SKRIPSI

ANALISIS PERBANDINGAN PERFORMA *STATE MANAGEMENT LIBRARY* PADA PENGEMBANGAN APLIKASI *MOBILE* MENGGUNAKAN *FRAMEWORK* FLUTTER

Disusun dan diajukan oleh:

AHMAD FATHANAH M.ADIL
D121191048

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2024

LEMBAR PENGESAHAN SKRIPSI

ANALISIS PERBANDINGAN PERFORMA STATE MANAGEMENT LIBRARY PADA PENGEMBANGAN APLIKASI MOBILE MENGGUNAKAN FRAMEWORK FLUTTER

Disusun dan diajukan oleh

Ahmad Fathanah M.Adil
D121191048

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 12 Februari 2024
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
A. Ais Prayogi Alimuddin, S.T., M.Eng.	Iqra Aswad, S.T., M.T.
NIP 198305102014041001	NIP 199011282019043001

Ketua Program Studi,

Prof. Dr. Ir. Indrabayu, S.T., M.T., M.Bus.Sys., IPM,ASEAN. Eng.
NIP 197507162002121004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini ;

Nama	: Ahmad Fathanah M.Adil
NIM	: D121191048
Program Studi	: Teknik Informatika
Jenjang	: S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

ANALISIS PERBANDINGAN PERFORMA STATE MANAGEMENT LIBRARY PADA PENGEMBANGAN APLIKASI MOBILE MENGGUNAKAN FRAMEWORK FLUTTER

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 12 Februari 2024

Yang Menyatakan

Ahmad Fathanah M.Adil

ABSTRAK

AHMAD FATHANAH M.ADIL. *Analisis Perbandingan Performa State Management Library pada Pengembangan Aplikasi Mobile Menggunakan Framework Flutter* (dibimbing oleh Ais Prayogi Alimuddin dan Iqra Aswad).

Dalam pengembangan menggunakan Flutter, manajemen *state* menjadi aspek penting. *State* mengacu pada informasi yang dapat berubah selama siklus hidup *widget* dan pengelolaan *state* yang efisien diperlukan untuk merender kembali hanya *widget* yang mengalami perubahan tanpa mempengaruhi seluruh aplikasi.

Penelitian ini mengkaji performa *state management* **Provider**, **BLoC**, **GetX** dan **Riverpod** pada pengembangan aplikasi *mobile* dengan studi kasus pada manajemen pendaftaran kegiatan kampus merdeka. Metrik yang diukur mencakup penggunaan CPU, penggunaan memori, waktu eksekusi, jumlah baris kode, dan *cyclomatic complexity*.

Hasil rata-rata dari 11 skenario menunjukkan perbandingan performa sebagai berikut: **Penggunaan CPU**: Riverpod (20.24%) < Provider (20.91%) < BLoC (20.34%) < GetX (20.98%). **Penggunaan Memori**: GetX (36.93 MB) < Provider (38.58 MB) < BLoC (38.15 MB) < Riverpod (40.4 MB). **Waktu Eksekusi**: Riverpod (478.44 ms) < GetX (537.91 ms) < BLoC (656.89 ms) < Provider (690.27 ms). **Jumlah Baris Kode**: GetX (5492) < Riverpod (5525) < BLoC (5592) < Provider (5553). ***Cyclomatic Complexity***: BLoC (133) < Riverpod (149) = GetX (149) < Provider (158).

Berdasarkan hasil eksperimen, Riverpod cocok untuk fokus pada performa CPU dan waktu eksekusi, GetX cocok untuk aspek memori dan kode yang lebih sederhana, BLoC cocok untuk kompleksitas yang lebih rendah, mudah diuji, dan dipelihara. Provider menjadi pilihan terakhir untuk aplikasi *mobile* yang kompleks. Penelitian ini diharapkan memberikan wawasan bagi pengembang dalam memilih *state management* yang sesuai dengan kebutuhan proyek mereka.

Kata Kunci: *Flutter, Analisis Performa, State Management.*

ABSTRACT

AHMAD FATHANAH M.ADIL. *Comparison Analysis of State Management Library Performance in Mobile Application Development Using the Flutter Framework* (*supervised by* Ais Prayogi Alimuddin and Iqra Aswad).

