Nusantara Journal of Multidisciplinary Science
Vol. 2, No. 1, Januari 2024
Hal 173-180
E-ISSN : 3024-8752
P-ISSN : 3024-8744
Site : https://jurnal.intekom.id/index.php/njms

Implementasi *Clean Architecture* Dalam Membangun Aplikasi Mobile Menggunakan Flutter

William Prapdeson Laksono[1], Bagas Satria Tri Wicaksana[2], Abdurrahman Yusuf Wijasena[3], Yoga Sahria[4]
[1,2,3]Fakultas Sains dan Teknologi, Universitas Teknologi Yogyakarta, Yogyakarta, Indonesia
[4]Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia

Article Info

Article history:

Received Januari 1, 2024
Revised Januari 2, 2024
Accepted Januari 10, 2024

Keywords:

Clean Architecture,
Aplikasi Mobile,
Flutter,
Pengembangan Perangkat Lunak

Keywords:

Clean Architecture,
Mobile Application,
Flutter,
Software Development

ABSTRAK

Clean Architecture, or better known as Clean Architecture, is a concept in software development that aims to create a software structure that is easy to understand, easy to test, and free from implementation details. So this research will test the development of mobile applications with Clean Architecture which uses BLoC state-management in Flutter-based applications. In the application being tested there are three features, namely the homepage display which displays all articles, the details display which displays details of a selected article and the display of articles that are liked and will be stored in the local database. The Clean Architecture test applied to the application this time was assessed in terms of maintainability of the features contained in the application. Clean Architecture makes it easy for developers to apply and develop a system and divide it into several parts so that they can divide it into a team and work on it together without destroying the main code itself. In terms of maintenance, it is also easier to do because changing a line of code will not damage the entire code that has been built. Application development using Clean Architecture produces good quality code that is maintainable for future development and provides time and load efficiency in the process of working on applications that are built without reducing the quality of the code produced.

ABSTRACT

Clean Architecture, atau yang lebih dikenal sebagai Arsitektur Bersih, adalah suatu konsep dalam pengembangan perangkat lunak yang bertujuan untuk menciptakan struktur perangkat lunak yang mudah dimengerti, mudah diuji, dan terbebas dari detail implementasi. Maka dari penelitian ini akan diuji pengembangan aplikasi mobile dengan Clean Architecture yang menggunakan BLoC state-management dalam aplikasi berbasis flutter. Dalam aplikasi yang diuji terdapat tiga fitur, yaitu tampilan beranda yang menampilkan semua artikel, tampilan detail yang menampilkan detail dari sebuah artikel yang dipilih serta tampilan artikel yang disukai dan akan disimpan dalam lokal database. Pengujian Clean Architecture yang diterapkan pada aplikasi kali ini dinilai dari segi maintainability pada fitur-fitur yang terdapat pada aplikasi. Clean Architecture memberikan kemudahan kepada pengembang untuk mengaplikasikan dan mengembangkan sebuah sistem serta membaginya ke beberapa bagian sehingga dapat membaginya kedalam sebuah tim dan mengerjakannya secara bersama-sama tanpa merusak kode utama itu sendiri. Dalam hal pemeliharaan juga lebih mudah dilakukan dikarenakan mengubah suatu baris kode tidak akan merusak keseluruhan kode yang telah dibangun. Pengembangan aplikasi dengan menggunakan Clean Architecture menghasilkan kode dengan kualitas baik yang maintainable untuk dikembangkan dikemudian hari serta memberikan efisiensi waktu dan beban dalam proses pengerjaan aplikasi yang dibangun tanpa mengurangi kualitas kode yang dihasilkan.

