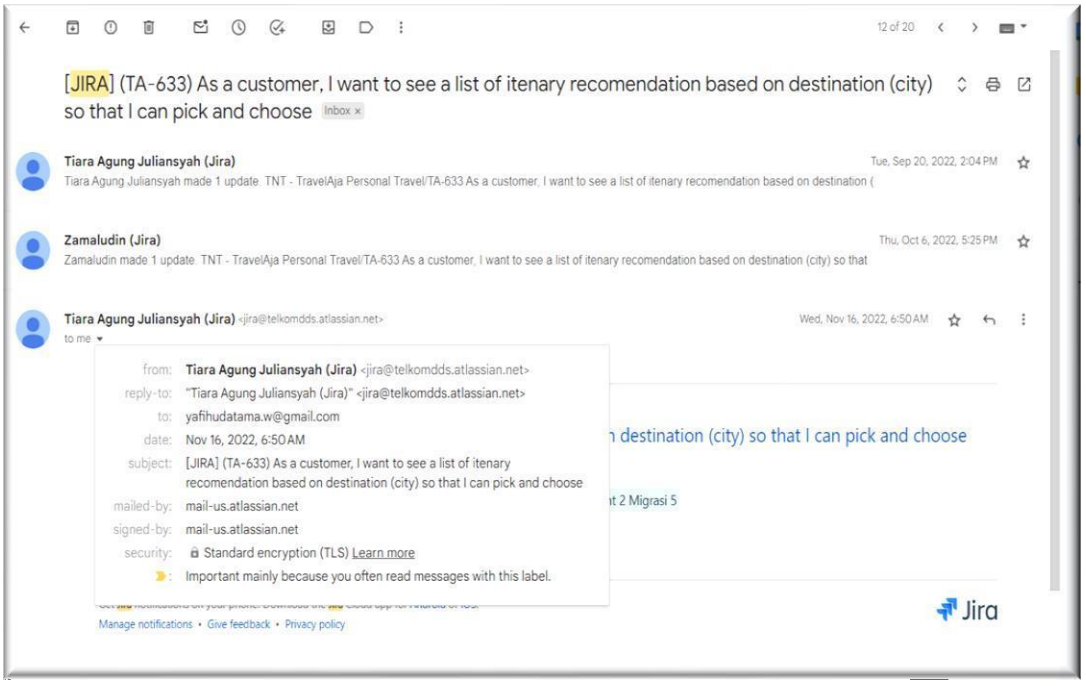


Gambar 3.2 *Sprint Planning* proyek migrasi

Product Owner bersama developer akan berdiskusi untuk memilih *product backlog item* yang akan dimasukkan ke dalam *sprint*. Ketika *product backlog item* sudah terpilih, maka akan dilanjutkan ke acara *sprint*. Sprint adalah proses pengerjaan *backlog item* yang sudah terpilih dengan durasi dua minggu (10 hari kerja). Lama waktu Sprint dapat disesuaikan dengan kebutuhan perusahaan. gambar 3.7 merupakan beberapa *backlog item* yang penulis kerjakan. Setiap developer memiliki *backlog item* masing-masing seperti gambar 3.8

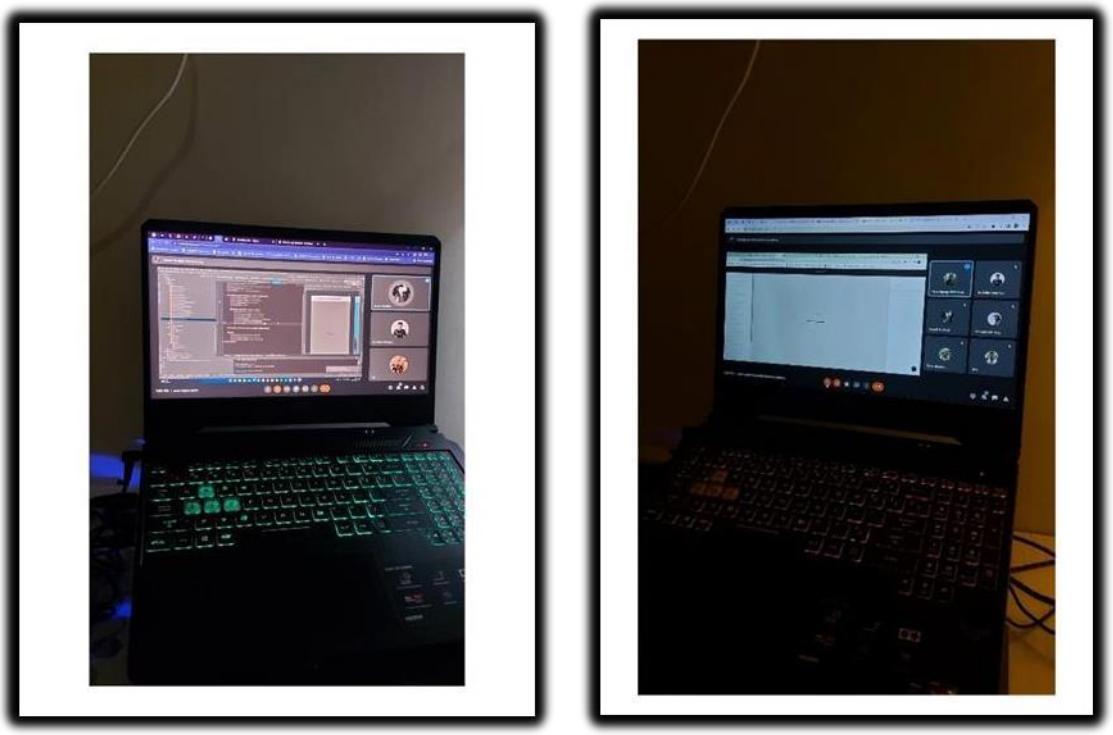


Gambar 3.3 *Backlog Item Sprint*



Gambar 3.4 *Backlog Item Sprint* Developer

Dengan penerapan *Clean Architecture* memudahkan pembagian *backlog item*, serta memperjelas tanggung jawab dari setiap developer terhadap *backlog item* yang dikerjakan. Pemisahan konsentrasi yang didapatkan dari penerapan *clean architecture* juga memperkecil terjadinya konflik ketika developer melakukan pengembangan, karena developer tidak akan melakukan perubahan pada kode yang tidak berkaitan dengan *backlog item*-nya. Pada saat Sprint berjalan, akan dilakukan *daily standup* yang dilakukan setiap hari dengan waktu dan tempat yang telah disepakati secara bersama. Seperti gambar 3.9 merupakan acara *daily*



Gambar 3.5 Acara Daily Standup

Acara ini bertujuan untuk memeriksa kemajuan dan perkembangan *sprint backlog*, serta kendala yang dialami selama proses pengerjaan. Kendala-kendala yang dialami oleh developer dapat didiskusikan pada *daily standup*, kemudian bersama-sama mencari solusi untuk menyelesaikannya.

Setelah *sprint* telah selesai dilakukan, maka akan dilanjutkan ke tahap *sprint review*. Developer akan melakukan demo *sprint backlog* yang telah selesai dikerjakan kepada *product owner*. Kemudian akan dilakukan inspeksi terkait *sprint backlog* yang belum dapat diselesaikan dalam satu Sprint dan menentukan potensi backlog untuk *sprint* selanjutnya.