In Flutter development, state management becomes a crucial aspect. State refers to information that can change during the lifecycle of a widget, and efficient state management is necessary to re-render only the widgets that undergo changes without affecting the entire application.

This research examines the performance of state management libraries, namely **Provider**, **BLoC**, **GetX**, and **Riverpod**, in the development of a mobile application with a case study on the management of registration for campus independence activities. The measured metrics include CPU usage, memory usage, execution time, lines of code, and cyclomatic complexity.

The average results from 11 scenarios show performance comparisons as follows: **CPU Usage**: Riverpod (20.24%) < Provider (20.91%) < BLoC (20.34%) < GetX (20.98%). **Memory Usage**: GetX (36.93 MB) < Provider (38.58 MB) < BLoC (38.15 MB) < Riverpod (40.4 MB). **Execution Time**: Riverpod (478.44 ms) < GetX (537.91 ms) < BLoC (656.89 ms) < Provider (690.27 ms). **Lines of Code**: GetX (5492) < Riverpod (5525) < BLoC (5592) < Provider (5553). **Cyclomatic Complexity**: BLoC (133) < Riverpod (149) = GetX (149) < Provider (158).

Based on the experimental results, Riverpod is suitable for focusing on CPU performance and execution time, GetX is suitable for aspects of memory and simpler code, BLoC is suitable for lower complexity, ease of testing, and maintenance. Provider is the last choice for complex mobile applications. This research is expected to provide insights for developers in choosing appropriate state management for their projects.

Keywords : *Flutter, Performance Analysis, State Management.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN DAN ARTI SIMBOL

Lambang/Singkatan	Arti dan Keterangan
UI	*User Interface*
TC	*Test Case*
$\bar{x}$	Rata-rata

DAFTAR LAMPIRAN

KATA PENGANTAR

Bismillahirrahmanirrahim

Segala puji hanya milik Allah Subhanallahu Wa Ta'ala atas segala limpahan rahmat dan hidayah-Nya. Shalawat dan salam senantiasa tercurahkan kepada baginda Rasulullah Shallallahu 'Alaihi Wa sallam. Alhamdulillahirabbil'aalamiin, berkat ridha' dan kemudahan yang diberikan oleh Allah Subhanallahu Wa Ta'ala, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Analisis Perbandingan Performa State Management Library Pada Pengembangan Aplikasi Mobile Menggunakan Framework Flutter". Skripsi ini disusun sebagai salah satu syarat penyelesaian Program Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

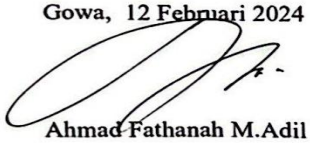
Penulis menyadari banyak kesulitan dan kendala yang dihadapi saat penyusunan tugas akhir ini. Dalam prosesnya, penulis memperoleh banyak bantuan, dukungan dan bimbingan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan terima kasih kepada:

1. Allah subhanallahu Wa Ta'ala, atas izin Rahmat, nikat dan kasih sayang-Nya sehingga skripsi ini dapat terselesaikan.
2. Kedua orang tua penulis, Bapak Muhammad Adil dan Ibu Bahriah, yang senantiasa mendoakan, memberikan dukungan dan memfasilitasi penulis dalam menyelesaikan perkuliahan.
3. Bapak A. Ais Prayogi Alimuddin, S.T., M.Eng. selaku pembimbing I dan bapak Iqra Aswad, S.T., M.T. selaku pembimging II, yang senantiasa menyediakan waktu, tenaga, pikiran dan menyediakan fasilitas kepada penulis serta memberikan perhatian yang luar biasa dalam mengarahkan penulis untuk menyelesaikan tugas akhir.
4. Segenap Dosen dan Staff Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah banyak membantu penulis selama masa perkuliahan.
5. Andi Gigatera Halil Makkasau, Agung Nugraha yang senantiasa menemami dan mendorong penulis untuk menyelesaikan tugas akhir ini.

6. Teman-teman Teknik Informatika Angkatan 2019 (S19NIFIER), teman-teman Lab Ubicons, serta teman-teman Kelas B yang telah memberi bantuan, dukungan, dan semangat selama masa perkuliahan dan penyusunan tugas akhir ini.
7. Teman-teman KKN Posko 5 Desa Ampekale Gel.108 (Cici, Nanda, Nufit, Ame, Alsa, Giga, Agung, Rauf, Ammar, Reza, Munawar, Zean, Alm. Aso, Akmal, Iqbal, Kak batara, Kak Rizqan, Kak Brama) yang telah memberikan pengalaman berkesan kepada penulis selama KKN di masa perkuliahan.
8. Serta berbagai pihak atas segala dukungan dan bantuan yang tidak dapat penulis tuliskan satu persatu.