Corresponding Author:

William Prapdeson Laksono
Fakultas Sains dan Teknologi, Universitas Teknologi Yogyakarta,
Yogyakarta, Indonesia
Email: william.5200411152@student.uty.ac.id

1. PENDAHULUAN

Dalam era modern yang didominasi oleh kemajuan teknologi, aplikasi mobile telah menjadi bagian integral dari kehidupan sehari-hari kita. Kemampuan untuk merancang dan mengembangkan aplikasi mobile yang efisien dan andal telah menjadi tantangan yang semakin mendesak bagi para pengembang. Dalam upaya ini, sebuah pendekatan yang semakin dikenal sebagai "*Clean Architecture*" telah muncul sebagai kerangka kerja yang kuat dan efektif. *Clean Architecture*, atau yang lebih dikenal sebagai Arsitektur Bersih, adalah suatu konsep dalam pengembangan perangkat lunak yang bertujuan untuk menciptakan struktur perangkat lunak yang mudah dimengerti, mudah diuji, dan terbebas dari detail implementasi[1]. Penerapan *Clean Architecture* dalam sebuah kerangka kerja semakin popular serta memberikan banyak keuntungan dalam hal pemeliharaan kode, skalabilitas, serta pengujian[2]. Penggunaan *Clean Architecture* akan sangat mengurangi pengeluaran untuk melakukan pemeliharaan dan sangat mengurangi resiko kerusakan yang tidak disengaja.

Berdasarkan penelitian yang sudah dilakukan sebelumnya menggunakan Agile with Scrum dan menggunakan framework Google Flutter, menghasilkan sebuah kerangka dan penerapan dari setiap layer *Clean Architecture* dengan menerapkan *SOLID Principle*[3]. Penelitian lain juga menerapkan prinsip *SOLID* dalam penelitiannya meskipun hanya menerapkan prinsip *OCP, LSP, ISP*, dan *DIP* tanpa *SRP*, oleh karena itu, dapat dilakukan penelitian lebih lanjut dengan menambahkan aktor elemen ke dalam metamodel arsitektur untuk menyediakan layanan baru dan memenuhi Prinsip Tanggung Jawab Tunggal[4]. Dalam penelitian lain juga di sebutkan bahwa Sistem aplikasi Android yang menggunakan arsitektur MVP dan menerapkan prinsip-prinsip *Clean Architecture* di dalamnya mampu menghemat 42% waktu dan beban dalam pengembangan aplikasi[5]. Dalam beberapa penelitian lain juga membandingkan antara aplikasi yang menerapkan prinsip *Clean Architecture* dengan aplikasi yang menerapkan arsitektur *Model-View- Controller (MVC),* dapat diketahui bahwa penerapan prinsip *Clean Architecture* mempunyai tingkat *maintainability* lebih besar daripada penerapan arsitektur MVC yaitu 78.2% dibanding 54.8%, dimana 78.2% dapat dikatakan *maintainability* karena mendapatkan nilai lebih dari 67%[6]. Dalam penelitian yang membahas tentang perbandingan Reactive Programming and *Clean Architecture* menjelaskan bahwa membuat aplikasi dengan *Clean Architecture* lebih mudah untuk diuji serta mudah untuk menambahkan fitur baru[7]. Maka dari penelitian ini akan diuji pengembangan aplikasi mobile menggunakan *Clean Architecture* yang menggunakan BLoC *state-management* dalam aplikasi berbasis flutter.

Melalui penelitian ini, peneliti juga akan mendokumentasikan langkah-langkah dan praktik terbaik dalam menerapkan *Clean Architecture* saat membangun aplikasi mobile dengan Flutter. Flutter sendiri merupakan framework *cross-platform* yang sering digunakan pada proses membangun aplikasi dalam berbagai platform, seperti Android, iOS, Windows, Linux, Mac, dan lain-lain[8]. Peneliti akan menjelaskan komponen-komponen utama dalam *Clean*

