BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka ini bertujuan untuk memberikan gambaran tentang teori-teori dan hasil-hasil penelitian terdahulu yang relevan dengan topik penelitian. Penelitian terdahulu diperlukan sebagai dasar yang mendukung pelaksanaan penelitian ini. Terdapat beberapa penelitian terdahulu yang telah membahas topik serupa akan dianalisis dan diintegrasikan ke dalam kerangka kerja penelitian ini. Dengan memanfaatkan literatur-literatur tersebut, penelitian ini bertujuan untuk membangun pengetahuan baru, mendalam, dan kontekstual dalam konteks topik penelitian yang bersangkutan.

Pada penelitian [14], melakukan penelitian dengan studi kasus pada Pendar *Foundation* yang mengalami kesulitan dalam proses pencatatan dan pendataan. Proses tersebut masih dilakukan secara manual serta kesulitan untuk melakukan sinkronasi data. Sebuah sistem dikembangkan untuk mengatasi permasalahan tersebut. Penelitian dilakukan secara kolaborasi dengan fokus pada sisi *front end* dengan menggunakan metode *Waterfall* dan UCD. Hasil dari penelitian berupa sebuah sistem informasi berbasis yang memerpudah *staff* dari Pendar *Foundation* untuk melakukan penyimpanan data.

Selanjutnya Pada penelitian [7], pengembangan *front end* berbasis *android* dilakukan menggunakan metode SDLC untuk membantu dalam proses penilaian karyawan di PT Mitra Hutama Bersinar yang masih menggunakan *microsoft excel* dan menyebabkan hasil keputusan yang tidak tepat karena manajer sering lupa melakukan kegiatan penilaian. Tujuan dibuatnya sebuah *front end*, yaitu untuk memudahkan bagian komisaris sehingga mendapatkan hasil penilaian setiap karyawan sesuai standar operasional prosedur. Penelitian tersebut menghasilkam sebuah tampilan aplikasi yang dapat digunakan oleh manajer dan admin TI.

Berikutnya penelitian [15], menerapkan metode *agile development* menggunakan model XP yang menghasilkan sebuah aplikasi sistem penjualan

berbasis web untuk menanggulangi permasalahan proses penjualan yang masih manual di Toko ST Jaya. Aplikasi tersebut berhasil dibuat sesuai dengan kebutuhan dari pihak toko dan dapat membantu dalam proses penjualan maupun dalam proses pengolahan data pakaian toko. Pembelian pakaian melalui sistem tersebut dapat memberikan alternatif dan kemudahan dalam menjalankan sebuah bisnis online

Kemudian Penelitian [16], dilakukan perancangan aplikasi berbasis *android* menggunakan *flutter* dengan metode XP untuk mendigitalisasi karya tulis keagamaan dari Pondok Pesantren At-Tarbiyah Guluk-Guluk Sumenep. Teks-teks keagamaan atau karya-karya tersebut masih dalam bentuk media cetak yaitu buku. Tujuan membuat sebuah aplikasi berbasis Android agar dapat membantu mempermudah santri, alumni, dan simpatisan dalam mengakses Kitab Arudh & Qawafi secara praktis. Hasil penelitian menunjukkan 87% perhitungan dengan skala likert yang berarti pengguna aplikasi sangat setuju dengan adanya aplikasi tersebut.

Penerapan metode *XP* dan menggunakan Flutter juga dilakukan pada penelitian [17]. Penelitian tesebut dilakukan saat masa pandemi *Covid-19*. Sebuah klinik bernama Munjul Jaya belum memiliki sebuah aplikasi untuk mendukung program pemerintah untuk mengurangi tatap muka secara langsung. Klinik Munjul Jaya mengalami hambatan dalam proses penyampaian informasi kepada pasien karena belum memiliki layanan *online*. Penelitian berhasil menghasilkan sebuah aplikasi berbasis *android* untuk meningkatkan pelayanan konsultasi kesehatan pada Klinik Munjul Jaya dan mampu meningkatkan operasional bisnis yang lebih efisien yang mampu merespon setiap pasien tanpa ada keterbatasan waktu.

Salah satu penelitian yang membahas tentang penerapan *Clean Architecture* dilakukan pada penelitian [18]. Menggunakan metode *agile* dengan model *scrum* yang menghasilkan bagaimana kerangka *Clean Architecture* diterapkan pada *framework* flutter. Kemungkinan terjadinya *bug* yang tidak disengaja ketika sedang melakukan perubahan, efeknya mampu menambah biaya risiko dalam perancangan maupun pengembangan suatu sistem mampu untuk memperbaiki

masalah pada baris kode dan dapat memudahkan ketika mengubah atau menambahkan fitur pada aplikasi. Penelitian menghasilkan kerangka penerapan *Clean Architecture* yang menjadi cara untuk memperbaiki masalah pada baris kode yang sudah ada dan mampu memudahkan untuk mengubah atau menambahkan fitur pada aplikasi *mobile* yang dibuat.

Pada penelitian [19], merasa kesulitan menentukan metode pengembangan yang dapat menjamin kebutuhan konsumen atau klien terpenuhi. Sebuah toko penjualan alat telekomunikasi ditetapkan sebagai studi kasus untuk mencoba menerapkan metode *agile development* dengan model XP. Penelitian menghasilkan pernyataan bahwa dokumentasi desain sistem dalam metode XP hanya pada fase awal pengembangan yaitu fase perencanaan dan eksplorasi. Sehingga tidak ada alat ukur jika aplikasi dikatakan selesai dibuat. Serta, menyarankan untuk menambahkan fase khusus yang membahas proses pendokumentasian tanpa harus menghilangkan aspek agile yang dimiliki oleh XP.