Sprint Retrospective adalah proses perencanaan untuk meningkatkan kualitas dan efektivitas *sprint*. *Scrum Team* akan melakukan pengkajian jalannya *sprint* dan melakukan identifikasi untuk mengetahui apa saja hal yang dapat meningkatkan efektivitas *sprint*, serta masalah-masalah yang dapat mengurangi efektivitas *sprint*. Pengkajian tersebut berguna untuk meningkatkan kualitas dan efektivitas *sprint* selanjutnya.

3.3 Pengembangan Fitur Rekomendasi Rencana Liburan pada proyek migrasi

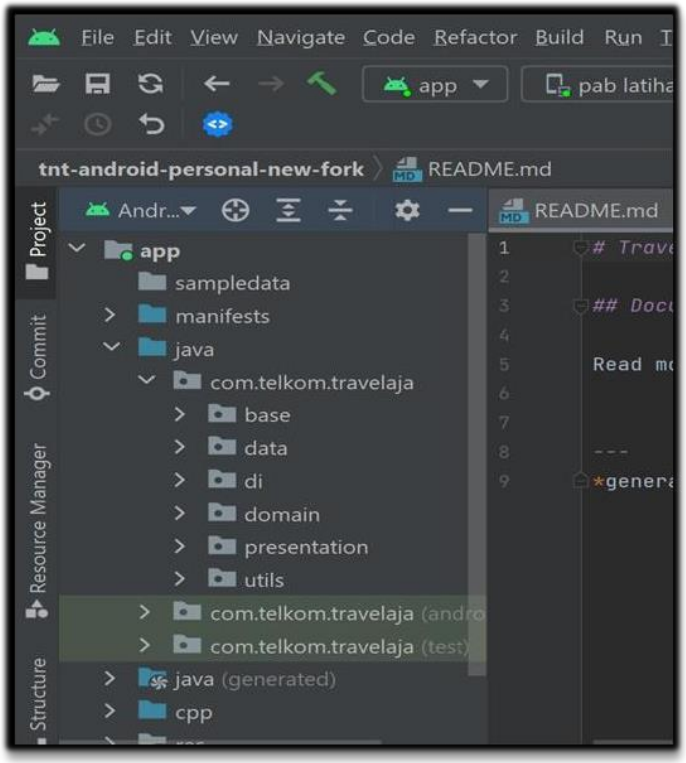
Proyek migrasi adalah proyek pembuatan aplikasi Travel Aja menggunakan Android *native* untuk menggantikan Aplikasi Travel Aja yang sebelumnya sudah dikembangkan menggunakan React *native*.

Alasan pemindahan pengembangan ini dikarenakan isu peforma aplikasi yang dikembangan menggunakan Android *native* lebih baik ketimbang menggunakan React *native*.

Fitur rekomendasi rencana liburan merupakan salah satu fitur yang terdapat pada aplikasi Travel Aja. Fitur ini bertujuan untuk memberikan referensi tempat liburan yang sudah disesuaikan dengan hasil input pengguna. Fitur ini dikembangan dengan menerapkan *Clean Architecture* dan *design pattern* MVVM.

3.3.1 Penerapan Clean Architecture dan design pattern MVVM pada Pengembangan Fitur Rekomendasi Rencana Liburan

Pada pengembangan aplikasi Travel Aja, penerapan *Clean architecture* dilakukan dengan membagi proyek menjadi tiga lapisan yang direpresentasikan dengan *package*. Lapisan-lapisan tersebut yaitu *data package* (*entities layer*), *domain package* (*use case layer*), dan *presentation* (*interface adapter layer*) yang dapat dilihat pada Penerapan *design pattern* MVVM dilakukan dengan membagi kode menjadi tiga komponen, yaitu *Model*, *View*, dan *ViewModel*.



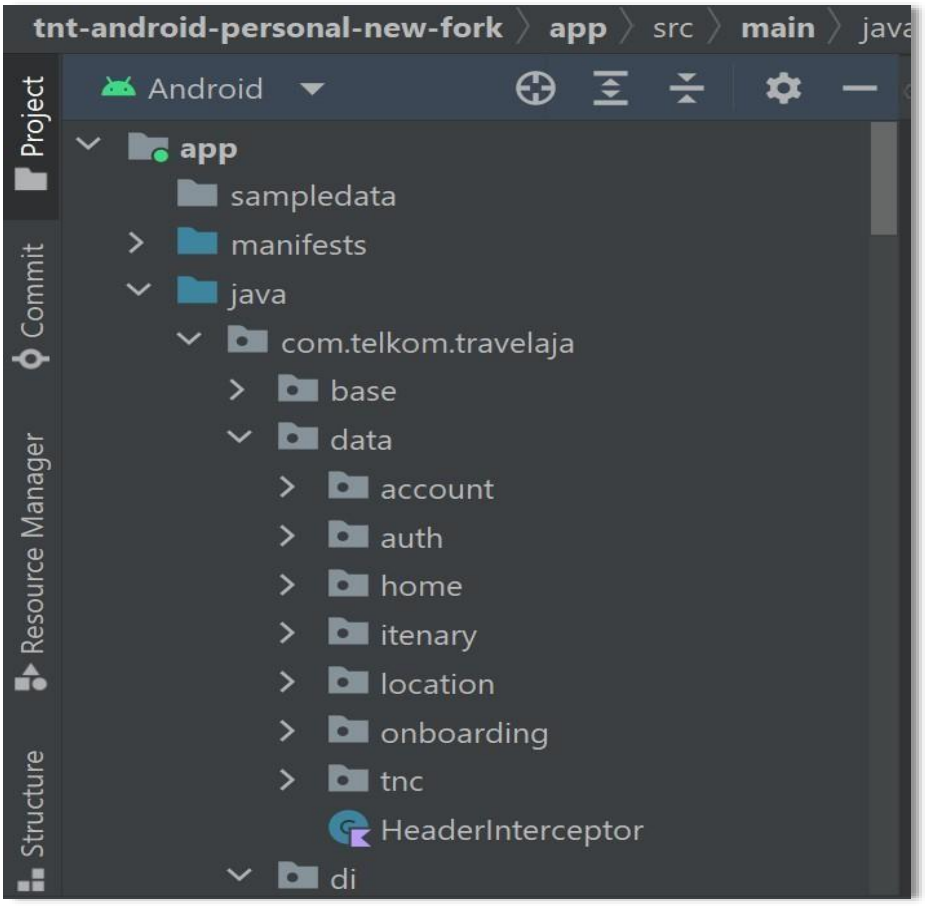
Gambar 3.6 Pembagian *Package Data*, *Domain*, dan *Presentation*

Komponen *Model* yang berkaitan dengan data dan logika bisnis akan ditempatkan pada data *package*. *Data package* merupakan representasi dari *entities layer* pada *Clean Architecture*. *Data package* menangani proses bisnis perusahaan serta pertukaran data melalui *web service*. Domain *package* merupakan representasi dari *use case layer* yang mengenkapsulasi proses bisnis yang terjadi pada *data package*. *Domain pakcage* meneruskan request data dari *presentation package* menuju data *package*. *Presentation layer* merupakan lapisan yang bertanggung jawab menampilkan data yang telah diterima dari *data package* melalui *domain package*. Presentation

package merupakan representasi dari *interface adapter layer* yang mana pada lapisan ini terdapat komponen *View* dan *ViewModel.*

A. Data Layer

Lapisan ini pada dasarnya menangani semua sumber data, mulai dari mendapatkan data dari layanan web atau menyimpannya ke lokal database, retrieving *data from web service(API) using Retrofit, Shared Preference using secured preference* dan *view model*. Di dalamnya, penulis dapat menyimpan layanan sebanyak yang penulis butuhkan dan penulis dapat memberi nama layanan itu seperti yang di inginkan, tetapi selalu mewakili fungsinya.



