Jurnal Ilmiah Informatika (JIF)
| ISSN (Print) 2337-8379 | ISSN (Online) 2615-1049 |
Jurnal online di akses di http://ejournal.upbatam.ac.id/index.php/jif

# Implementasi Clean Architecture Menggunakan Metode Agile pada Pengembangan Aplikasi Android: Studi Kasus Buang.in

*Neng Fia Anggita Zalsabila*

Program Studi Teknik Informatika, Universitas Muhammadiyah Sukabumi, Jl. R. Syamsudin, S.H No 50, Kota Sukabumi 43113, Indonesia

INFORMASI ARTIKEL

*Sejarah Artikel:*
Diterima Redaksi: 29-07-2024
Revisi Akhir: 26-08-2024
Diterbitkan *Online*:05-09-2024

KATA KUNCI

Agile;

Android;

*Clean Architecture;*

*SDLC;*

*Trash;*

KORESPONDENSI

E-mail: nengfiaanggitazalsabilaa@gmail.com

**A B S T R A C T**

*Waste management presents a significant challenge that requires innovative strategies to enhance both efficiency and sustainability. Application development has been identified as a potential solution. However, the quality of application code often poses a major hurdle in maintenance and adaptation to changes. This study aims to improve the code quality and maintainability of the Buang.in application through the implementation of Clean Architecture. The expectation is that employing Clean Architecture will make the Buang.in application more structured, clear, and comprehensible, addressing existing maintenance challenges. Additionally, this research aims to provide effective waste management solutions by introducing flexible development options. The study's limitations include analyzing and improving code quality in the Buang.in application, with a particular focus on implementing Clean Architecture as a code maintenance solution. Changes to the core functionality of the application are not the primary focus. By incorporating Agile and Scrum methodologies, this research adopts adaptive and collaborative software development practices. Agile with Scrum enables development teams to adapt to evolving requirements and prioritize tasks based on the highest business value. Consequently, integrating Agile and Scrum in this research ensures that the development process of the Buang.in application remains efficient and responsive to changes throughout the development cycle.*

## 1. PENDAHULUAN

Undang-Undang No. 18 Tahun 2008 mengartikan sampah sebagai hasil sisa kegiatan manusia atau proses alam dalam bentuk padat atau semi-padat, yang dianggap tidak berguna dan dibuang ke lingkungan [1]. Penanganan sampah yang efektif memerlukan peran aktif dari semua pihak untuk mengurangi produksi sampah, melakukan daur ulang, serta menerapkan prinsip 3R (*Reduce, Reuse, Recycle)*.

Menurut data dari Kementerian Lingkungan Hidup dan Kehutanan (KLHK) tahun 2022, total sampah nasional mencapai 21,1 juta ton dengan 65,71% di kelola secara efektif, sementara 34,29% belum tertangani [2]. Di Indonesia penggunaan *smartphone* mencapai 89% dari populasi dengan pertumbuhan perangkat seluler sebesar 3,6% pada awal 2022 [3].

Agar sistem perangkat lunak menjadi lebih andal dan mudah dikelola, penting untuk menggunakan kerangka kerja yang fleksibel. Inilah sebabnya pendekatan *Clean Architecture* semakin populer, karena pendekatan ini menekankan pembangunan arsitektur perangkat lunak yang dapat dikembangkan dengan mengikuti prinsip-prinsip *Clean Code* [4]. *Clean Architecture* yang mengutamakan struktur perangkat lunak yang skalabel dan sesuai prinsip *clean code*, dapat memperbaiki pemeliharaan aplikasi. Pendekatan ini memungkinkan modifikasi pada dependensi eksternal tanpa mempengaruhi logika bisnis serta mempermudah pengujian dan penambahan fitur [5]. Pada penelitian milik Aflah Taqiu Sondha [6] menunjukkan bahwa penerapan *clean architecture* dapat mengurangi biaya dan waktu pengembangan aplikasi hingga 42%.