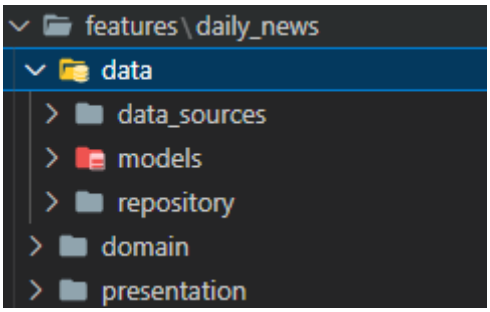
Architecture, termasuk *entity, use cases,* dan *repository*, serta bagaimana mereka berinteraksi satu sama lain untuk menciptakan aplikasi mobile yang stabil. Selain itu, penelitian ini juga akan membahas manfaat *Clean Architecture* dalam pengembangan aplikasi, seperti pemisahan logika bisnis dari lapisan tampilan, peningkatan kemampuan pengujian, serta fleksibilitas dalam pengembangan. Penelitian ini ditujukan untuk para pengembang Flutter, baik yang telah berpengalaman maupun yang baru memulai, serta bagi para profesional di bidang teknologi informasi yang tertarik untuk lebih memahami konsep *Clean Architecture* dalam konteks pengembangan aplikasi mobile. Dengan pemahaman dan penerapan *Clean Architecture*, diharapkan para pengembang akan mampu menciptakan aplikasi mobile yang lebih efisien, mudah untuk dipelihara, dan siap menghadapi perubahan yang cepat dalam dunia teknologi

2. METODE

Prinsip dari desain *Clean Architecture* ialah mendorong pemisahan code dengan tujuan membuat basis kode yang modular, terukur, dan dapat diuji. Dalam membangun aplikasi dengan menerapkan *Clean Architecture* maka perlu dibagi kedalam tiga layer, yaitu *Domain layer* yang berisi logika dan entitas bisnis, *Data layer* yang mengelola akses dan penyimpanan data, serta *Presentation layer* yang bertugas menangani UI dan interaksi pengguna[1].

2.1 Data Layer

Layer data merupakan inti dari *Clean Architecture*, tempat dari logika bisnis diimplementasikan serta tidak bergantung pada lapisan maupun fungsi lainnya. Lapisan ini terdiri dari *repository* dan sumber data, dimana menyediakan lapisan abstrak untuk mengakses dan memanipulasi data sedangkan sumber data dapat berupa API, database lokal, maupun penyedia data eksternal lainnya. Lapisan data melindungi lapisan domain dari detail penyimpanan dan pengambilan data[1]. Adapun layer data dapat dilihat pada gambar 1.

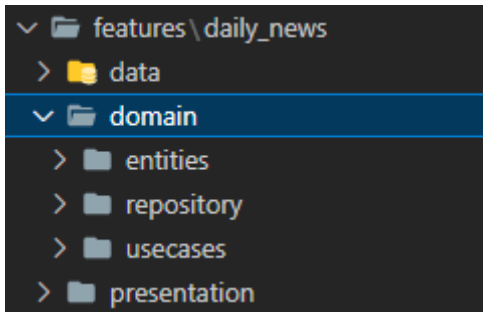


Gambar 1. *Data Layer*

2.2 Domain Layer

Layer domain mewakili semua logika bisnis inti aplikasi, use cases menentukan *method* atau operasi yang dapat dilakukan dalam aplikasi, sedangkan entitas bisnis mewakili objek penting dalam domain dan merangkum perilaku serta keadaannya. Lapisan domain juga harus bersifat mandiri atau *independent* terhadap layer-layer lainnya[1]. Contoh layer domain dapat dilihat pada gambar 2.

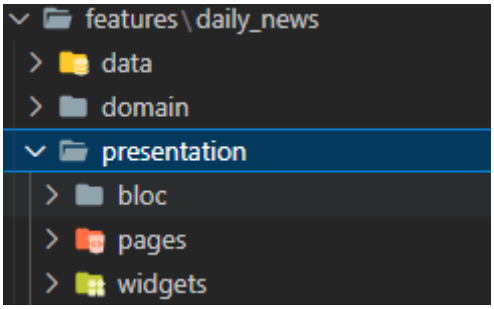
E-ISSN : 3024-8752
P-ISSN : 3024-8744



Gambar 2. *Domain Layer*

2.3 Presentation Layer

Lapisan ini berisi komponen dari antarmuka pengguna, seperti widget, layer, halaman, tampilan, dll. Layer presentasi bertanggung jawab dalam menangani interaksi pengguna dan merender tampilan antarmuka pengguna. Lapisan ini juga harus bersifat *independent* atau mandiri terhadap logika bisnis maupun detail implementasi akses data[1]. Adapun contoh layer presentasi dapat dilihat pada gambar 3.