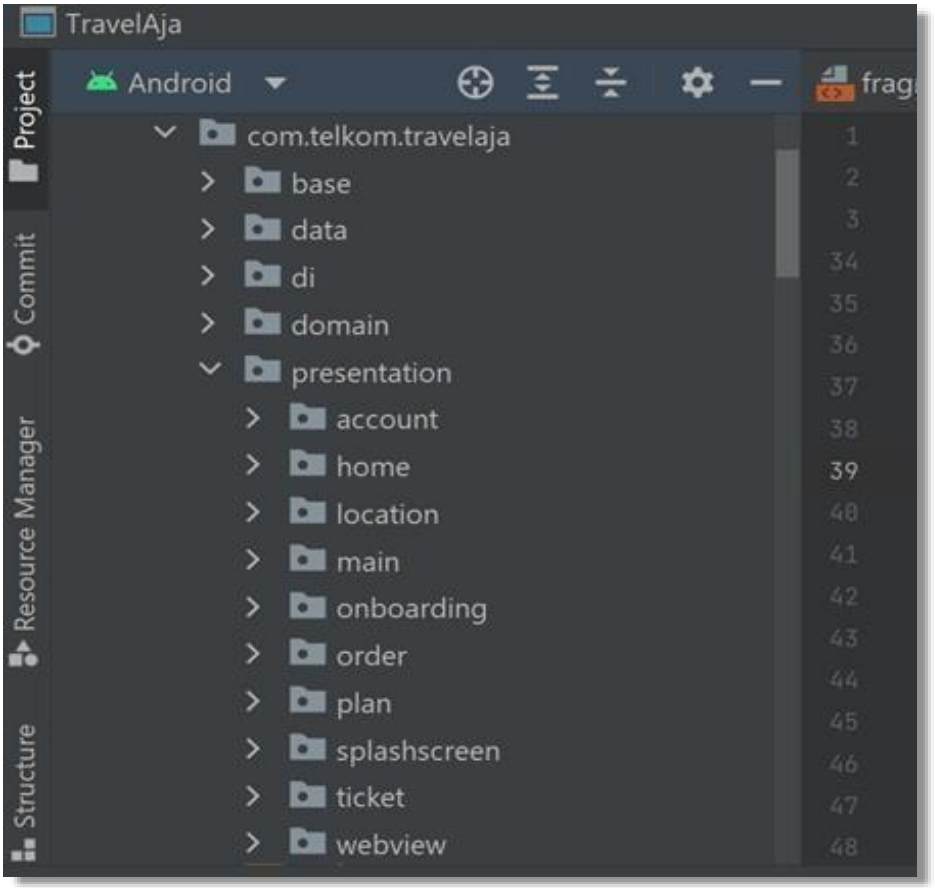
Gambar 3.7 *Package Data*

Data package merupakan *layer* yang mengangani proses pengambilan data baik dari *local* maupun dari *network*. Lapisan ini berisikan kelas dan fungsi-fungsi yang merupakan proses bisnis suatu perusahaan. merupakan kelas diagram pada data *package*. Pada lapisan ini terdapat *Interface ItenaryApiClient* dan kelas *ItenaryApi* yang mengimplementasikan *interface* tersebut.

B. Presentation

Pada modul atau lapisan *presentation* ini terjadi konfirgurasi untuk tampilan yang dibutuhkan. Mulai dari *class activity, fragment, recyclerView Adapter* and *ViewHolder* dan

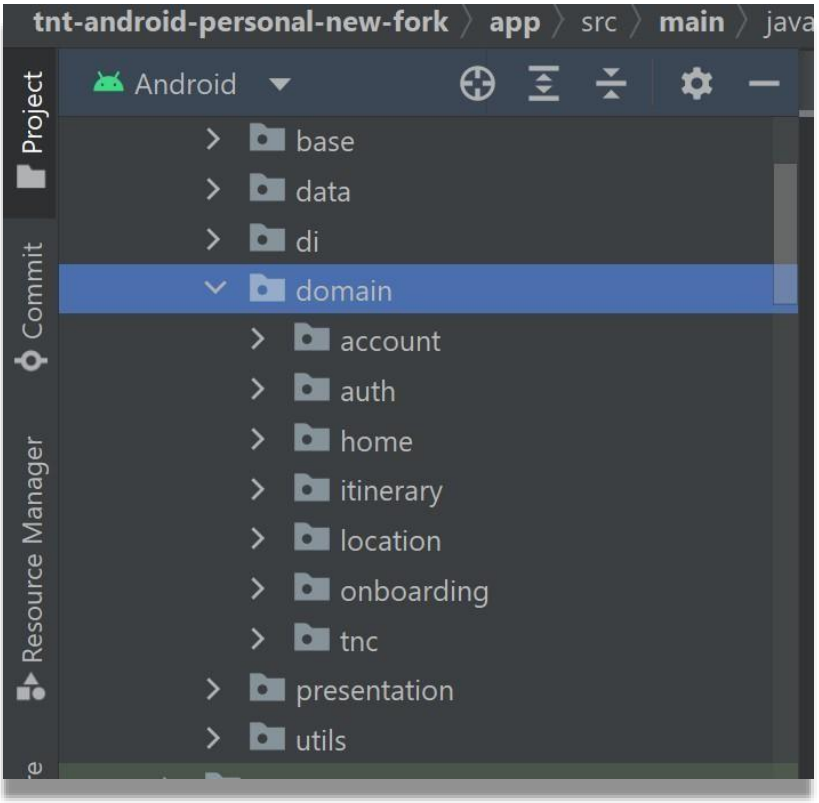
hal lainnya yang berkaitan dengan tampilan. Di dalam lapisan ini bertanggung jawab untuk menampilkan antarmuka pengguna kepada pengguna akhir. *View* berisi elemen UI seperti tombol, input pengguna, dan tampilan *data*. View hanya berfokus pada tampilan data dan tidak memiliki logika bisnis.



Gambar 3.8 *Package Presentation*

C. Domain

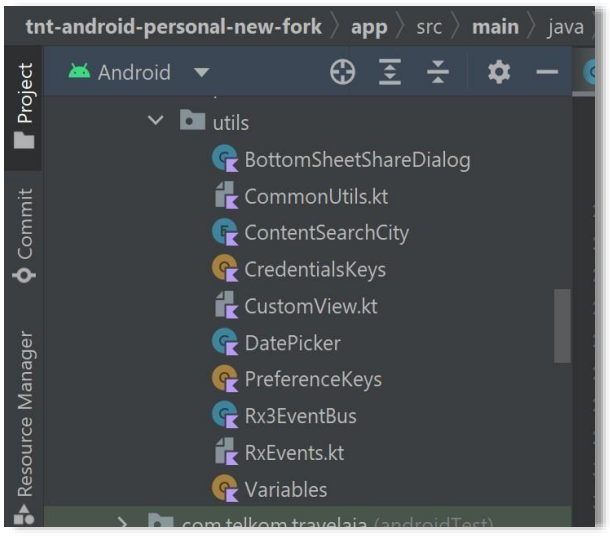
Pada modul ini menentukan konfigurasi untuk fitur layanan seluler (GMS atau HMS). Itu termasuk kelas untuk Menangani peristiwa khusus *Analytics*, properti pengguna, dan *userId*, Menangani otentikasi menggunakan Google, Twitter, Microsoft, Github, dan Yahoo. Kemudian, pada modul ini juga berkaitan dengan kelas layanan dasar untuk *cloud messaging* menangani nilai konfigurasi jarak jauh serta menangani tautan dinamis menangani pendengar untuk pesan dalam aplikasi.