Kemudian penelitian [20], meneliti tentang penggunaan Clean Architecture menggunakan metode *Test-Driven Development* dikarenakan adanya sebuah masalah muncul ketika beberapa vendor terlibat dalam pengembangan aplikasi, karena program-program tersebut sering mengalami gangguan dan upgrade membutuhkan waktu yang lama. Objek penelitian berupa aplikasi *tourism* pada kota Bantul. Penelitian menghasilkan bukti empiris, yang diperoleh melalui penggunaan sebuah aplikasi bernama LCOV untuk visualisasi hasil pengujian, menunjukkan tingkat keberhasilan sebesar 97,83% untuk pengujian unit, pengujian *widget*, dan pengujian integrasi. Keberhasilan eksekusi *clean architecture* terlihat dari kemampuannya untuk menyajikan data yang diinginkan secara efektif.

Tabel 2.1 Tinjauan Pustaka

No	Judul	Masalah	Metode	Hasil	Perbedaan
1.	Pengembangan Front end Sistem Informasi Pendataan Sekolah Desa Pendar Foundation Yogyakarta[14]	Pendar *Foundation* kesulitan dalam proses pencatatan dan pendataan karena masih dilakukan secara manual serta kesulitan untuk melakukan sinkronasi data	Metode *Waterfall* dan UCD	Hasil usability testing pada aspek learnability diperoleh sebesar 71,52% dengan aspek efficiency yaitu 0,009 tujuan/detik, dan aspek error sebesar 0.	Metode penelitian sebelumnnya menggunakan Waterfall dan UCD, Sistem yang dibuat adalah sistem pendataan sekolah berbasis web. Sedangkan penelitian yang akan dilakukan adalah *E-commerce* menggunakan metode *Agile* model *Extreme Programming yang menerapkan d*esain *Clean Architecture.* Sistem yang dibuat berbasis *android*.
2.	Rancang Bangun Front End Sistem Pemilihan Kinerja	Proses penilaian karyawan pada PT.	Tidak dicantumkan metode	Sebuah tampilan aplikasi yang	Penelitian sebelumnya tidak dijelaskan metode

No	Judul	Masalah	Metode	Hasil	Perbedaan
	Karyawan Berbasis *Android* Dan Situs Pada PT. Mitra Hutama Bersinar[7]	Mitra Hutama Bersinar menggunakan *microsoft excel* yang menghasilkan keputusan yang tidak tepat dan seringkali manajer lupa melakukan kegiatan penilaian	perancangan yang dilakukan	digunakan oleh *user* sebagai manajer dan admin TI	yang digunakan serta tidak melakukan integrasi ke server database. Sedangkan penelitian yang akan dilakukan adalah *E-commerce* menggunakan metode Agile model *Extreme Programming* yang menerapkan desain *Clean Architecture* untuk pengintegrasian *API*.
3.	Penerapan *Extreme Programming* Pada Sistem Informasi Penjualan Pakaian Berbasis Web (Studi Kasus Toko ST Jaya)[15]	Toko ST JAYA adalah salah satu toko pakaian yang masih menjual secara manual seperti, pencatatan penjualan, jumlah stok pakaian	Metode *Extreme Programming* (*XP*)	aplikasi sistem informasi penjualan berhasil dibuat sesuai dengan kebutuhan.dan diterima oleh pengguna.	Penelitian sebelumnya membuat sistem informasi berbasis web tanpa membahas teknis pengkodean. Sedangkan penelitian yang akan dilakukan menerapkan

No	Judul	Masalah	Metode	Hasil	Perbedaan
		yang tidak sesuai dengan yang ada dan lainnya.			desain *Clean Architecture* dan dibuat menggunakan *framework* flutter yang berbasis *android*.
4.	*Digitizing Arudh and Qowafi Classics as Android-Based Student Learning Media Using Flutter*[16].	Teks-teks keagamaan atau karya-karya Pondok Pesantren At-Tarbiyah Guluk-Guluk Sumenep masih dalam bentuk media cetak, yaitu buku.	Metode *Extreme Programming* (*XP*)	Dari hasil perhitungan skala likert pada kuesioner menunjukkan hasil 87% dengan keterangan pengguna sangat setuju dengan aplikasi	Penelitian sebelumnya membuat aplikasi media pembelajaran agama, database menggunakan sistem lokal. Sedangkan penelitian yang akan dilakukan adalah sebuah *E-commerce* yang menerapkan desain *Clean Architecture* untuk integrasi *API*. Sistem yang dibuat berbasis *android*.
5.	Aplikasi Hallo Sehat Berbasis *Mobile* Pada Klinik Munjul	Klinik Munjul Jaya mengalami hambatan	Metode *Extreme Programming* (*XP*)	aplikasi *android* Hallo Sehat berhasil	Aplikasi sebelumnya membuat aplikasi tentang

No	Judul	Masalah	Metode	Hasil	Perbedaan
	Jaya Purwakarta Menggunakan Metode *Extreme Programming*[17]	dalam proses penyampaian informasi kepada pasien karena masih belum memiliki layanan online		dirancang menggunakan *framework* Flutter untuk meningkatkan pelayanan seperti konsultasi kesehatan yang lebih efisien.	pelayanan kesehatan, database menggunakan firebase. Sedangkan penelitian yang akan dilakukan adalah sebuah *E-commerce* yang menerapkan desain *Clean Architecture* untuk integrasi *API* ke server database.
6.	Penerapan *Clean Architecture* Dalam Membangun Aplikasi Berbasis *Mobile* Dengan *Framework* Google Flutter[18]	kemungkinan terjadinya bug yang tidak disengaja ketika sedang melakukan perubahan, efeknya mampu menambah biaya risiko	Metode *Agile* dengan model *Scrum*	Hasil berupa kerangka penerapan *Clean Architecture* yang menjadi cara untuk memperbaiki masalah pada baris kode yang sudah ada dan mampu memudahkan untuk	Penelitan sebelumnya hanya terfokus membahas teori dari *Clean Architecture* serta aspek dalam menulis sebuah kode. Sedangkan penelitian yang akan dilakukan adalah implementasi langsung