Gambar 3. *Presentation Layer*

Sebelum mulai membangun sebuah aplikasi, perlu dipersiapkan dependencies apa saja yang akan digunakan selama membangun aplikasi, seperti *state-management* apa yang digunakan, bagaimana aplikasi dapat melalukan request ke database maupun API, menggunakan apa database lokal akan dibuat, serta API apa yang akan digunakan dalam aplikasi yang dibangun. Dalam contoh aplikasi kali ini yang dibangun menggunakan penyedia data external yaitu NEWS API, yaitu penyedia data external REST-API sederhana dan mudah digunakan dalam pencarian JSON untuk artikel berita terkini. Sedangkan REST-API sendiri merupakan antarmuka yang digunakan oleh dua system computer dalam bertukar informasi melalui internet[9]. Aplikasi ini juga menggunakan BLoC *state-management*, yaitu sebuah pola terstruktur untuk menangani aliran data dan peristiwa dalam aplikasi serta memisahkan lapisan presentasi dari logika bisnis[10]. Untuk melakukan request ke penyedia layanan external juga dibutuhkan Retrofit, yaitu sebuah *library* android yang bisa digunakan untuk melakukan consume API dengan cara memparsing data JSON ke dalam data class sehingga dapat ditampilkan dalam interface[11]. Selain penyedia data external, dalam aplikasi ini juga terdapat sebuah database lokal menggunakan paket Floor. Floor menyediakan abstraksi SQLite yang rapi dengan cara pemetaan otomatis antar objek dalam baris database dengan tetap menawarkan control penuh atas database SQL[12]. Dalam aplikasi yang diuji terdapat tiga fitur, yaitu tampilan beranda yang menampilkan semua artikel, tampilan detail yang menampilkan detail

dari sebuah artikel yang dipilih serta tampilan artikel yang disukai dan akan disimpan dalam lokal database.

3. HASIL DAN PEMBAHASAN

Dalam pembuatan sebuah aplikasi terkadang dibangun dengan tergesa-gesa sehingga mengakibatkan lebih sedikit waktu untuk membangun fitur baru karena terlalu sibuk dalam menangani kode-kode yang mengalami kesalahan, oleh karena itu *Clean Architecture* dibutuhkan. Setiap sistem aplikasi dibangun dengan dasar perilaku dan struktur, perilaku adalah tentang apa yang bisa dilakukan dari aplikasi yang dibangun sedangkan struktur sendiri mengacu pada bentuk kode dimana semakin sedikit kode yang digunakan maka semakin baik. Sebuah aplikasi harus mudah diubah seperti memperbaiki sebuah bug maupun menambahkan atau menghapus sebuah fitur.

Clean Architecture memberikan kemudahan kepada pengembang untuk mengaplikasikan dan mengembangkan sebuah sistem serta membaginya ke beberapa bagian sehingga dapat membaginya kedalam sebuah tim dan mengerjakannya secara Bersama-sama tanpa merusak kode utama itu sendiri. Dalam hal pemeliharaan juga lebih mudah dilakukan dikarenakan mengubah suatu baris kode tidak akan merusak keseluruhan kode yang telah dibangun. Pengujian *Clean Architecture* yang diterapkan pada aplikasi kali ini dinilai dari segi *maintainability* pada fitur-fitur yang terdapat pada aplikasi.