Gambar 3.9 *Package Domain*

D. Utils

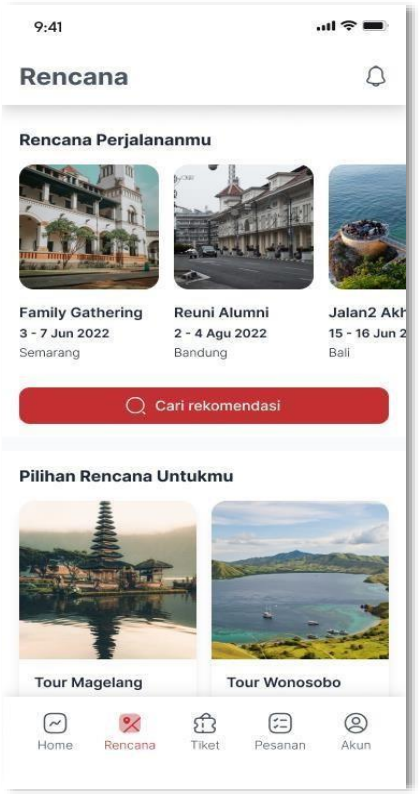
Pada modul ini mendefinisikan fungsi global dan utilitas untuk aplikasi. ini mendefinisikan penangan kesalahan khusus fungsi ekstensi untuk tanggal dan *imageview*. Lalu, modul ini berfungsi sebagai metode global untuk hal-hal seperti menampilkan dialog, dan keamanan verifikasi sms kelas aplikasi dasar.



Gambar 3.10 *Package Utils*

3.4 Hasil Pembuatan Fitur Rekomendasi Rencana Liburan dengan menerapkan Design Pattern MVVM dan Clean Architecture

Penulis bertanggung jawab untuk menyelesaikan fitur rekomendasi rencana liburan, fitur ini memiliki beberapa page yang saling berkaitan atau berhubungan satu sama lainnya. Dengan menerapkan *architecture Model View View Model* (MVVM). Dimana penulisa memisahkan antara tampilan dan logika bisnis.



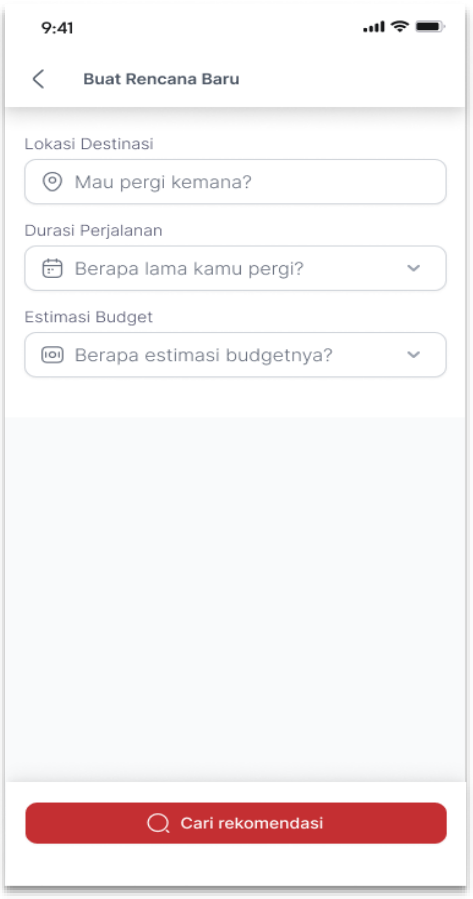
Gambar 3.11 *Page* depan *rencana liburan*

Dengan menerapkan pola design ini, penulis memiliki keuntungan dam kemudahan dalam pengembangan aplikasi Travel Aja. Kemudahan pertama, pengujian (*testing*) karena komponen-komponen dapat diuji secara terpisah, dan memungkinkan pengembang UI (*User Interface*) dan pengembang logika bisnis untuk bekerja secara independen satu sama lain. Gambar 3.11 merupakan page awal ketika user ingin menggunakan fitur rekomendasi rencana liburan. Di *page* ini terdapat beberapa button untuk pindah ke page lainnya. Di page ini terdapat *button* berwarna merah bertuliskan cari rekomendasi ketika *user* klik *button* tersebut akan menuju *page* buat rencana liburan seperti pada Gambar 3.12 yang memiliki beberapa isian atau field yang perlu diisi oleh user.

Dalam pengembangan fitur rekomendasi rencana liburan, penulis membuat sebuah *package* bernama *Presentation package* yang berada pada lapisan *interface adapter layer* bertanggung jawab untuk melakukan pembaharuan tam pilan berdasarkan *data* yang telah diperoleh dari *viewmodel*.

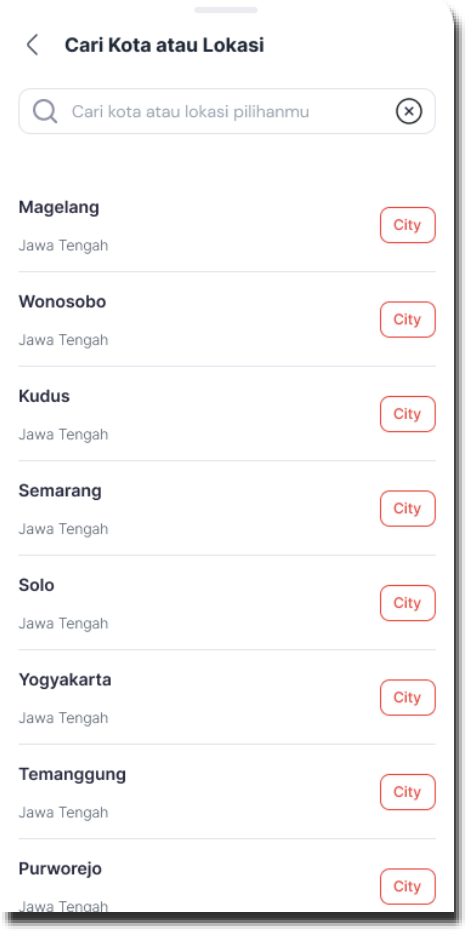
Kemudian, pada *presentation package* terdapat komponen tampilan, yaitu kelas *PlanFragment* yang mengatur tampilan pada layar smartphone. Pada lapisan ini juga diterapkannya *design pattern* MVVM, dimana kelas *PlanViewModel* yang mengimplementasikan kelas abstrak *ViewModel b*ertugas untuk menghubungkan komponen *view* (*presentation package*) dengan komponen *model* (*data package*) melalui bantuan use *case layer* (*domain package*).

Domain package bertindak sebagai mediator atau penghubung antara *presentation package* dan *data package*. Kelas *PlanViewModel* pada presentation *package* akan melakukan *request* melalui *domain package* menggunakan *interface HomeUseCase* dan kemudian *domain package* meneruskan request tersebut ke *data package* menggunakan *interface PlanRepository*.