No	Judul	Masalah	Metode	Hasil	Perbedaan
				mengubah atau menambahkan fitur pada aplikasi *mobile* yang dibuat.	dari *Clean Architecture* yang digambarkan dalam sebuah aplikasi *E-commerce*.
7.	Pendekatan Metodologi *Extreme Programming* pada Aplikasi *E-commerce* (Studi Kasus Sistem Informasi Penjualan Alat-alat Telekomunikasi)[19]	Kesulitan menentukan metode pengembangan yang dapat menjamin kebutuhan konsumen atau klien terpenuhi.	Metode *Agile* dengan model *Extreme Programming* (*XP*)	Dokumentasi desain sistem dalam metode XP hanya pada fase awal pengembangan yaitu fase perencanaan dan eksplorasi. Sehingga tidak ada alat ukur jika aplikasi dikatakan selesai dibuat.	Penelitan sebelumnya hanya terfokus membahas teori dari metode *Extreme Programming* dan tidak ada bahasan mengenai bagaimana pemrogramannya. Sedangkan penelitian yang akan dilakukan adalah implementasi langsung dari *Clean Architecture* yang digambarkan dalam sebuah aplikasi *E-commerce*.

No	Judul	Masalah	Metode	Hasil	Perbedaan
8.	*Implementing Flutter Clean Architecture for Mobile Tourism Application Development*[20]	terdapat masalah muncul karena program sering mengalami gangguan. Meskipun telah menerapkan praktik pemeliharaan, kode sulit dibaca yang menunjukkan kekurangan dalam pola keseluruhan dan pengelolaan unit kode individual.	*Metode Test-driven development* (TDD) yang merupakan bagian dari agile development	Bukti empiris, yang diperoleh melalui penggunaan metodologi *Test-Driven Development* dan LCOV untuk visualisasi hasil pengujian, menunjukkan tingkat keberhasilan sebesar 97,83% untuk pengujian unit.	Penelitian sebelumnya terbatas membahas mengenai seberapa efektif penggunaan *Clean Architecture*. Sedangkan penelitian yang akan dilakukan selain implementasi dari *Clean Architecture*, namun juga dilakukan dalam perancangan sebuah aplikasi E-commerce menggunakan metode *XP*.

Berdasarkan tinjauan pustaka yang telah dilakukan, beberapa penelitian terdahulu yang relevan telah dirangkum dan dikaji secara teliti. Penelitian-penelitian tersebut membahas penerapan metode *agile development* dengan model *extreme programming*, *clean architecture*, dan pengembangan *front end* pada aplikasi. Beberapa di antaranya menunjukkan hasil yang positif terkait peningkatan kualitas dan efektivitas pengembangan perangkat lunak.

Oleh karena itu, penelitian yang akan dilakukan juga akan menerapkan metode *agile development* khususnya *extreme programming* sebagai model pengembangannya dikarenakan fasenya sangat berkesinambungan dengan penerapan *clean architecture* yang mengutamakan dari segi kualitas kode yang *testable* dan *maintanable*.

2.2 Landasan Teori

2.2.1 Rancang Bangun

Rancang adalah sebuah kata dasar dari "merancang" merupakan serangkaian langkah yang digunakan untuk mengubah hasil analisis sistem ke dalam bahasa pemrograman. Hal ini bertujuan untuk menggambarkan secara rinci bagaimana semua komponen-komponen sistem diterapkan. Sementara itu, bangun yang merupakan kata dasar dari "membangun" dalam konteks membangun sebuah sistem adalah proses membangun sesuatu yang baru atau mengganti dan meningkatkan sistem yang telah ada secara keseluruhan atau sebagian[21].

Rancang bangun adalah suatu kegiatan yang kompleks dan penting dalam pengembangan sistem informasi. Kegiatan ini bertujuan untuk memastikan bahwa sistem informasi yang dibangun memenuhi kebutuhan pengguna dan dapat berfungsi secara optimal[22]. Jadi, dapat dikatakan bahwa rancang bangun adalah kegiatan yang bertujuan untuk merancang sistem baru guna mengatasi berbagai masalah yang dihadapi perusahaan, yang didasarkan pada pemilihan alternatif sistem terbaik. atau juga bisa diartikan sebagai proses pembangunan sistem untuk menciptakan sistem baru maupun mengganti atau

memperbaiki sistem yang telah ada baik secara keseluruhan maupun hanya sebagian.

Seiring perkembangan zaman, proses perancangan dan pengembangan sebuah sistem dibagi menjadi menjadi dua bagian yaitu *front end* dan *back end*. Hubungan erat antara keduanya memainkan peran penting dalam evolusi seorang pengembang web atau aplikasi, terutama dengan pertumbuhan yang pesat dalam industri teknologi dan peningkatan jumlah perusahaan yang fokus pada pengembangan web dan aplikasi[23]. Tujuan dibedakan antara *front end* dan *back end* adalah untuk memisahkan tugas dan tanggung jawab antara pengembangan antarmuka dan seluruh elemen visual aplikasi (*front end*) dan pengoptimalan fungsionalitas, pengelolaan basis data, dan logika pemrograman dari sisi *server* (*back end*). Berikut jenis pemisahan tugas dari bagian posisi tersebut:

2.2.2 *Front end*

Front end adalah bagian yang bertanggung jawab untuk membuat tampilan *website* atau aplikasi web yang interaktif dan menarik bagi pengguna atau sisi klien. Selain itu juga memiliki tugas sebagai untuk mengoptimalkan performa dan pengalaman pengguna yang efisien, serta mampu mendefinisikan dan mengkonsumsi kebutuhan *API* yang telah disediakan oleh *Back end*.