Penulis berharap semoga Allah membalas segala kebaikan yang telah diterima oleh penulis dari berbagai pihak yang telah membantu mempermudah penulis dalam mengerjakan tugas akhir ini. Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna, oleh karena itu penulis mengharapkan segala bentuk saran serta masukan yang membangun dari berbagai pihak. Semoga tugas akhir ini dapat memberikan pengetahuan dan manfaat bagi penulis dan pembaca.

Gowa, 12 Februari 2024



Ahmad Fathanah M.Adil

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kebutuhan aplikasi *mobile* di dunia industri digital kian meningkat dari tahun ke tahun. Oleh karena itu para *developer* dituntut untuk membuat aplikasi yang dapat memenuhi segala kebutuhan perusahaan dari segala aspek yang tentunya dapat diunduh dari Google Play Store dan Apple App Store. Tetapi jika membuat aplikasi *mobile* secara *native* akan menimbulkan masalah biaya karena membutuhkan masing – masing tim untuk Android dan iOS (Nagaraj & Prabakaran, n.d.). Dengan demikian, dibutuhkan sebuah teknologi yang dapat langsung dijalankan di *platform* yang berbeda tanpa perlu lagi memisahkan kode projek untuk masing-masing *platform*. Dengan membuat aplikasi menggunakan teknologi *cross platform* tentunya akan mengurangi biaya pembuatan karena *developer* bisa fokus ke satu file projek saja.

Pada tahun 2018, Google merilis *framework* buatannya dengan nama "Flutter", yang menjanjikan banyak kelebihan sebagai solusi untuk membuat aplikasi *cross platform* (Faust, n.d.). Flutter menyediakan komponen *interface*-nya sendiri, mekanisme *rendering* dan alur pengembangan yang cepat, dan tentunya dapat mempertahankan performa asli dari *platform*-nya.

Dalam pengembangan aplikasi menggunakan Flutter, hal yang paling penting untuk diperhatikan bagi para *developer* adalah bagaimana cara mengelola *state* untuk memperbarui tampilan aplikasi. *State* adalah informasi yang dapat dibaca secara sinkron saat *widget* dibuat dan dapat berubah selama *widget* hidup (*State Class - Widgets Library - Dart API*, n.d.). Contoh umum ketika *state* berubah adalah ketika kita ingin memilih warna dari *color picker* atau memilih aksi dari *radio button*. Selama pengguna berinteraksi dengan UI, tentunya tampilan harus terus di-*render* kembali agar pengguna bisa melihat perubahan yang terjadi (Ventura, n.d.). Flutter menyediakan fungsi yang dapat digunakan untuk merubah *state* pada aplikasi yaitu setState(). Tetapi fungsi ini tidak direkomendasikan untuk sepenuhnya digunakan pada pengelolaan *state* karena ketika kita memanggil fungsi setState(), seluruh *widget* yang berada dalam *class*

tersebut akan di-*render* kembali. Tentu saja hal ini akan mempengaruhi performa pada aplikasi tersebut. Oleh karena itu, manajemen *state* diperlukan untuk mengatasi hal ini. Dengan manajemen *state*, kita dapat dengan mudah untuk me-*render* kembali *widget* yang hanya memiliki perubahan pada *state* tanpa harus me-*render* kembali seluruh *widget* yang ada (Prayoga et al., 2021).