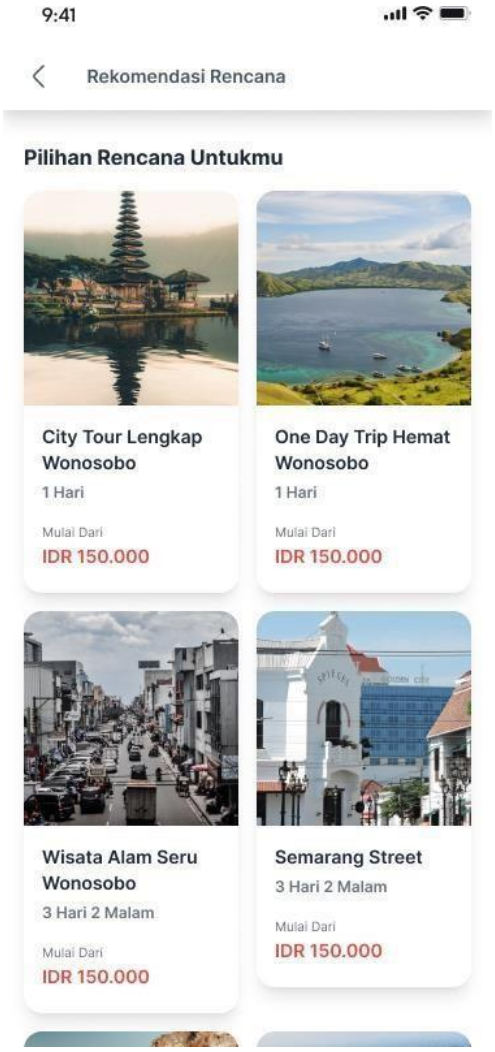
Gambar 3.12 *Page buat rencana baru*

Data package akan melakukan request *data* ke *server* melalui *REST API* dengan menggunakan kelas *PlanApi*, kemudian mengembalikan data ke *domain package* dan diteruskan ke *presentation package*. Data package ini berperan penting dalam menampilkan data yang diinginkan hasil dari pencarian lokasi di Gambar 3.13 data akan diteruskan ke layer presentation page buat rencana baru pada Gambar 3.12



Gambar 3.13 *Page Seacrh Location*

Pada pengembangan *page search location* di Gambar 3.13 *user* mencari lokasi sesuai dengan yang diinginkan. Kemudian, user memilih salah satu dari pencarian tersebut. Setelah user memilih lokasi yang dituju, page akan beralih ke Gambar 3.12 setelah field-field terisi *Data* tersebut akan disimpan oleh *variable InboxList* yang merupakan sebuah *LiveData*. *PlanFragment* di *presentation package* mendapatkan data melalui *PlanViewModel* dengan menggunakan interface *PlanUseCase* yang ada pada *domain package*. Sementara itu *domain package* mendapatkan *data* dari *data package* dengan menggunakan *interface PlanRepository*. Data yang disimpan oleh *variable inboxList* akan diobservasi oleh *PlanFragment* untuk melakukan pembaharuan tampilan berdasarkan data yang disimpan.



Gambar 3.14 *Page hasil pencarian rekomendasi liburan*

Pada fitur *search* lokasi liburan sesuai dengan Gambar 3.13 berguna untuk melakukan pencarian lokasi liburan berdasarkan nama lokasi, sedangkan fitur rekomendasi dapat digunakan untuk melakukan pencarian lokasi berdasarkan kategori *bugdet* dan lama durasi liburan. Hasil pencarian rekomendasi rencana liburan dapat dilihat di Gambar 3.14 yang menampilkan gambar lokasi liburan dan detail *budget* liburan. Fitur-fitur tersebut dikembangkan dengan memanfaatkan *design pattern* MVVM dan *Clean Architecture*. Jika terdapat penambahan fitur di fitur rekomendasi, programmer hanya perlu menambahkan kode padapackage *data*, *domain*, dan *presentation* yang berkaitan dengan fitur rekomendasi rencana liburan, sehingga tidak mengganggu kode fitur yang lainnya.

3.5 Pengujian aplikasi Travel Saja

Pengujian perangkat lunak adalah proses yang dilakukan untuk mengidentifikasi dan mengevaluasi kualitas perangkat lunak yang dikembangkan. Tujuan dari pengujian perangkat lunak adalah untuk memastikan bahwa perangkat lunak tersebut berfungsi dengan baik, sesuai

dengan kebutuhan pengguna, dan bebas dari bug atau kesalahan. Dalam pengujian perangkat lunak, berbagai metode dan teknik digunakan untuk menguji fungsionalitas, kehandalan, kinerja, keamanan, dan aspek-aspek lain dari perangkat lunak. Tes dilakukan dengan memberikan input yang beragam ke dalam perangkat lunak, dan mengamati output yang dihasilkan serta membandingkannya dengan hasil yang diharapkan.

3.5.1 Pengujian Black Box Testing

Black box testing adalah metode pengujian perangkat lunak yang melibatkan pengujian fungsionalitas sistem dari perspektif pengguna akhir. Tester tidak memperhatikan implementasi internal sistem, tetapi fokus pada pengujian input dan output untuk memverifikasi kinerja dan kepatuhan sistem terhadap spesifikasi yang telah ditentukan. Hasil pengujian *black box testing* dapat dilihat pada Tabel 3.1

Tabel 3.1 Pengujian Black Box Testing

No	Aktivitas Pengujian	Realisasi yang diharapkan	Hasil Pengujian	Kesimpulan
1	Halaman Awal	Pengguna dapat memilih bahasa	Tampil halaman pilih bahasa	Berhasil
2	Klik tombol “Registrasi”	Menampilkan pengguna dapat registrasi akun	Menampilkan halaman registrasi	Berhasil
3	Klik tombol “*Login*”	pengguna dapat login ke halaman *dashboard*	Fitur *Login* dapat Berfungsi dengan baik	Berhasil
4	Klik tombol “masuk dengan google”	pengguna dapat login ke halaman *dashboard*	Fitur *Login* dengan google dapat berfungsi dengan baik	Berhasil
5	Klik tulisan “masuk tanpa akun”	Menampilkan halaman *dashboard*	Menampilkan Halaman dashboard	Berhasil
6	Klik tombol “Cari Rekomendasi”	Menampilkan halaman rekomendasi liburan	pengguna berhasil masuk ke halaman rekomendasi liburan	Berhasil
7	Klik field Lokasi Destinasi”	Menampilkan halaman memilih lokasi liburan	pengguna berhasil memilih lokasi liburan	Berhasil

		halaman pencarian lokasi liburan	pencarian lokasi liburan dan berhasil memilih lokasi	
8	Klik field"durasi lburan"	Pengguna dapat memlih durasi liburan	Pop up memilih durasi liburan muncul	Berhasil
9	Klik field"*budget lburan*"	Pengguna dapat memlih *budget* liburan	Pop up memilih *budget* liburan muncul	Berhasil
10	Klik tombol "Cari Rekomendasi"	Pengguna dapat memlih rekomendasi liburan sesuai *field* yang diisi	Pengguna berhasil masuk ke halaman rekomendasi liburan dan berhasil memilih lokasi liburan	Berhasil

BAB IV
HASIL DAN PEMBAHASAN

4.1 *Relevansi Akademik*

Aplikasi Travel Aja yang berbasis Android dalam pengembangannya menerapkan *Clean Architecture* dan *design pattern Model View View Model* (MVVM). Pada pelaksanaannya tidak terdapat perbedaan antara teori Android *Clean Architecture* dan penerapannya di pengembangan aplikasi Travel Aja. Teori-teori yang dijelaskan diterapkan dengan baik pada pengembangan aplikasi Travel Aja. Aplikasi Travel Aja dibagi menjadi tiga lapisan, yaitu data *package* (*entities layer*), *domain package* (*use case layer*), dan *presentation package* (*interface adapter layer*) yang merepresentasikan *layer* pada *Clean Architecture*.