2.2.3 *Back end*

Back end adalah bagian yang bertanggung jawab untuk menciptakan proses logika bisnis dan mengakses sumber daya lain seperti basis data, server file, layanan *cloud*, dan lainnya. *Back end* developer juga bekerja dengan teknologi seperti sistem operasi, web server, bahasa pemrograman, dan membuat *API* yang nantinya dapat dikonsumsi oleh perangkat/klien lain.

2.2.4 Aplikasi

Aplikasi adalah perangkat lunak yang berisi serangkaian perintah untuk menjalankan tugas-tugas tertentu, seperti pengolahan data, dengan tujuan agar data tersebut dapat diolah sesuai dengan kebutuhan spesifik. Aplikasi ini dirancang untuk memenuhi berbagai kebutuhan dalam berbagai aktivitas[24]. Perangkat lunak ini dirancang dengan tujuan memberikan layanan dan solusi

untuk mendukung dan mempermudah pelaksanaan berbagai tugas atau kegiatan pengguna.

Aplikasi dapat mencakup berbagai fungsi, mulai dari pengelolaan data hingga penyediaan antarmuka yang memudahkan interaksi pengguna dengan sistem. Tujuan utama dari pembuatan aplikasi adalah untuk memberikan pengalaman pengguna yang baik dan efisien, sesuai dengan keperluan yang diinginkan oleh pengguna. Aplikasi memiliki jenis tertentu berdasarkan platformnya sebagai berikut:

a) Desktop

   Merupakan aplikasi yang dirancang untuk dijalankan pada komputer desktop atau laptop tanpa memerlukan koneksi internet.

b) Web

   Merupakan jenis aplikasi yang hanya diakses melalui sebuah aplikasi browser dan membutuhkan koneksi internet

c) Mobile

   Merupakan jenis aplikasi yang hanya diakses melalui perangkat bergerak yaitu handphone

Berdasarkan pengertian di atas, disimpulkan bahwa aplikasi adalah program yang telah dirancang dan dibangun untuk memenuhi kebutuhan pengguna dalam berbagai aktivitas, termasuk pengolahan data. Aplikasi dapat berupa perangkat lunak yang berdiri sendiri atau bagian dari sistem yang lebih besar.

2.2.5 *E-commerce*

E-*commerce* adalah proses membeli, menjual, mentransfer, atau bertukar produk, layanan, atau informasi melalui jaringan komputer, termasuk internet. Perdagangan elektronik (E-*commerce*) merupakan aktivitas perdagangan yang dilakukan dengan memanfaatkan jaringan telekomunikasi, terutama internet[25]. Umumnya, *E-commerce* menyediakan sistem pembayaran elektronik seperti kartu kredit atau transfer bank online untuk kemudahan transaksi. Beberapa *E-commerce* juga menyertakan pengiriman fisik produk melalui jasa logistik ke pelanggan. Dengan demikian, inti dari *E-commerce*

adalah memanfaatkan teknologi internet untuk menghubungkan antara penjual dan pembeli serta memfasilitasi transaksi komersial atau jual-beli produk maupun jasa tertentu.

E-commerce secara umum didefinisikan sebagai transaksi perdagangan elektronik yang dilakukan melalui media online. Selain itu, *E-commerce* juga dapat didefinisikan sebagai proses bisnis yang menggunakan teknologi elektronik untuk menghubungkan konsumen, bisnis, dan masyarakat umum dalam bentuk perdagangan elektronik dan pembelian, penjualan, dan pertukaran barang, jasa, dan informasi secara elektronik[26].

E-*commerce* merupakan bagian dari bisnis elektronik yang mencakup tidak hanya perdagangan, tetapi juga kolaborasi dengan mitra bisnis, dukungan pelanggan, lowongan pekerjaan, dan lainnya. Selain teknologi jaringan internet, E-*commerce* memerlukan teknologi basis data, email, serta sistem transmisi dan alat pembayaran elektronik yang tidak selalu terkait dengan teknologi komputer[27].

2.2.6 *Flutter*

Flutter adalah sebuah *framework* atau kerangka kerja aplikasi *mobile* yang dikembangkan oleh *Google*. *Flutter* digunakan untuk membuat aplikasi *mobile* lintas platform. *Flutter* memiliki dua komponen penting, *Software Development Kit* (SDK) dan antarmuka pengguna *framework*. SDK adalah kumpulan alat yang digunakan untuk membuat aplikasi yang dapat berjalan di berbagai platform. *Framework* adalah komponen antarmuka pengguna yang memungkinkan untuk menyesuaikan elemen-elemen pembentuk tampilan (*widget*) seperti teks, tombol, dan navigasi.

Flutter menggunakan bahasa pemrograman *Dart* dan menyediakan berbagai fitur yang memungkinkan pengembangan antarmuka pengguna yang responsif dan menarik, serta fitur *hot reload* yang memungkinkan pengembang untuk melihat perubahan kode secara langsung tanpa perlu me-*restart* aplikasi. Manfaat-manfaat berikut ini membuat *Flutter* menjadi pilihan yang disukai untuk kerangka kerja pengembangan aplikasi *mobile multiplatform*[28]:

1. *Flutter* dibuat menggunakan kompilasi asli kode *Dart*, kinerjanya sangat dioptimalkan dan hampir seperti aplikasi asli.
2. *Flutter* menggunakan mesin grafis *Skia* milik *Google*, proses rendering antarmuka UI-nya sangat cepat dan dapat diandalkan di berbagai perangkat.
3. *Flutter* menawarkan berbagai fitur yang ramah bagi para pengembang, seperti *widget* inspektur untuk debugging UI dan *hot-reload* yang memudahkan untuk memvisualisasikan perubahan kode secara langsung.