Pemilihan *state management library* yang tepat, akan sangat berpengaruh ketika kita membuat aplikasi yang kompleks menggunakan Flutter. Ketika aplikasi semakin besar, pastinya kode aplikasi akan semakin bertumpuk. *State* yang disimpan dalam satu tempat akan lebih mudah untuk dikontrol daripada harus menduplikasinya disetiap *widget* yang tentunya membutuhkan banyak usaha untuk tetap sinkron dengan aplikasi (Ventura, n.d.). Secara umum, kompleksitas sebuah aplikasi akan tumbuh secara eksponensial dengan *state*-nya. Oleh karena itu, manajemen *state* diperlukan untuk mengatasi permasalahan ini. Saat ini terdapat banyak sekali *state management library* yang disediakan oleh Flutter di *website* resminya. Berdasarkan *website* resmi Flutter , 4 *library* paling populer saat ini adalah BLoC, Provider, GetX, dan Riverpod. BLoC adalah sebuah *state management library* yang dibuat dari *class stream* atau *observable*. Sementara Provider adalah versi lanjut dari *Inherited Widget* dengan mengkombinasikan *Change Notifier*. Kemudian, GetX berasal dari *reactive programming* yang disederhanakan dan Riverpod adalah versi lanjutan dari *Inherited Widget* dengan memperbaiki masalah yang dimiliki oleh Provider. Setiap *library* memiliki kelebihan dan kekurangannya masing-masing, baik dari segi kompleksitas maupun performa. Tentunya hal ini akan membuat *developer* kebingungan untuk memilih *library* yang cocok untuk digunakan ketika memulai *project* baru. Karena Flutter tergolong teknologi yang baru, masih sedikit studi yang membahas mengenai topik ini (Slepnev, n.d.).

Berdasarkan hal yang telah diuraikan diatas, pada penelitian ini penulis mengajukan penelitian untuk menganalisis performa *state management library* dalam pengembangan aplikasi *mobile* menggunakan Flutter. Adapun dalam penelitian ini, metrik yang akan diukur adalah penggunaan memori (*memory usage*), penggunaan CPU (*CPU usage*), waktu eksekusi (*execution time*), jumlah baris kode (*lines of code*) dan *cyclomatic complexity*.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan di atas maka rumusan masalah dalam penelitian ini adalah:

a. Bagaimana cara mengimplementasi *state management library* tersebut pada pengembangan aplikasi *mobile* menggunakan Flutter ?
b. Bagaimana menguji dan menganalisis performa dari *state management library* tersebut pada aplikasi *mobile* ?

1.3 Tujuan Penelitian

Penelitian ini memiliki tujuan sebagai berikut :

a. Mengimplementasikan *state management library* pada pengembangan aplikasi *mobile* menggunakan Flutter.
b. Untuk menguji dan menganalisis performa dari *state management library* tersebut pada aplikasi *mobile*.

1.4 Manfaat Penelitian

a. Bagi pembaca umum, dapat dijadikan sebagai referensi tentang cara pengimplementasian dan analisis performa dari beberapa *state management library* pada aplikasi *mobile* serta sebagai acuan untuk penelitian lebih lanjut.
b. Bagi *mobile developer*, dapat dijadikan referensi sebagai bahan pertimbangan untuk memilih *state management library* yang akan digunakan.
c. Bagi peneliti, dapat menambah pengetahuan mengenai cara mengimplementasikan *state management library* serta pengukuran performa pada pengembangan aplikasi *mobile*.

1.5 Ruang Lingkup

Adapun batasan masalah pada penelitian ini adalah sebagai berikut :

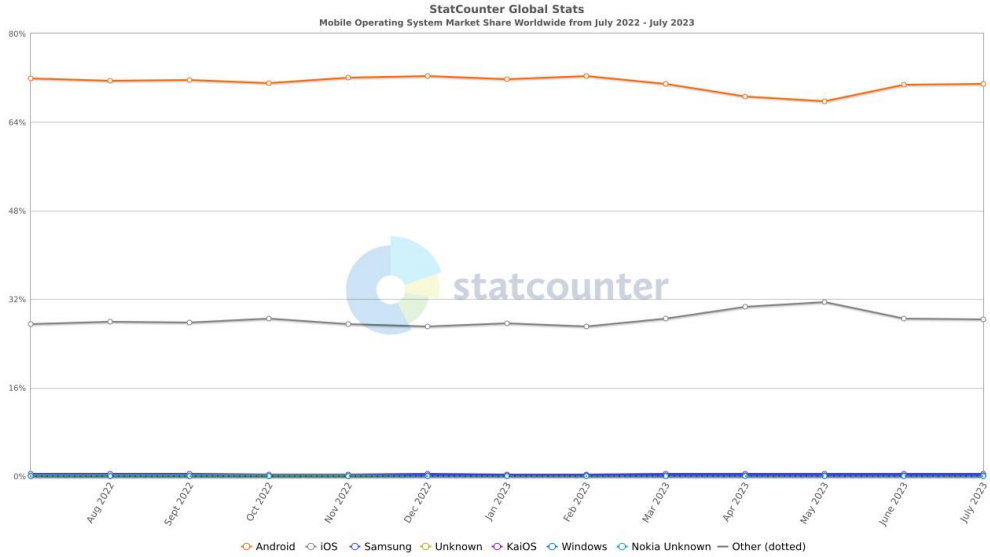
a. Bahasa pemrograman yang digunakan adalah Dart.
b. *Framework* yang digunakan adalah Flutter.
c. *State management library* yang akan diuji adalah BLoC, Provider, GetX dan Riverpod.