Penerapan *Model View View Model* (MVVM) pada pengembangan aplikasi Travel Aja juga tidak terdapat perbedaan dengan teori yang dijelaskan. Pada kelas *ViewModel* menerapkan *live* data sebagai *observable data holder* yang dapat diobservasi oleh komponen *View*. Kelas *ViewModel* juga mengamati dan mengkoordinasikan pembaruan dari data package, kemudian diteruskan ke domain package dan menyimpan hasil pembaruan *Model* ke dalam variable *LiveData*. Kemudian, *variable LiveData* tersebut akan digunakan oleh komponen *View* untuk memperbaharui tampilan antarmuka aplikasi.

Pengembangan aplikasi di Travel Aja menerapkan kerangka kerja SCRUM. *Scrum Team* di proyek Travel Aja terdiri dari *scrum master*, developer, dan *product owner*. Pengembangan dimulai dengan *sprint planning*, dimana *product owner* akan memberikan *product backlog item* dan kemudian akan dibahas bersama developer untuk menentukan *backlog item* apa saja yang akan dibawa ke tahap *sprint*. Kemudian, *sprint* akan dilaksanakan selama dua minggu. Setelah *sprint* selesai dilakukan, akan dilakukan *sprint review* untuk melihat hasil pengerjaan *backlog item*. Acara terakhir yang dilakukan adalah *sprint retrospective* yang bertujuan untuk mengevaluasi kekurangan dari *sprint* yang telah selesai agar *sprint* selanjutnya menjadi lebih baik. Proses-proses *scrum* yang dilakukan di Travel Aja juga telah sesuai dengan teori yang dijelaskan pada Bab II Pembelajaran Magang.

4.2 Pembelajaran Magang

Setelah menjalani magang selama 6 bulan, penulis mendapatkan banyak pengalaman pada saat bekerja. Pengalaman-pengalaman tersebut belum pernah penulis dapatkan sebelumnya. Pada saat magang, penulis dihadapkan dengan masalah yang sesungguhnya dan harus dapat

menyelesaikan masalah tersebut dengan mengaplikasikan ilmu yang sudah dipelajari selama kuliah. Pembelajaran yang penulis dapatkan ketika menjalani magang dapat dikategorikan menjadi pembelajaran teknis dan non-teknis.

A. Teknis

Sebagai seorang Android programmer, penulis ditugaskan di dua proyek dan mengerjakan berbagai macam fitur. Banyak ilmu yang didapatkan ketika mengerjakan proyek tersebut. Rekan-rekan programmer lainnya juga berbagi ilmu tentang best practice pada saat melakukan coding. Ilmu-ilmu yang diberikan menjadi bekal untuk penulis agar bisa mengupgrade skill menjadi lebih baik. Berikut adalah pembelajaran teknis yang penulis dapatkan ketika melaksanakan magang.

a. Model View ViewModel

Model View View model merupakan salah satu deign pattern yang digunakan pada saat mengembangkan aplikasi Android. *design pattern* ini menjadi penghubung antara *ui layer* dan *data layer*. Selain itu, MVVM juga memisahkan *ui logic* dan *business logic* sehingga *ui* dapat dengan mudah diubah. Selama magang penulis menggunakan *design pattern* MVVM untuk mengembangan fitur Rekomendasi Rencana Liburan. Design pattern ini sangat membantu dalam membuat kode menjadi lebih bersih karena memisahkan *ui logic* dan business *logic*. Sehingga sangat memudahkan penulis untuk melakukan pembaharuan UI atau penambahan fitur seperti fitur *Search* dan fitur Filter.

b. Android *Clean Architecture*

Clean Architecture bertujuan untuk memisahkan kepentingan dengan membagi aplikasi menjadi beberapa *layer*. Dalam proyek magang yang penulis kerjakan, aplikasi dibagi menjadi tiga lapisan, yaitu *data package* (*entities layer*), domain package (*use case layer*), dan *presentation package* (*interface adapter layer*). *Data package* berfokus kepada proses pengambilan dan pengiriman data, presentation package berfokus kepada tampilan *User Interface*, dan *domain package* berfungsi sebagai pusat komunikasi antara *presentation package* dan *data package*.

Dalam pengembangan aplikasi Travel Aja diterapkan *Clean Archicture* agar kualitas kode lebih baik dan mudah dipahami oleh programmer yang lain. Dikarenakan banyak programmer yang mengerjakan proyek tersebut. Bagi penulis *Clean Architecture* sangat membantu penulis untuk memahami *flow* dari kode-kode yang ada. Karena *Clean Architecture* membuat kode-kode menjadi lebih terstruktur dan mudah dipahami. *Clean*

Architecture juga mempermudah pengembangan karena programmer tidak akan mengubah kode yang tidak termasuk dalam fitur tersebut, sehingga memperkecil kemungkinan terjadinya konflik pada saat pengembangan.

c. Git

Git merupakan *software version control system* yang berfungsi untuk mencatat semua perubahan dalam suatu proyek. Aplikasi Travel Aja memiliki banyak progammer dan fitur sehingga diperlukan sebuah *version control system* untuk mengatur *flow* pengerjaan sebuah fitur agar tidak terjadi konflik. Penulis mempelajari bagaimana menerapkan sebuah *Git flow* yang baik untuk menghindari konflik antar programmer. Selain itu penulis juga belajar cara meng-*cloning* project dari *repostiory*, serta belajar melakukan *push, pull*, dan *merging* antar *branch*.

d. Manfaat Migrasi Teknologi React Native ke Kotlin Native

Migrasi teknologi dari React Native ke Kotlin Native memberikan sejumlah manfaat yang dirasakan sebagai berikut.

1. Kotlin *native* memberikan performa yang lebih baik dalam pengembangan aplikasi. Dengan mengompilasi kode menjadi kode mesin langsung, Kotlin *native* menghilangkan *overhead* yang terkait dengan jembatan komunikasi, sehingga meningkatkan kinerja aplikasi.
2. Kotlin *native* memungkinkan akses langsung ke *api-platform* dan sumber daya perangkat. Ini berarti pengembang dapat menggunakan fitur dan kemampuan perangkat secara langsung, meningkatkan fleksibilitas dan kualitas pengembangan aplikasi.
3. Migrasi ke Kotlin *native* memungkinkan pengembangan *modular* yang lebih baik.
4. Dengan sintaks yang bersih dan berbasis objek, Kotlin meningkatkan produktivitas pengembang dan memungkinkan penulisan kode yang lebih mudah dibaca dan dipelihara.

Terakhir, migrasi ke Kotlin *native* memungkinkan akses ke ekosistem Android yang luas. Dengan dukungan penuh untuk perpustakaan dan *framework* Android yang ada, pengembang dapat memanfaatkan sumber daya yang tersedia dan mengintegrasikan aplikasi dengan komponen Android lainnya.

e. Manfaat Penerapan *Clean Architecture* dan *design pattern* MVVM

Manfaat yang didapatkan dari penerapan *Clean Architecture* dan *design pattern* MVVM adalah sebagai berikut.