3.1 Beranda

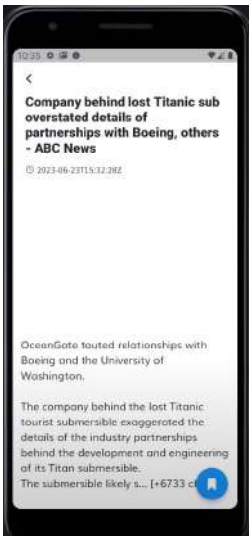
Fitur merupakan fitur inti dari aplikasi yang telah dibangun, seperti yang telah dijelaskan sebelumnya bahwa setiap fitur sendiri dibagi menjadi tiga layer, yaitu layer data, layer domain, dan layer presentation. Dengan difokuskannya masing-masing layer secara mandiri maka dapat membuat aplikasi yang dibangun menjadi bersifat modular. Pemrogaman modular merupakan sebuah teknik membagi struktur program yang lebih besar menjadi modul-modul yang lebih sederhana dan dapat dengan mudah untuk dilakukan pemeliharaan[13]. Tampilan dari fitur beranda dapat dilihat pada gambar 4.

Gambar 4. Beranda

Seluruh alur dari sumber data ditangani oleh layer data, *method-method* yang digunakan dalam mengolah data juga ditangani oleh layer domain, sedangkan layer presentasi menangani objek-objek yang ditampilkan pada layar.

3.2 Detail berita

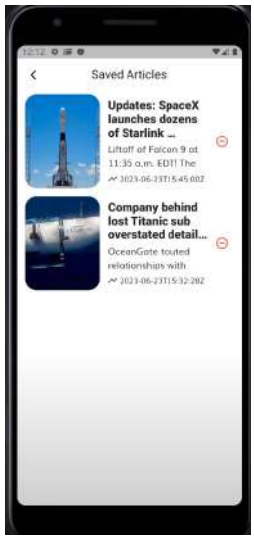
Fitur detail berita menampilkan detail dari artikel berita yang sebelumnya ditampilkan pada halaman beranda. Pada fitur ini ditampilkan keseluruhan judul berita, isi berita, tanggal terbit berita, sumber berita, serta gambar dari berita tersebut. Dikarenakan pembagian yang sudah dilakukan pada sebelumnya, pada fitur-fitur yang tersisa hanya perlu menambahkan *method* dan fungsi yang dibutuhkan. Tampilan dari fitur detail berita dapat dilihat pada gambar 5.



Gambar 5. Detail Artikel

3.3 Berita tersimpan

Pada fitur berita tersimpan, artikel yang sebelumya didapatkan secara daring disimpan ke dalam database lokal pada penyimpanan internal perangkat tersebut. Dengan sistem modular yang ditawarkan oleh *Clean Architecture* dapat memudahkan dalam mengubah sumber data mana yang akan digunakan pada fitur tertentu serta dapat juga diubah sewaktu-waktu apabila sumber data ekternal dari aplikasi yang dibangun berubah dikemudian hari. Pada layer data sendiri terdapat dua *data_source* dimana terdapat *remote data_source* dan *local data_source*. *Remote data_source* merupakan sumber data eksternal sedangkan *local data_source* membuat sebuah basis data secara lokal pada penyimpanan internal perangkat tersebut yang selanjutnya ditampilkan pada fitur ini. Fitur berita tersimpan dapat ditambahkan melalui tombol yang tersedia pada halaman detail berita sedangkan untuk mengakses halaman berita yang tersimpan dapat diakses melalui tombol yang terdapat pada tampilan beranda. Adapun detail dari berita yang telah disimpan dapat diakses melalui tampilan ini serta dapat juga menghapus berita yang sebelumnya sudah disimpan pada penyimpanan internal. Gambar dari tampilan berita tersimpan dapat dilihat pada gambar 6.