d. IDE yang digunakan adalah Android Studio.
e. Pengembangan aplikasi menggunakan *Clean Architecture*.
f. Pengukuran performa menggunakan Widget Test, Flutter Devtools, SonarQube, dan Snapdragon Profiler Tool.
g. Metrik yang akan diukur adalah penggunaan memori (*memory usage*), penggunaan CPU (*CPU usage*), waktu eksekusi (*execution time*), jumlah baris kode (*lines of code*) dan *cyclomatic complexity*.

BAB II
TINJAUAN PUSTAKA

2.1 Pengembangan Aplikasi *Mobile*

Pengembangan aplikasi *mobile* adalah proses dimana perangkat lunak yang terutama berjalan pada perangkat digital genggam seperti ponsel dikembangkan. Pengembangan ini merupakan industri yang terus berkembang yang sekarang menjadi bagian integral dari bisnis apa pun yang dapat kita pikirkan. Meskipun ada beragam ponsel yang tersedia di pasaran, Android dan iOS mengambil bagian terbesar (C. G. & Devi, 2021).



Gambar 1 Grafik persentase *market* sistem operasi *mobile* diseluruh dunia

Sumber : (*Mobile Operating System Market Share Worldwide*, n.d.).

Menurut laman *StatCounter* (*Mobile Operating System Market Share Worldwide*, n.d.), terdapat 70,9% pengguna android, 28,36% pengguna iOS dan sisanya pengguna *Operating System* lain yang tercatat hingga bulan Juli 2023 di seluruh dunia.

Saat mengembangkan aplikasi *mobile*, sama halnya seperti mengembangkan perangkat lunak biasa yang membutuhkan pemilihan bahasa pemrograman atau *framework* yang sesuai. Meskipun kita mungkin melihat Android dan pengembangan iOS secara individual, penting untuk melihat berbagai jenis pengembangan aplikasi *mobile* tersedia. Ada tiga jenis utama pengembangan aplikasi, yaitu :

2.1.1. *Native App Development*

Native App Development adalah pengembang membuat aplikasi dengan menargetkan *platform* tertentu seperti Android atau iOS. Pengembangan ini mendukung semua fitur yang disediakan OS dan juga memungkinkan kita untuk memanfaatkannya potensi maksimum perangkat seluler sehingga memberikan kinerja yang jauh lebih unggul daripada pengembangan aplikasi jenis lainnya. Untuk mengembangkan aplikasi Android, *developer* menggunakan Android Studio sebagai IDE resmi dan Java/Kotlin sebagai bahasa yang digunakan untuk membuat aplikasi. Sedangkan, pengembang iOS menggunakan XCode dan Objective-C/Swift sebagai bahasa pemrograman mereka untuk mengembangkan aplikasi. Pengembangan aplikasi asli memiliki kelebihan dan kekurangan tersendiri (C. G. & Devi, 2021).

2.1.2. *Cross-Platform App Development*

Cross-Platform App Development menawarkan solusi untuk mengembangkan aplikasi *mobile* tanpa perlu pengetahuan luas baik di Java atau Swift. Pengembang dapat menggunakan C# dengan Xamarin, Java Script dengan React Native, Dart dengan Flutter atau bahkan teknologi *web* dengan Adobe PhoneGap (Aurelius, 2020).

2.1.3. *Hybrid App Development*

Pengembangan aplikasi *hybrid* melibatkan kombinasi elemen-elemen dari aplikasi *native* dan aplikasi *web*. Dalam pendekatan ini, kode aplikasi dikembangkan menggunakan teknologi *web* seperti HTML, CSS dan JavaScript, kemudian ditempatkan dalam wadah (*container*) *native* untuk mengakses fitur-fitur perangkat.