1. *Design pattern* MVVM memisahkan antara logika tampilan dan logika bisnis. Sehingga memudahkan pengembangan secara *paralel*. Pengembangan kelas tampilan antarmuka dapat dilakukan bersamaan dengan pengembangan kelas data oleh developer yang berbeda.
2. Developer dapat dengan mudah untuk menambahkan fitur baru.
3. Pemisahan kosentrasi pada kode membuat kemungkinan konflik pada saat pengembangan menjadi kecil. Hal tersebut dikarenakan developer tidak akan menggangu kode yang tidak berkaitan dengan *Backlog Item*-nya. Jika developer melakukan perubahan desain antarmuka, maka developer hanya perlu mengubah kelas *Activity* atau *Fragment* yang ada pada *presentation package*

B. Non-Teknis

Pembelajaran non-teknis yang penulis rasakan secara pribadi selama melaksanakan magang di Travel Aja adalah sebagai berikut.

a. Komunikasi

Selama magang penulis dituntut bekerja di bawah tekanan dan harus bisa bekerja sama dengan baik dalam sebuah tim. Komunikasi menjadi hal yang penting pada saat bekerja sama dalam sebuah tim. Jika komunikasi berjalan dengan lancar, maka tugas-tugas yang ada dapat diselesaikan dengan baik pula dan memperkecil kesalahan karena miskomunikasi. Komunikasi di dalam tim biasa dilakukan melalui Whatsapp dan Google Meet.

b. Manajemen Waktu

Setiap fitur yang dikembangkan dilakukan dalam satu Sprint yang berdurasi 10 hari kerja, sehingga penulis harus bisa memanajemen waktu dengan baik agar fitur dapat diselesaikan dengan tepat waktu. Karena hal tersebut, skill manajemen waktu menjadi salah satu skill yang penting. Manajemen waktu dilakukan sesuai dengan prioritas pekerjaan dan hard skill yang di miliki.

c. Memperluas Industri

Selama magang penulis juga mendapat banyak relasi baru dengan rekan-rekan programmer lain yang lebih senior. Tidak hanya rekan programmer, penulis juga

mendapat relasi dengan *quality assurance*, *scrum master*, *back-end engineer*, dan *project manager*. Relas-relasi tersebut bermanfaat untuk perkembangan karir penulis kedepannya.

d. Megenal Industri Teknologi

Sebelumnya penulis tidak mengetahui bagaimana sebuah industri teknologi bekerja dan seperti apa budaya kerja di dalam industri teknologi. Setelah melaksanakan magang barulah penulis mengetahui bagaimana industri teknologi bekerja. Industri teknologi bekerja dengan banyak orang dan tim di dalamnya. Setiap tim berkolaborasi dengan tim lainnya dalam bekerja untuk mencapai *product goal* yang telah ditetapkan.

4.2.1 Hambatan dan Tantangan Magang

Kendala adalah sebuah faktor yang mencegah tercapainya suatu target (KBBI Daring, 2016). Terdapat beberapa aspek yang menjadi kendala penulis yaitu keadaan yang jauh dari kantor sehingga terkadang kesulitan berkomunikasi ketika mendapatkan kendala teknis terkait coding. Komunikasi hanya mengandal gawai dan *via* aplikasi Whatsapp. Kedua software tersebut memerlukan internet sehingga apabila terjadi mati listrik atau gangguan internet membuat komunikasi menjadi terkendala dan dapat berakibat menghambat pekerjaan.

Tantangan yang harus penulis hadapi adalah mengatur atau memanajemen waktu. Hal tersebut dikarenakan penulis memiliki kelas kuliah serta tugas magang. Penulis melakukan manajemen waktu berdasarkan prioritas. Jika suatu tugas memiliki prioritas yang tinggi, maka penulis akan mendahulukan tugas tersebut, kemudian mengerjakan tugas yang prioritasnya lebih rendah.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil migrasi menggunakan Kotlin *native*, dapat disimpulkan bahwa penggunaan Kotlin *native* dalam pengembangan aplikasi memiliki beberapa keunggulan. Pertama, React *native* yang merupakan cross-platform dari segi performa lebih rendah dibandingkan Kotlin *native* memberikan performa yang lebih baik. Kode Kotlin *native* dikompilasi menjadi kode mesin langsung, yang menghasilkan kinerja yang lebih baik tanpa memerlukan jembatan komunikasi. Dari penerapan design pattern *Model View ViewModel* (MVVM) dan *clean architecture* di proyek migrasi aplikasi.

Travel Aja, terdapat beberapa hal yang dapat disimpulkan, yaitu sebagai berikut.

a. Di dalam penggunaan Android Studio diimplementasikan dengan membagi proyek menjadi tiga *layer*, yaitu *entities layer* (*data package*), *use case layer* (*domain package*), dan *interface adapter layer* (*presentation package*).
b. Design pattern *Model View ViewModel* dapat diterapkan bersamaan dengan *Clean Architecture*. Penerapan *design pattern Model View ViewModel* (MVVM) ini dilakukan pada *interface adapter layer* (*presentation package*) dengan membuat kelas yang mengimplementasikan kelas abstrak *viewmodel*.
c. Supaya penerapan *Clean Architecture* berjalan sukses, pengembang harus mematuhi prinsip *Dependency Inversion Principle*.
d. Memudahkan pembagian *backlog item* kepada developer dan memperjelas tanggung jawab developer terkait *backlog item* yang dikerjakan.
e. Membuat code menjadi lebih mudah dipahami dan dicerna oleh developer baru. Hal ini dikarenakan alur kode yang jelas, mulai dari proses pengambilan data sampai dengan menampilkan data.

Migrasi ke Kotlin *native* juga memberikan fleksibilitas dalam pengembangan. Pengembang dapat menggunakan bahasa Kotlin yang modern dan ekspresif untuk mengembangkan aplikasi yang lebih mudah dipelihara dan ditingkatkan di masa depan. Selain itu, Kotlin *native* juga mendukung penggunaan perpustakaan dan *framework* Android yang sudah ada, sehingga memungkinkan pengembang untuk memanfaatkan ekosistem Android yang luas.

Secara keseluruhan, migrasi pengembangan aplikasi Travel Aja dari React *native* ke Kotlin *native* merupakan langkah yang efektif untuk meningkatkan performa, mengakses fitur perangkat secara langsung, dan memungkinkan pengembangan *modular*. Dengan memperhatikan tantangan yang mungkin muncul dan mempersiapkan tim dengan baik, migrasi ini dapat memberikan hasil yang positif bagi pengembangan aplikasi Travel Aja serta meningkatkan daya saing perusahaan.

5.2 Saran

Setelah melaksanakan program magang di PT. Telkom Indonesia selama enam bulan, terdapat beberapa saran yang dicatat dengan target sasaran sebagai berikut:

a. PT. Telkom Indonesia

Penulis menjumpai beberapa kesulitan yang dialami oleh mahasiswa ketika hendak memilih program magang termasuk penulis. Setelah diteliti bahwa sebagian besar dari mahasiswa kesulitan informasi mengenai magang, persyaratan apply, cara apply dan lain sebagainya. Oleh karena hal tersebut, saran dari penulis bahwa akan lebih baik apabila prodi menyediakan solusi dalam satu wadah seperti sub aplikasi yang berada di gateway. Harapannya, pada wadah tersebut mahasiswa dapat melihat informasi perusahaan apa saja yang sedang menerima mahasiswa magang, apa saja persyaratannya, dan sekaligus dapat melakukan apply. Kemudian, penulis mengalami kendala untuk mendapatkan referensi atau tutorial terkait penggunaan *dev-codebase* untuk developer baru dxi PT.Telkom Indonesia. Diharapkan dapat sering mendokumentasikan materi-materi terkait di sosial media.