Bahasa pemrograman *Dart* digunakan dalam pengembangan beragam tipe aplikasi modern, mulai dari *mobile*, *web*, *IoT*, *backend*, hingga game *development*. *Dart* termasuk bahasa pemrograman dengan tipikal dinamis sehingga lebih luwes digunakan. Dengan demikian, kode *Dart* dapat dijalankan bahkan sebelum proses final kompilasi ke *binary*. Fitur ini sangat mendukung sifat interaktif dan cepat dalam proses pengembangannya[29].

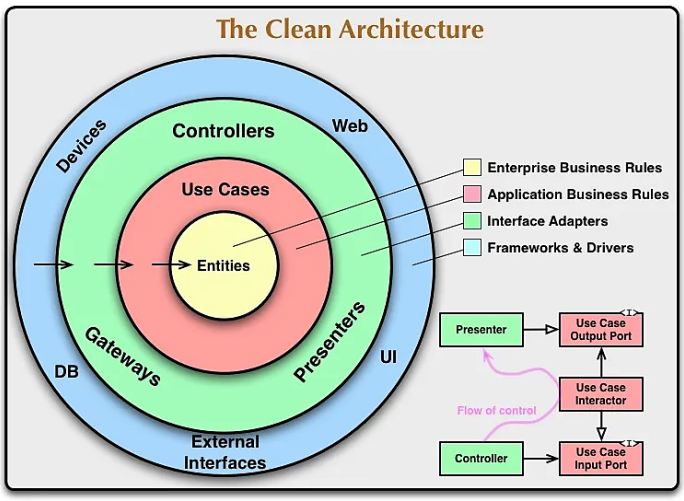
2.2.7 *RESTful* API (*Application Programming Interface*)

Application Programming Interface (API) adalah sekumpulan perintah, fungsi, dan protokol yang memungkinkan para pengembang perangkat lunak untuk mengintegrasikan dan membuat sistem perangkat lunaknya saling terhubung[30]. API didefinisikan sebagai lapisan protokol yang memfasilitasi pertukaran data antar sistem, sehingga memungkinkan bisnis untuk berbagi informasi dan fungsi aplikasi mereka dengan unit bisnis lain, departemen dalam perusahaan, dan pihak luar. API menyediakan akses data dan fungsionalitas aplikasi, serta menyediakan dokumentasi yang memfasilitasi integrasi aplikasi.

2.2.8 *Clean Architecture*

Arsitektur suatu sistem perangkat lunak adalah sebuah kerangka kerja yang menggambarkan struktur dan hubungan antar komponen-komponennya[31]. Arsitektur yang baik dapat membantu untuk memastikan bahwa sistem/perangkat lunak memenuhi kebutuhan pengguna, dapat berfungsi secara optimal, dan mudah untuk dimodifikasi atau dikembangkan.

Salah satu arsitektur perangkat lunak yang dapat digunakan adalah *Clean Architecture*. Arsitektur ini membagi perangkat lunak menjadi beberapa lapisan, di mana masing-masing lapisan memiliki tanggung jawab yang berbeda. Lapisan terdalam, yaitu lapisan domain, bertanggung jawab untuk logika bisnis. Lapisan di atasnya, merupakan lapisan aplikasi, bertanggung jawab untuk antarmuka pengguna dan sistem. Robert C. Martin, dalam bukunya menggambarkan *Clean Architecture* sebagai serangkaian lingkaran konsentris. Konsep ini dapat dilihat lebih rinci pada gambar 2.1, di mana setiap lingkaran merepresentasikan lapisan-lapisan berbeda dalam aplikasi.

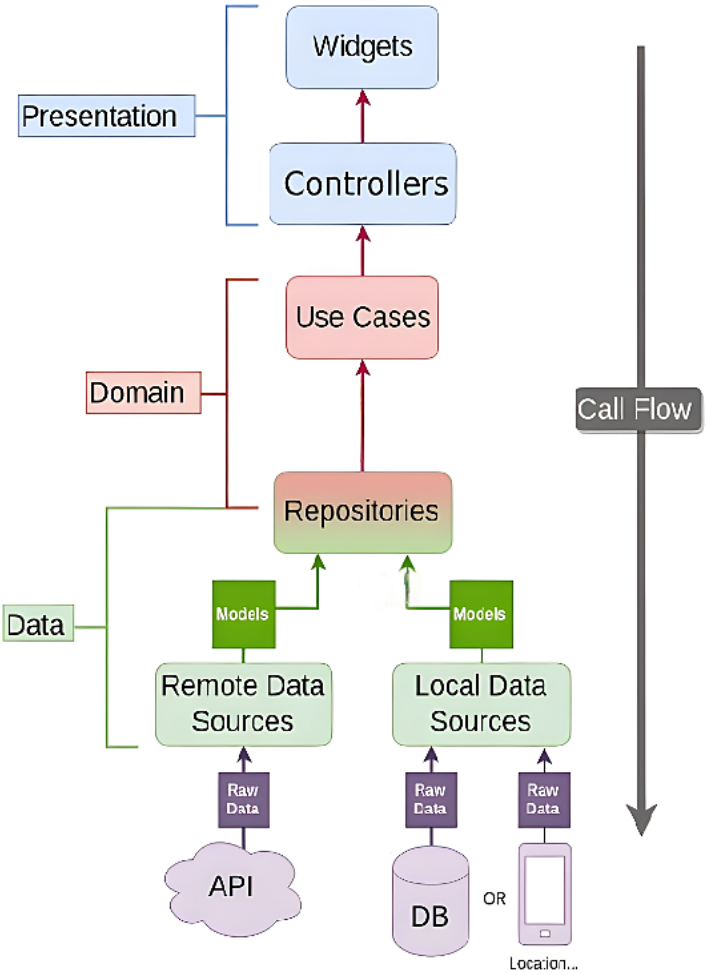


Gambar 2.1 Kerangka *Clean Architecture Robert C. Martin*[32]

Poin utamanya, *Clean Architecture* merupakan arsitektur yang relatif tidak memiliki standar yang tetap karena harus tetap disesuaikan oleh *programmer* masing-masing, namun tetap memiliki alur kerja yang sama yaitu memisahkan logika bisnis untuk memudahkan pengujian dan aplikasi yang dapat dipelihara[20].