Keuntungan utama dari pengembangan aplikasi *hybrid* adalah kemampuannya untuk mengurangi biaya dan waktu pengembangan dengan menggunakan satu kode sumber untuk beberapa platform. Namun, ada beberapa pertimbangan yang perlu diperhatikan, seperti performa yang mungkin sedikit

lebih rendah daripada aplikasi *native* dan keterbatasan akses ke beberapa fitur perangkat (C. G. & Devi, 2021).

2.2 Flutter

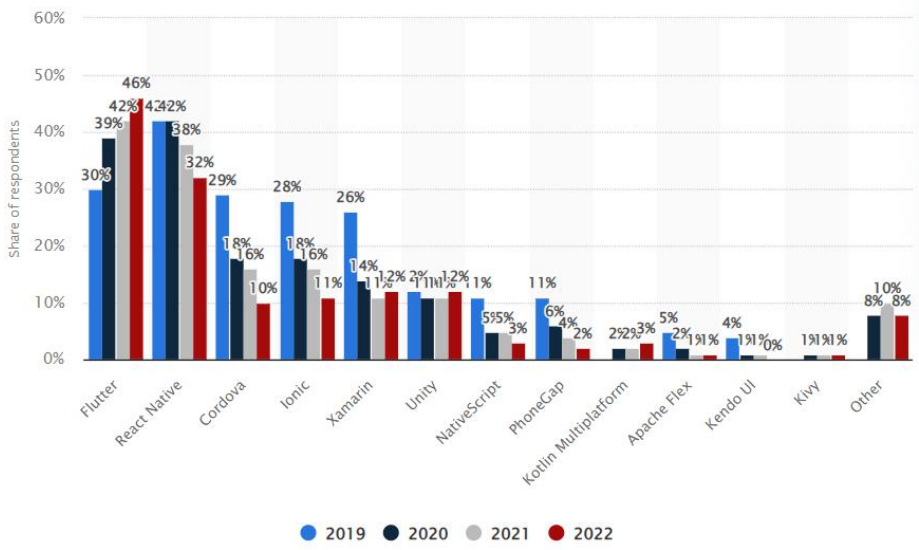
Flutter adalah *toolkit* UI portabel Google untuk membuat aplikasi yang dikompilasi secara *native* untuk *mobile*, *web* dan *desktop* dari satu basis kode. Flutter berfungsi dengan kode yang sudah ada, digunakan oleh *developer* dan organisasi di seluruh dunia, serta gratis dan *open source*. Flutter dapat digunakan oleh *programmer* yang akrab dengan konsep *object oriented programming* (*class*, *method, variable*, dll) dan konsep pemrograman imperatif (*loop, conditional*, dll) (*FAQ*, n.d.). Flutter dirilis pada tahun 2018 dan dikembangkan oleh Google serta merupakan SDK *open-source*, yang berarti komunitas dapat membantu mengembangkan dan meningkatkan kerangka kerjanya. Flutter seperti framework pembuatan aplikasi *hybrid* lainnya, menggunakan basis kode tunggal untuk berbagai *platform*, seperti *web*, *mobile* dan *desktop*. *Framework* ini bertujuan untuk memberikan kinerja yang sama seperti aplikasi *mobile* yang dibangun secara *native* (Sinatria et al., 2023).

Flutter dirancang untuk mendukung aplikasi *mobile* yang berjalan di Android dan iOS, serta aplikasi interaktif yang ingin dijalankan pada halaman *web* atau di *desktop*. Aplikasi yang perlu menghadirkan desain bermerek tinggi sangat cocok untuk Flutter. Selain itu, Flutter juga dapat membuat pengalaman piksel sempurna yang cocok dengan bahasa desain Android dan iOS. Ekosistem *package* Flutter mendukung beragam perangkat keras (seperti kamera, GPS, jaringan dan penyimpanan) dan layanan (seperti pembayaran, penyimpanan *cloud*, autentikasi dan iklan) (FAQ, n.d.) .

Flutter berbeda dari kebanyakan opsi lain untuk membangun aplikasi *mobile* karena tidak bergantung pada teknologi *browser web* atau rangkaian *widget* yang disertakan dengan setiap perangkat. Sebagai gantinya, Flutter menggunakan mesin *rendering* performa tinggi miliknya sendiri untuk menggambar *widget* (FAQ, n.d.).