Metode SDLC (*software development life cycle*) dengan pendekatan agile dirancang untuk mengatasi masalah dalam pengembangan sistem, seperti perubahan permintaan pelanggan dan biaya tinggi, dengan waktu penyelesaian yang lebih singkat [7]. Metode Agile memiliki tujuan dan karakteristik yang sesuai untuk mengatasi permasalahan penelitian ini. Pendekatan ini menawarkan solusi efektif dalam pengelolaan sampah dengan memanfaatkan fasilitas pengembangan yang fleksibel. Proses Agile Development dipilih untuk meningkatkan kemampuan aplikasi dalam beradaptasi dengan kebutuhan yang terus berkembang[8].

Penelitian ini bertujuan menerapkan *clean architecture* pada aplikasi pengelolaan sampah buang.in untuk meningkatkan *maintainability* kode, termasuk kemampuan untuk menambah fitur baru, memperbaiki bug dan meningkatkan kualitas kode secara keseluruhan.

## 2. TINJAUAN PUSTAKA

### 2.1 Aplikasi

Menurut KBBI [Kamus Besar Bahasa Indonesia] aplikasi di definisikan sebagai model sistem untuk pengolahan data atau peraturan dalam bahasa pemrograman tertentu [9]. Sementara dalam kamus komputer beranotasi, aplikasi dijelaskan sebagai alat untuk memecahkan masalah dengan memanfaatkan teknologi komputer, yang bertujuan untuk memenuhi ekspektasi dalam pengolahan data [10]. Aplikasi menjadi lebih canggih dan mudah diakses, aplikasi mendukung efisiensi dan produktivitas kehidupan.

### 2.2 Sampah

Sampah adalah sesuatu yang kita temui setiap hari, karna makhluk hidup secara terus-menerus menghasilkan limbah. Penumpukan sampah yang melimpah memiliki dampak buruk yang signifikan terhadap lingkungan dan dapat menimbulkan masalah kesehatan.

### 2.3 *Software Development Life Cycle*

Model tradisional yang diterapkan secara sistematis dalam pembangunan perangkat lunak. Dalam pendekatan agile, persyaratan pengembangan sistem dibagi menjadi bagian-bagian kecil yang dapat dikembangkan secara bertahap dan berulang. Scrum adalah kerangka kerja yang memungkinkan pengembangan berulang dan penyelesaian tugas tepat waktu, sekaligus memaksimalkan hasil yang disampaikan [11]. Tim yang mengatur diri sendiri dapat menyelesaikan tugas dengan lebih cepat dan kualitas yang lebih baik. Motivasi diri yang tinggi tercapai dalam tim, yang menjadi alasan mengapa Scrum dapat meningkatkan produktivitas tim dengan cepat.

### 2.4 Kotlin

JetBrains memulis memulai proyek *open source* baru pada tahun 2010. Bahasa pemrograman yang diketik secara statis ini ditujukan untuk JavaScript, Native, JVM, dan Android [12]. Kotlin pertama kali dirilis pada bulan Februari 2016 dan tanggal 14 Juli 2021 itu versi 1.5.21. Bahasa kotlin dapat digunakan secara gratis karena proyek ini bersifat *open source*.

### 2.5 *Clean Architecture*

*Clean Architecture* menyediakan metode untuk menyusun struktur arsitektur aplikasi dan menangani masalah terkait state. Tujuan utamanya adalah memisahkan tamggung jawab dan meningkatkan kemampuan sistem untuk berkembang
Menurut [13], *Clean Architecture* mengintegrasikan beberapa arsitektur menjadi satu konsep untuk menghasilkan sistem yang dapat diuji. Tujuan utama arsitektur ialah memisahkan tanggung jwab dan memungkinkan skalabilitas, membagi sistem jadi beberapa lapisan, logika bisnis dari implementasi spesifikasi *platform* [14].