Seiring perkembangannya, konsep *Clean Architecture* yang diperkenalkan oleh Robert C. Martin telah diadopsi secara luas dalam industri pengembangan perangkat lunak, tidak terkecuali dalam komunitas *Flutter*. Salah satu kontributor Flutter yang menerapkan prinsip *Clean Architecture* adalah Reso Coder, yaitu dengan membuat *boilerplate* yang memisahkan lapisan data, domain, hingga presentasi pada struktur projek *Flutter*. Hal ini

mempermudah pengembangan aplikasi Flutter yang kompleks dengan mengikuti standar *Clean Architecture* yang diimplementasikan langsung pada gambar 2.2 berikut



Gambar 2.2 Adopsi kerangka *Clean Architecture*[33]

Berikut masing-masing penjelasan dari 3-layer pada gambar 2.2:

a) Data

Lapisan data bertanggung jawab untuk mengakses dan memanipulasi sumber data, seperti database atau *API web service*. Tugasnya mencakup eksekusi *query*, *mapping* objek, hingga abstraksi. Lapisan ini berisi implementasi konkret dari operasi CRUD (*Create, Read, Update, Delete*) sesuai sumber data yang digunakan.

b) Domain

Lapisan domain bertanggung jawab untuk logika bisnis. Logika bisnis adalah aturan-aturan yang menentukan bagaimana data diolah dan tidak boleh terpengaruh oleh perubahan struktur data. Tugasnya memastikan konsistensi data yang digunakan oleh *use case* dan entitas bisnis. Lapisan ini harus benar-benar independen dari semua lapisan lainnya.

c) Presentation

Lapisan presentasi bertanggung jawab menyajikan informasi kepada pengguna (*user interface*) serta menerima input dari sebuah *widget* untuk diteruskan ke lapisan bisnis. Bisa berupa antarmuka berbasis web, *mobile* atau desktop.

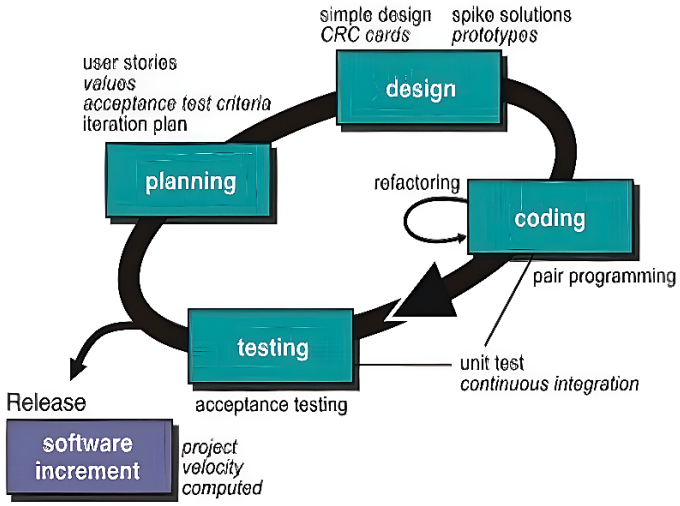
Penggunaan *Clean Architecture* dalam *flutter* terbukti sangat efektif dan menguntungkan. *Clean Architecture* mempercepat proses pengembangan dengan memaksimalkan potensi untuk bekerja secara paralel[12]. Dengan pemisahan lapisan tersebut, setiap bagian menjadi lebih fokus, mudah dikelola dan dikembangkan secara terpisah. Perubahan satu area minim berdampak luas ke area lain. Inilah salah satu keuntungan utama dari penerapan *Clean Architecture*.

2.2.9 *Extreme Programming*

Metode *agile* (tangkas) terkenal dengan model interaktif dan tambahannya. Program yang menggunakan teknik proses *agile* dibuat dengan desain minimal, pengujian bertahap, dan dokumentasi secukupnya[34]. Model pengembangan *agile* harus beradaptasi dengan pertumbuhan bisnis yang cepat. Sistem yang fleksibel juga akan meningkatkan produktivitas bisnis sekaligus menurunkan risiko kegagalan implementasi non-teknis. Diantara berbagai model pengembangan *Agile* yang tersedia, yang paling terkenal adalah *Extreme Programming* (XP).

XP adalah sebuah model *agile* yang cepat, efisien, dan fleksibel untuk mengembangkan perangkat lunak[35]. Hal tersebut memungkinkan pengembang mampu dengan cepat menyesuaikan sistem berdasarkan umpan

balik pengguna, sehingga risiko kegagalan dapat diminimalkan. Iterasi dan fleksibilitas requirement inilah yang menjadi ciri khas metodologi *Agile*, seperti yang dapat dilihat pada gambar 2.3.



Gambar 2.3 Alur *Extreme Programming* (*XP*)[17]

Metodologi *Extreme Programming* (XP) pertama kali diperkenalkan oleh Kent Beck, seorang pakar rekayasa perangkat lunak yang ditugaskan untuk menangani proyek C3 milik Chrysler yang saat itu tengah menghadapi risiko kegagalan[19]. Secara singkat, XP lahir dari pengalaman nyata Kent Beck dalam menyelamatkan proyek perangkat lunak yang hampir gagal dengan menerapkan metode-metode yang lincah dan efektif.