Selain itu, Flutter berbeda karena hanya memiliki lapisan tipis kode C/C++. Flutter mengimplementasikan sebagian besar sistemnya (pengomposisian, gestur, animasi, kerangka kerja, *widget*, dll.) di Dart (bahasa modern, ringkas,

berorientasi objek) yang dapat dengan mudah didekati oleh pengembang untuk dibaca, diubah, diganti atau dihapus. Hal ini memberi pengembang kontrol yang luar biasa atas sistem, serta secara signifikan menurunkan standar kemampuan untuk didekati untuk sebagian besar sistem (FAQ, n.d.).



Gambar 2 Grafik pertumbuhan penggunaan *cross-platform framework*

Sumber : (Sujoy, 2023).

Flutter adalah *mobile framework* lintas *platform* paling populer yang digunakan oleh *developer* global, menurut survei *developer* tahun 2022. Berdasarkan survei, 46 persen pengembang perangkat lunak menggunakan Flutter. Secara keseluruhan, kira-kira sepertiga *mobile developer* menggunakan *framework* atau *cross-platform technologies*; *mobile developer* lainnya menggunakan *native tools* (Sujoy, 2023).

2.3 *Clean Architecture*

Clean Architecture menyediakan metode yang digunakan untuk menyusun aplikasi secara arsitektural dan untuk memecahkan masalah manajemen *state*. Tujuan mendasar dari *clean architecture* adalah pemisahan tanggung jawab dan skalabilitas. Dengan membagi sistem menjadi lapisan-lapisan yang memisahkan logika bisnis dari implementasi spesifik platform. Dengan model arsitektur berlapis, aplikasi *mobile* yang dikembangkan bisa menjadi *framework-agnostic*. Logika bisnis aplikasi tidak bergantung *external framework*. Dalam kasus Flutter,

lapisan bisnis aplikasi ditulis dengan bahasa pemrograman Dart tanpa mengetahui keberadaan logika tersebut dalam aplikasi Flutter (Sinatria et al., 2023).

2.4 *State*

State adalah representasi dari sistem yang diberikan pada suatu waktu. *State* dari suatu aplikasi memahami seluruh yang ada pada memori saat aplikasi sedang berjalan. Secara praktis, *state* digunakan untuk membangun kembali UI setiap waktu termasuk aset dari aplikasi, varibel, animasi, tekstur, *font*, status jaringan, pemanggilan *database*, *timer* dan lain-lainnya (Ventura, n.d.).

2.5 *State Management*

State management adalah suatu tempat yang memegang satu atau lebih kontrol antarmuka pengguna seperti inputan kata, tombol aksi, *radio button*, dll (Slepnev, n.d.).

Setiap elemen UI dinamis di Flutter harus memiliki *state* yang sesuai dan elemen ini dapat diubah hanya dengan mengubah *state* dan memberi tahu *framework* bahwa perubahan mungkin telah terjadi. Antarmuka pengguna biasanya berisi lebih dari satu elemen dinamis. Satu layar dapat memiliki puluhan atau bahkan ratusan *widget* dinamis, yang *state*-nya dapat berubah berdasarkan *state widget* lainnya (Slepnev, n.d.).

Jika suatu aplikasi menjadi lebih kompleks dan fungsionalitas seperti pemanggilan OS API, akses penyimpanan lokal, permintaan HTTP dan lain-lain ditambahkan, maka akan menjadi jauh lebih sulit untuk melacak *state* semua *widget* dan memperbarui status dengan benar dan efisien dengan tetap menjaga kode agar mudah dibaca, diuji dan dipelihara. Sehingga developer perlu menyadari bahwa diperlukan beberapa pendekatan yang lebih maju untuk memanajemen *state* aplikasi (Slepnev, n.d.) .

2.6 *Static Code Analysis*

Static code analysis atau analisis statis adalah aktivitas verifikasi perangkat lunak yang menganalisis kode sumber untuk *quality*, *reliability* dan *security* tanpa menjalankan kode. Menggunakan analisis statis, dapat mengidentifikasi kecacatan

dan kerentanan keamanan yang dapat membahayakan aplikasi. Analisis statis dapat menjadi pendekatan yang menghemat biaya untuk mengukur dan melacak metrik kualitas perangkat lunak tanpa biaya tambahan untuk menulis kasus uji atau melengkapi kode (*What Is Static Code Analysis?*, n.d.).