Gambar 6. Artikel Favorit

4. KESIMPULAN

Berdasarkan pengujian dan pembahasan yang telah disampaikan pada bab sebelumnya, maka dapat diambil kesimpulan bahwa dalam membangun sebuah aplikasi mobile dengan menggunakan *Clean Architecture* dapat memberikan banyak manfaat, seperti sistem yang modular, kemudahan dalam alur data hingga mengganti sumber data tanpa merusak keseluruhan kode, membagi menjadi beberapa modul dan layer sehingga dapat dikerjakan dengan tim secara bersamaan, memudahkan dalam melakukan pemeliharaan, penambahan fitur, serta memperbaiki kesalahan suatu fitur dikemudian hari. Pengembangan aplikasi dengan menggunakan *Clean Architecture* menghasilkan kode dengan kualitas baik yang *maintainable* untuk dikembangkan dikemudian hari serta memberikan efisiensi waktu dan beban dalam proses pengerjaan aplikasi yang dibangun tanpa mengurangi kualitas kode yang dihasilkan.

REFERENSI

[1] A. Tashildar, N. Shah, R. Gala, T. Giri, and P. Chavhan, "Aplication Development Using Flutter," *International Research Journal of Modernization in Engineering Technology and Science @International Research Journal of Modernization in Engineering*, pp. 2582–5208, 2020, [Online]. Available: www.irjmets.com

[2] R. C. Martin, *Clean Architecture*. 2017.

[3] M. Hilmy Bad Rudduja and R. E. Putra, "Penerapan *Clean Architecture* pada Aplikasi Pemesanan Makanan menggunakan Metode Slope One Algorithm," *Journal of Informatics and Computer Science*, vol. 3, no. 4, 2022.

[4] D. Sanchez, A. E. Rojas, and H. Florez, "Towards a *Clean Architecture* for Android Apps using Model Transformations," *International Journal of Computer Science* , vol. 49, no. 1, 2022, [Online]. Available: https://developer.android.com/jetpack

[5] M. B. Sinatria, Oman Komarudin, and Kamal Prihamdani, "Penerapan *Clean Architecture* Dalam Membangan Aplikasi Berbasis Mobile Dengan Framework Google Flutter," *INFOTECH journal*, vol. 9, no. 1, pp. 132–146, May 2023, doi: 10.31949/infotech.v9i1.5237.

[6] Aflah Taqiu Sondha, Umi Sa'adah, Fadilah Fahrul Hardiansyah, and Maulidan Bagus Afridian Rasyid, "Framework dan Code Generator Pengembangan Aplikasi Android

dengan Menerapkan Prinsip *Clean Architecture*," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 9, no. 4, pp. 327–335, Dec. 2020, doi: 10.22146/jnteti.v9i4.572.
[7] T. B. Duy, "Reactive programming and *Clean Architecture* in Android Development," 2017.
[8] H. Hussain, K. Khan, F. Farooqui, Q. Ali Arain, and I. Farah Siddiqui, "Comparative Study of Android Native and Flutter App Development," *International Conference on Internet*, pp. 36-37., 2021, [Online]. Available: https://www.statista.com/statistics/1020964
[9] M. Gilvy Langgawan Putra, M. Ihsan Alfani Putera, P. Studi Informatika, and J. Matematika dan Teknologi Informasi Institut Teknologi Kalimantan, "Analisis Perbandingan Metode SOAP dan REST yang Digunakan Pada Framework Flash Untuk Membangun Web Service," 2019.
[10] M. M. F. Abdillah, I. L. Sardi, and A. Hadikusuma, "Analisis Performa GetX dan BLoC *State-management Library* Pada Flutter Untuk Perangkat Lunak Berbasis Android," *Jurnal Penelitian Informatika*, vol. 1, pp. 73–78, 2023, doi: 10.25124/logic.v1i1.6479.
[11] J. Duarte and A. Ravara, "Retrofitting Typestates into Rust," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Sep. 2021, pp. 83–91. doi: 10.1145/3475061.3475082.
[12] L. P. De las Heras, O. R. Terrades, S. Robles, and G. Sánchez, "CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool," *International Journal on Document Analysis and Recognition*, vol. 18, no. 1, pp. 15–30, Mar. 2015, doi: 10.1007/s10032-014-0236-5.
[13] S. Amrizal, M. Kom, and Si, *Teknik Pemrograman Terstruktur*. Karya Mitra Sejati, 2014.