Empat prinsip nilai dasar model *Extreme Programming* (XP) adalah *Communication* (Komunikasi), *Simplicity* (Kesederhanaan), *Feedback* (Umpan Balik), dan *Courage* (Keberanian)[19]. Prinsip ini menyoroti perlunya fleksibilitas dan kemampuan untuk menyesuaikan diri dengan perubahan kebutuhan user seiring dengan berjalannya proses pengembangan aplikasi kustom. Diharapkan dengan komunikasi yang intens antara klien dan *programmer*, mereka dapat saling memahami kebutuhan satu sama lain. XP memiliki empat tahapan proses, berikut adalah tahapan tersebut:

1. *Planning*

   Tahap ini adalah tahap perencanaan proyek, di mana tim pengembangan membuat rencana pengembangan yang detail dan menentukan bagaimana mereka akan mencapai tujuannya. Tim pengembangan juga memutuskan tentang spesifikasi perangkat lunak dan membuat jadwal pengembangan. Terdapat salah satu komponen pada tahap ini, yaitu *user stories*.

   User Stories adalah adalah sebuah instrumen yang dipakai dalam metode *Agile* untuk menguraikan fitur perangkat lunak yang diinginkan oleh pengguna akhir, mencakup informasi tentang siapa pengguna, apa yang mereka perlukan, dan mengapa hal tersebut diperlukan[36].

2. *Design*

   Tahap ini adalah tahap desain, di mana tim pengembangan menentukan bagaimana perangkat lunak akan terlihat dan bekerja. Tim pengembangan juga membuat model sistem dan memutuskan tentang bagaimana perangkat lunak akan dirancang.

3. *Coding*

   Tahap ini adalah tahap implementasi, di mana tim pengembangan menulis kode program. Proses pemrograman dilakukan setiap hari dan mencakup uji coba dan *debugging*. Tim pengembangan juga melakukan tes unit dan integrasi API dari *back end* untuk memastikan bahwa perangkat lunak berfungsi seperti yang diharapkan.

4. *Testing*

   Tahap ini adalah tahap pengujian, di mana tim pengembangan melakukan pengujian sistem dan memastikan bahwa perangkat lunak bekerja seperti yang diharapkan dan memenuhi spesifikasi. Tim pengembangan juga melakukan pengujian dan melakukan perbaikan apa pun yang diperlukan sebelum melakukan perilisan.

2.2.10 UML (*Unified Modelling Language*)

UML (*Unified Modeling Language*) merupakan bahasa pemodelan standar dalam industri teknologi informasi untuk memvisualisasikan, merancang, dan mendokumentasikan sistem perangkat lunak[37]. Dengan menyediakan notasi standar berupa diagram, UML menjadikan perancangan sistem perangkat lunak menjadi lebih terstruktur, konsisten, dan mudah dipahami pihak-pihak terkait dalam pengembangan sistem

UML adalah bahasa pemodelan yang digunakan untuk membangun perangkat lunak berorientasi objek. UML muncul karena kebutuhan untuk mendefinisikan, menggambarkan, membangun, dan mendokumentasikan sistem perangkat lunak secara visual[15].

a. *Use Case Diagram*

Menggambarkan hubungan antara aktor (*user*) dengan berbagai fungsi utama (*use case*) yang ada pada sebuah sistem. Bisa menunjukkan fungsionalitas apa saja yang ada. Sering digunakan untuk memodelkan *requirements* dari sudut pandang pengguna sistem. Simbol dan keterangan yang digunakan dalam pembuatan use case diagram dapat dilihat pada tabel 2.2.

Tabel 2.2 Simbol pada *Use Case Diagram*[38]

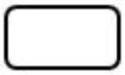
Simbol	Keterangan
	Aktor: Merepresentasikan peran seseorang, sistem lain, atau perangkat yang berinteraksi dengan use case.
	*Use Case*: Merupakan abstraksi dari interaksi antara sistem dengan aktor.
	*Association* : Abstraksi dari penghubung antara aktor den an use case
	Generalisasi: Menggambarkan spesialisasi aktor untuk bisa

	berpartisipasi dalam use case.
<<include>>	Include: Menunjukkan bahwa suatu use case sepenuhnya merupakan bagian fungsional dari use case lainnya.
<<extend>>	Extend: Menunjukkan bahwa suatu use case adalah tambahan fungsional dari use case lain apabila kondisi tertentu terpenuhi

b. *Activity Diagram*

Menggambarkan alur aktivitas dalam sebuah proses bisnis atau sistem. Bisa berupa aktivitas manual maupun otomatisasi. Membantu memahami urutan langkah dan proses pengambilan keputusan. Simbol dan keterangan yang digunakan dalam pembuatan *activity diagram* dapat dilihat pada tabel 2.3.

Tabel 2.3 Simbol pada *Activity Diagram*[38]

Simbol	**Keterangan**
	*Start*: Setiap diagram aktivitas memiliki status awal.
	*Activity*: Aktivitas yang dijalankan oleh sistem, biasanya dimulai dengan kata kerja.
	*Decision*: Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Line: Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	*End*: Menandakan status akhir yang dicapai oleh sistem; setiap diagram aktivitas memiliki status akhir.
	*Swimlane*: Memisahkan entitas bisnis

	yang bertanggung jawab atas aktivitas yang terjadi.

c. *Sequence Diagram*

Menggambarkan interaksi antar objek di dalam sebuah sistem, terutama urutan pesan yang dikirim antar objek. Berguna untuk memodelkan skenario/alur spesifik dari *use case* tertentu. Simbol dan keterangan yang digunakan dalam pembuatan *sequence diagram* dapat dilihat pada tabel 2.4.