2.7 *Cyclomatic Complexity*

Cyclomatic complexity mengukur jumlah logika keputusan dalam satu modul perangkat lunak. *Cyclomatic complexity* digunakan untuk dua tujuan terkait dalam metodologi pengujian terstruktur. Pertama, memberikan jumlah tes yang direkomendasikan untuk perangkat lunak. Kedua, digunakan selama semua fase siklus hidup perangkat lunak, dimulai dengan desain untuk menjaga agar perangkat lunak tetap andal, dapat diuji dan dapat dikelola. *Cyclomatic complexity* sepenuhnya didasarkan pada struktur grafik aliran kontrol perangkat (Wallace et al., 1996).

2.8 SonarQube

SonarQube adalah alat peninjau kode otomatis yang dikelola sendiri yang secara sistematis membantu memberikan kode yang bersih (*SonarQube 10.1*, n.d.). SonarQube terintegrasi ke dalam alur kerja yang sudah ada dan mendeteksi masalah dalam kode untuk membantu melakukan pemeriksaan kode terus-menerus pada suatu projek. Alat ini menganalisis lebih dari 30 bahasa pemrograman yang berbeda dan terintegrasi ke dalam *pipeline* CI dan *platform* DevOps untuk memastikan bahwa kode memenuhi standar kualitas tinggi (*SonarQube 10.1*, n.d.).

Menulis kode yang bersih sangat penting untuk menjaga basis kode yang sehat. SonarQube mendefinisikan kode bersih sebagai kode yang memenuhi standar yang ditentukan yaitu kode yang andal, aman, dapat dipelihara, dapat dibaca dan modular, selain memiliki atribut kunci lainnya. Hal ini berlaku untuk semua kode yaitu kode sumber, kode pengujian, infrastruktur sebagai kode, dll (*SonarQube 10.1*, n.d.).

2.9 Flutter DevTools

DevTools adalah rangkaian alat kinerja dan *debug* untuk Dart dan Flutter. Berikut beberapa hal yang dapat dilakukan dengan DevTools (*DevTools*, n.d.) :

a. Periksa tata letak UI dan status aplikasi Flutter.
b. Diagnosis masalah performa UI di aplikasi Flutter.
c. Pembuatan profil CPU untuk aplikasi Flutter atau Dart.
d. Pembuatan profil jaringan untuk aplikasi Flutter.
e. Proses *debug* tingkat sumber dari aplikasi Flutter atau Dart.
f. *Debug* masalah memori di aplikasi baris perintah Flutter atau Dart.
g. Lihat *log* umum dan informasi diagnostik tentang aplikasi baris perintah Flutter atau Dart yang sedang berjalan.
h. Menganalisis kode dan ukuran aplikasi.

2.10 Snapdragon Profiler

Snapdragon Profiler adalah perangkat lunak profil yang berjalan di platform Windows, Mac, dan Linux. Snapdragon Profiler terhubung dengan perangkat Android yang ditenagai oleh prosesor Snapdragon® melalui USB. Snapdragon Profiler memungkinkan pengembang untuk menganalisis data CPU, GPU, DSP, memori, daya, termal dan jaringan, sehingga dapat menemukan dan memperbaiki *bottleneck* kinerja (*Snapdragon Profiler*, n.d.).

2.11 CPU Profiler

CPU Profiler memungkinkan pengguna untuk merekam dan membuat profil sesi dari aplikasi Dart atau Flutter. Profiler dapat membantu pengguna memecahkan masalah kinerja atau secara umum memahami aktivitas CPU aplikasi. Dart VM mengumpulkan sampel CPU (*snapshot* tumpukan panggilan CPU pada satu titik waktu) dan mengirimkan data ke DevTools untuk visualisasi. Dengan menggabungkan banyak sampel CPU, Profiler dapat membantu memahami dimana CPU menghabiskan sebagian besar waktunya (*Using the Memory View*, n.d.).

2.12 Memory View

Tampilan memori memberikan wawasan tentang detail alokasi memori aplikasi dan alat untuk mendeteksi dan men-*debug* masalah tertentu (*Using the Memory View*, n.d.).