b. Migrasi Aplikasi Travel Aja

Dengan menerapkan *Clean Architecture*, pengembang dapat mencapai aplikasi yang terstruktur, teruji, mudah dikembangkan, dan mudah dipelihara. *Clean Architecture* membantu dalam mencapai tujuan ini dengan pemisahan tanggung jawab, ketergantungan terbalik, skalabilitas, keterbacaan, kemandirian teknologi, serta kemudahan dalam pengujian dan pemeliharaan aplikasi. Hanya saja pada penerapannya di proyek Travel Aja tidak selalu menerapkan *Unit Test*, sehingga belum memenuhi satu tujuan dari *Clean Architecture*, yaitu Testable. Tujuan tersebut dapat dipenuhi dengan menerapkan *Unit Test* dan *Instrument Test* pada proyek aplikasi Travel Aja.

c. Mahasiswa Magang

Adanya kesenjangan antara kebutuhan perusahaan dan kemampuan mahasiswa sudah menjadi hal umum. Oleh karena hal tersebut, saran dari penulis bahwa mahasiswa agar membiasakan diri, berlatih, pada hal-hal baru dan tidak terpaku pada teori-teori dari perkuliahan. Di perusahaan Telkom Indonesia, mahasiswa magang memiliki kesempatan luas untuk mempraktikkan teori-teori tersebut dan terlibat aktif di dalam proyek yang sedang dikembangkan. Hal itu menjadi manfaat positif bagi mahasiswa

DAFTAR PUSTAKA

Aldiansyah, M., Kharisma, A., … I. A.-T. I. dan, & 2021, undefined. (2021a). Pengembangan Aplikasi Ngobat: Aplikasi Ketaatan Regimen Pengobatan menggunakan Gamification pada Platform Android. *J-Ptiik.Ub.Ac.Id*, *5*(8), 3600–3608. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/9637

Aldiansyah, M., Kharisma, A., … I. A.-T. I. dan, & 2021, undefined. (2021b). Pengembangan Aplikasi Ngobat: Aplikasi Ketaatan Regimen Pengobatan menggunakan Gamification pada Platform Android. *J-Ptiik.Ub.Ac.Id*, *5*(8), 3600–3608. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/9637

AUTOMATA, R. G.-, & 2021, undefined. (2021). Implementasi Scrum Pada Manajemen Proyek Pengembangan Aplikasi Sistem Monitoring dan Evaluasi Pembangunan (SMEP). *Journal.Uii.Ac.Id*, *1*. https://journal.uii.ac.id/AUTOMATA/article/view/17420

Avinda, C., Sudiarta, I., ISSN, N. K.-J. I., & 2016, undefined. (2016). Strategi promosi Banyuwangi sebagai destinasi wisata (studi kasus pada Dinas Kebudayaan dan Pariwisata). *Ojs.Unud.Ac.Id*, *4*(1). https://ojs.unud.ac.id/index.php/pariwisata/article/download/22527/14802

Bui, H. (2020). *ANDROID APPLICATION FOR STUDENTS'PERSONAL FINANCES.* https://www.theseus.fi/bitstream/handle/10024/339855/Bui_Huy.pdf?sequence=2

Duy, T. B. (2017). *Reactive programming and clean architecture in Android development*. https://www.theseus.fi/bitstream/handle/10024/126982/Thesis2016-Sunshine.pdf

Fajri, A. (2022). *Penerapan Design Pattern Mvvm Dan Clean Architecture Pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree)*. https://dspace.uii.ac.id/handle/123456789/40624

Kurniawan, H., Apriliah, W., … I. K.-J. I., & 2020, undefined. (n.d.). Penerapan Metode Waterfall Dalam Perancangan Sistem Informasi Penggajian Pada Smk Bina Karya Karawang. *E-Journal.Rosma.Ac.Id*. https://doi.org/10.35969/interkom.v14i4.58

Makarenko, V., Olshevska, O., & Kornienko, Yu. (2017). AN ARCHITECTURAL APPROACH FOR QUALITY IMPROVING OF ANDROID APPLICATIONS DEVELOPMENT WHICH IMPLEMENTED TO COMMUNICATION APPLICATION FOR MECHATRONICS ROBOT LABORATORY ONAFT. *Automation of Technological and Business Processes*, *9*(3). https://doi.org/10.15673/ATBP.V9I3.714

Nugroho, A. (2010). *Rekayasa perangkat lunak berorientasi objek dengan metode USDP*. https://books.google.com/books?hl=en&lr=&id=CB0IKsa9cNEC&oi=fnd&pg=PA267&dq=Entities+adalah+komponen+dalam+arsitektur+perangkat+lunak+yang+merepresentasikan+objek-objek+penting+dalam+sistem.+Mereka+menggambarkan+data+dan+perilaku+yang+terkait+dengan+domain+bisnis+aplikasi.+Entities+biasanya+merupakan+bagian+inti+dari+sistem,+dan+t&ots=DV_qPS1gUP&sig=22FaCUCuIGhYOJruxuTswkmutQg

Oleh, D., & Rahman, A. (2022). *Penerapan Design Pattern Mvvm Dan Clean Architecture Pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree)*. https://dspace.uii.ac.id/handle/123456789/40624

Putra, I., Kharisma, A., … I. A.-T. I. dan I., & 2021, undefined. (n.d.). Pengembangan Aplikasi Berbasis Android Untuk Meningkatkan Kepatuhan Pasien Gagal Jantung Dalam Merawat Diri Di Rumah. *Jptiik.Multi.Web.Id*. Retrieved July 3, 2023, from https://jptiik.multi.web.id/index.php/j-ptiik/article/view/9463

Rama, B. (2020). *Tata Kelola Destinasi Wisata: dan Peraturan Perundangan Pariwisata*. https://books.google.com/books?hl=en&lr=&id=hqH6DwAAQBAJ&oi=fnd&pg=PR3&dq=Walaupun+demikian,+pemanfaatan+teknologi+pada+bidang+wisata+masih+relatif+sedikit,+padahal+masyarakat+membutuhkan+bantuan+teknologi+untuk+mendapatkan+pengalaman+liburan+yang+lebih+terencana+dan+informasi+mengenai+potensi+wisata+yang+ditawarkan+dari+setiap+&ots=bMadPL7-Wv&sig=FQo5lBTMWsDzcpRL-Nwlolnex3c

Rizaldy, R., & Dirgahayu, R. T. (2020). Pengembangan Front-End Sistem Informasi Pendataan Pendar Foundation Yogyakarta. *AUTOMATA*, *1*(2). https://journal.uii.ac.id/AUTOMATA/article/view/15591

Setiawan, E. I., Prakoso, H. K. B., Gunawan, T. P., Setyati, E., & Santoso, J. (2021). Aplikasi Mobile Untuk Memantau Body Mass Index Dengan Metodologi Scrum. *Teknika*, *10*(3), 242–250. https://doi.org/10.34148/teknika.v10i3.405

Sihite, M. (2022). *Aplikasi Mobile Periklanan Digiprom Berbasis Android*. http://digilib.unila.ac.id/id/eprint/66380

Wahyudi, A., Riadi, I., & Dahlan Jl Ahmad Yani Tamanan, A. (2022). PERAN STRATEGIS SCRUM MASTER PADA PENGEMBANGAN PERANGKAT LUNAK

PERPUSTAKAAN SEKOLAH BERBASIS ANDROID. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *7*(3), 711–717. https://doi.org/10.29100/JIPI.V7I3.2994