## 3. METODOLOGI

Metode penelitian adalah serangkaian prosedur atau langkah-langkah yang digunakan untuk memperoleh pengetahuan ilmiah. Menurut Prof. Dr. Suryana, metode penelitian cara sistematis untuk menyusun laporan penelitian [15]. Metode yang digunakan Agile dengan Scrum dalam *Software Development Life Cycle*.

### 3.1 Teknik Pengumpulan Data

Ada dua pengumpulan data: observasi dan studi pustaka. Data diperoleh melalui pengamatan langsung terhadap kegiatan penelitian yang terkait dengan masalah yang diteliti untuk mendapatkan informasi yang diperlukan secara menyeluruh. Data mencakup dokumen dan data lapangan yang diperoleh dari aplikasi buang.in termasuk hasil lint dari android studio.
Pengumpulan data dilakukan melalui studi pustaka. Sumber referensi yang digunakan meliputi jurnal, buku, *e-book* , dan artikel. Teori yang mendukung penelitian ini berkaitan dengan konsep metode agile dan *clean architecture*.

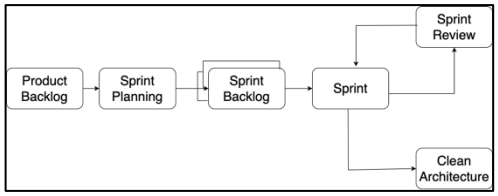
### 3.2 Perangkat Penunjang Penelitian

Tabel 3.1. Kebutuhan perangkat penelitian

| Perangkat | | Spesifikasi |
|---|---|---|
| Perangkat Keras/*Hardware* | : | Macbook Pro Mid 2012 |
| Perangkat Lunak/*Software* | : | MacOs |
| *Tools* | : | Android Studio |
| Bahasa Pemrograman | : | Kotlin |

## 4. HASIL DAN PEMBAHASAN

Pendekatan utama yang digunakan adalah agile dengan scrum untuk mengelola proses penelitian. Agile dengan scrum memungkinkan pengembangan secara iteratif yang berfokus pada kebutuhan pengguna dan perbaikan berkelanjutan. Metode ini dikenal sebagai fleksibilitas dan kemampuan untuk mengatasi tantangan bertahap.




Gambar 4.1. Tahapan-tahapan metode agile

Gambar 4.1 diatas ialah merupakan tahapan-tahapan metode agile yang akan diterapkan pada penelitian kali ini

### 4.1 *Product Backlog*

Tahap pertama adalah *product backlog* yang bertujuan untuk mengidentifikasi, memperbaiki dan mengubah aspek-aspek yang diperlukan dalam pengembangan aplikasi buang.in. Pengembang dapat merencanakan pekerjaan yang akan dilakukan. Berikut adalah *product backlog* yang ditentukan:

Tabel 4.1. Product Backlog

| No | Fitur dan Perbaikan | Deskripsi |
|---|---|---|
| 1 | Analisis dan Peningkatan, Perbaikan Kualitas Kode | Melakukan analisis mendalam pada kode aplikasi buang.in untuk mengidentifikasi area |

| | | |
|---|---|---|
| | | yang memerlukan perbaikan. |
| 2 | Perbaikan/ Kode | Menyempurnakan struktur dan kualitas kode pada Buang.in |
| 3 | Implementasi *Clean Architecture* | Mengimplementasikan *clean architecture,* untuk meningkatkan pemeliharaan dan pengembangan aplikasi. |

### 4.1.1 Analisis dan Peningkatan/Perbaikan Kualitas Kode

Melaksanakan evaluasi mendalam terhadap kode sumber aplikasi Buang.in guna menemukan dan mendokumentasikan area yang memerlukan perbaikan. Metode yang digunakan adalah Lint Tools di Android Studio untuk mendeteksi masalah umum seperti potensi bug, kode yang tidak efisien, dan pelanggaran standar kode. Hasil dari Lint Tools menunjukkan adanya 2 kesalahan dan 136 peringatan, dengan beberapa yang berprioritas sedang dan tinggi. Berikut ini adalah ringkasan hasil dari analisis kualitas kode tersebut:

Tabel 4.2. Hasil Resume analisis

| No | List Masalah | Penjelasan |
|---|---|---|
| 1 | Elemen ImageView tidak ada ContentDescription | Beberapa elemen ImageView dalam aplikasi tidak menyertakan atribut contentDescription, yang dapat mempengaruhi aksesibilitas. |
| 2 | Hardcode di layout | Dalam file layout XML, sering kali digunakan string literal yang hardcoded, alih-alih memanfaatkan resource string. |
| 3 | setTxt tanpa resource | Metode setText() dalam kode Java/Kotlin sering menggunakan string literal secara langsung tanpa merujuk pada sumber string. |
| 4 | Tidak ada setText18n | Tidak ada penggunaan annotation @StringRes atau android pada string literal dalam metode setText() |
| 5 | Logika bisnis tercampur | Logika bisnis dan kode UI tercampur, menciptakan ketergantungan antara keduanya yang dapat menyulitkan pemeliharaan. |
| 6 | Tidak ada pemisahan akses data & logika bisnis | Akses data dilakukan langsung dalam logika bisnis, mengurangi tingkat abstraksi dan menyulitkan pengujian. |

### 4.1.2 Restrukturisasi Kode

Perbaikan struktur dan kualitas kode pada aplikasi Buang.in untuk meningkatkan skalabilitas dan kemudahan pemeliharaan. Perbaikan dilakukan secara bertahap pada sprint kedua untuk meningkatkan kualitas kode dan keberlanjutan aplikasi.

### 4.1.3 Implementasi Clean Architecture

Buang.in saat ini menghadapi rintangan dalam pemeliharaan dan pengembangan efisien akibat struktur kode tidak terorganisir dengan baik. Implementasi *clean architecture* akan membantu memisahkan komponen UI, logika bisnis dan layer akses data sehinggaa aplikasi mudah dikembangkan, diuji dan dipeliharan ke depannya.

## 4.2 Sprint Pertama

Pada tahap ini dilakukan analisis menyeluruh terhadap kode aplikasi menggunakan tools di android studio yaitu “Lint” yang dilakukan secara manual. Fokusnya adalah memeriksa dan memperbaiki area yang telah di identifikasi dengan masalah umum di sprint kedua nanti, seperti masalah bug, kode tidak efisien dan standar kode yang tidak terpenuhi. Berikut ini hasil analisis dari lint:

Tabel 4.3. Hasil Lint

| No | Item Backlog | Deskripsi | Prioritas | Kriteria Solusi |
|---|---|---|---|---|
| 1 | Peninjauan Kode untuk Aksesibilitas | ImageView tidak memiliki atribut | Sedang | Semua ImageView memiliki atribut contectDescription yang relevan. |
| 2 | Peninjauan Kode Internasionalisasi | Penggunaan hardcoded string dalam file layout dan kode kotlin | Sedang | Tidak ada hardcode string dalam file layout dan kode kotlin. |
| 3 | Peninjauan Penggunaan setText18n | Penggunaan string literal dalam setText di RiwayatAdapter.kt | Sedang | Menggunakan sumber daya string untuk setText di RiwayatAdapter.kt |
| 4 | Refactoring kode untuk Aksibilitas | Penerapan atribut contentDescription pada elemen ImageView | Tinggi | Semua elemen ImageView dalam file layout memiliki atribut contectDescription |
| 5 | Refactoring kode Internasionalisasi | Ektraksi hardcoded string dari file layout ke strings.xml | Tinggi | Semua teks dalam file layout diambil dari string.xml |
| 6 | Refactoring kode setTextl18n | Perbaikan penggunaan setText dengan string literal | Tinggi | Menggunakan sumber daya string untuk setText |
| 7 | Penerapan *data layer* | Menerapkan lapisan data sesuai prinsip | Tinggi | Membuat komponen layer, dao database dan entity |
| 8 | Penerapan domain layer | Menerapkan lapisan domain dengan memisahkan logika bisnis | Tinggi | Membuat repository |

| | | | | |
|---|---|---|---|---|
| 9 | Penerapan presentation layer | Menerapkan presentation layer dengan memisahkan logika bisnis dari UI | Tinggi | Membuat ViewModel dan dihubungkan dengan adapter. |