Tabel 2.4 Simbol pada *Sequence Diagram*[38]

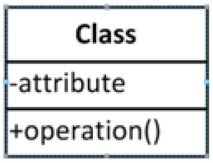
Simbol	Keterangan
	*Object*: Mempresentasikan class atau object
	*Activation boxes*: Menunjukkan durasi waktu yang dibutuhkan oleh sebuah objek untuk menyelesaikan tugasnya.
	Aktor: Merepresentasikan pengguna yang berinteraksi dengan sistem
	Lifeline: Menggambarkan "garis hidup" sebuah object.
	Message: Menggambarkan pesan atau interaksi antar objek
	Message to self: Menunjukkan pesan balikan atau reaksi dari objek itu sendiri.

d. *Class Diagram*

Menampilkan kelas-kelas yang ada pada sebuah sistem beserta relasinya. Bisa juga menunjukkan atribut & operasi setiap kelas. Membantu visualisasi struktur sistem terutama dari sisi desain perangkat

lunak. Simbol dan keterangan yang digunakan dalam pembuatan *class diagram* dapat dilihat pada tabel 2.5.

Tabel 2.5 Simbol pada *Class Diagram*[38]

Simbol	Keterangan
	• *Class*: struktur objek sistem yang akan ditampilkan dalam sistem informasi • *Attribute*: keadaan dari suatu objek didalam kelas • *Operation*: penggambaran mengenai fungsi yang terdapat dalam kelas
	*Association*: Relasi antar kelas dengan asosiasi biasanya juga disenai dengan multiplecities.

2.2.11 *Blackbox Testing*

Pengujian *Blackbox* pada perangkat lunak berfokus mengevaluasi kesesuaian fungsionalitas dan non-fungsional terhadap spesifikasi yang ditentukan tanpa mengetahui implementasi internal kode program. Pengujian dilakukan dengan mengeksekusi fungsi perangkat lunak dan menganalisis apakah *input* dan *output* sudah sesuai dengan yang diharapkan[24].

Salah satu teknik dalam metode Blackbox adalah *Equivalence Partitions*, yang merupakan pengujian berdasarkan input pada setiap *form* dalam sebuah aplikasi. Teknik ini membandingkan hasil yang diharapkan dengan hasil aktual untuk menarik kesimpulan apakah pengujian tersebut berhasil atau gagal[39]. Metode *Blackbox testing* cukup mudah digunakan karena hanya membutuhkan data uji valid dan tidak valid untuk setiap fungsi berdasarkan batas minimum dan maksimum yang diperbolehkan.

2.2.12 *Whitebox Testing*

Pengujian *whitebox* dilakukan untuk menguji dan menganalisis kode program guna mendeteksi kesalahan, dan dikenal sebagai pengujian *whitebox*. Ada pendapat lain yang menyatakan bahwa pengujian *whitebox* dilakukan dengan fokus pada kode murni tanpa memperhatikan tampilan antarmuka dari halaman aplikasi[40]. Pengujian ini berfokus pada struktur internal perangkat lunak dengan menganalisis kode sumber untuk menemukan kesalahan logika dan memastikan setiap jalur eksekusi dalam kode telah diuji dengan benar. Pengujian ini dilakukan tanpa mempedulikan tampilan antarmuka pengguna.

2.2.13 *User Acceptance Test* (UAT)

User Acceptance Testing (UAT) merupakan tahapan validasi akhir dalam siklus pengembangan perangkat lunak. Pada fase ini, beberapa aktor terlibat langsung untuk memastikan bahwa aplikasi yang dibangun telah memenuhi kebutuhan mereka dan berfungsi dengan baik dalam kondisi nyata penggunaan sehari-hari. Pengujian ini dilakukan setelah serangkaian pengujian menyeluruh lainnya, seperti pengujian unit, pengujian integrasi. Tujuan utama UAT adalah memverifikasi apakah aplikasi perangkat lunak tersebut telah sesuai dengan ekspektasi dan persyaratan dari pengguna akhir atau pelanggan, serta layak untuk digunakan sesuai tujuan yang diinginkan[41].

Pengujian UAT dilakukan menggunakan skala *likert* untuk menentukan tingkat penilaian terhadap berbagai aspek aplikasi, seperti kemudahan penggunaan, keandalan, kecepatan akses, dan tampilan antarmuka. Rincian bobot dari setiap jawaban yang ada dalam tabel 2.6 berikut[42].

Tabel 2.6 Bobot jawaban

Kode	Keterangan	Bobot
SS	Sangat Setuju	5
S	Setuju	4
C	Cukup	3
TS	Tidak Setuju	2

STS	Sangat Tidak Setuju	1

Hasil dari UAT merupakan dokumen yang menunjukkan bukti pengujian, berdasarkan bukti pengujian inilah dapat diambil kesimpulan, apakah aplikasi yang diuji telah dapat diterima atau tidak. Dapat ditunjukan pada tabel 2.7 berikut[42].

Tabel 2.7 Kriteria interpretasi skor

Indeks	Keterangan
0%-20%	Sangat tidak diterima
21%-40%	Tidak diterima
41%-60%	Cukup
61%-80%	Diterima
81%-100%	Sangat diterima

Acceptance terhadap aplikasi dapat diketahui dengan melakukan perhitungan dengan rumus sebagai berikut[42].

a. Rata-rata skor UAT:

$$\left(\frac{Total\ skor\ 1 +\ Total\ skor\ 2 +\ Total\ skor\ 3 +\ \ldots . + Total\ skor\ 6}{6}\right)$$

b. Indeks UAT:

$$\left(\frac{Rata - rata\ skor}{Skor\ maksimum}\right) \times 100\%$$

Hasil pengujian dilakukan dengan menggunakan rumus sebagai berikut[42].

a. Skor maksimum:

   n x 5 = 50 (n = Jumlah responden, 5 = bobot nilai tertinggi)

b. Skor minimum:

   n x 1 = 10 (n = Jumlah responden, 1 = bobot nilai terendah)

c. Indeks (%):

$$\left(\frac{Total\ skor}{Skor\ maksimum}\right) \times 100\%$$