## 4.3 Sprint Kedua

Pada sprint kedua ini, fokus utama adalah memperbaiki struktur dan kinerja aplikasi Buang.In. Tujuannya adalah untuk meningkatkan skalabilitas dan efisiensi aplikasi secara keseluruhan. Restrukturisasi akan dilakukan secara bertahap guna meningkatkan kualitas dan keberlanjutan aplikasi. Berikut adalah daftar perbaikan yang akan dilakukan pada tahap ini:

Tabel 4.4 Daftar-daftar perbaikan:

| No | Item Backlog | Deskripsi | Prioritas | Kriteria Selesai |
|---|---|---|---|---|
| 1 | Restrukturisasi Kode untuk Aksesibilitas | Implementasi atribut contentDescription pada elemen ImageView. | Tinggi | Semua elemen ImageView dalam file layout memiliki atribut contentDescription. |
| 2 | Restrukturisasi Kode untuk Internasionalisasi | Ekstraksi string hardcoded dari file layout ke strings.xml. | Tinggi | Semua teks dalam file layout diambil dari strings.xml. |
| 3 | Restrukturisasi Kode untuk setTextI18n | Perbaikan penggunaan setText dengan string literal. | Tinggi | Menggunakan sumber daya string untuk setText. |

### 4.3.1 Kode untuk aksebilitas

ImageView tidak memiliki atribut contentDescription dan lokasi kesalahan pada file di XMl, Widget non-text ini pada ImageView dan ImageButton diharuskan menggunakan atribut contentDescription untuk memberikan informasi text.

Sebelum perubahan pada activity_camera.xml

```
<ImageView
    android:id="@+id/captureImage"
    android:layout_width="65dp"
    android:layout_height="65dp"
    android:layout_marginBottom="30dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:srcCompat="@drawable/ic_capture"/>
```

Gambar 4.2. Activity_camera.xml

Setelah memperbaiki kode, berikut adalah hasil perbaikannya:

```
<ImageView
    android:id="@+id/captureImage"
    android:layout_width="65dp"
    android:layout_height="65dp"
    android:layout_marginBottom="30dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:srcCompat="@drawable/ic_capture"/>
android:contentDescription="@string/capture_image_description"
/>
```

Gambar 4.3 Activity_camera.xml setelah perbaikan.

### 4.3.2 Perbaikan kode untuk internasionalisasi

Hasil dari tools selanjutnnya ialah memperbaiki kode pada bagian internasionalisasi dan menunjukkan bahwa text *hardcode* di file layout. Tujuan pada perbaikan kode disini meningkatkan kualitas dan *maintenance* keberlanjutan dan semua text bisa disatu tempat yaitu string.xml.

Berikut kode sebelum diperbaiki activity_profile.xml

```
<com.google.android.material.button.MaterialButton
    android:id="@+id/btn_logout"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    style="@style/ButtonGreen"
    android:text="logout"
    android:layout_margin="24dp"/>
```

Gambar 4.4 Activity_profile.xml pada UI

berikut kode activity_profile.xml sudah diperbaiki:

```
<com.google.android.material.button.MaterialButton
    android:id="@+id/btn_logout"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    style="@style/ButtonGreen"
    android:text="@string/logout"
    android:layout_margin="24dp"/>
```

Gambar 4.5 Activity_profile.xml pada UI

### 4.3.3 Kode untuk setText18n

Perbaikan kode ini memperbaiki string.xml daripada string literal pada di setText menunjukkan warning masalah. Berada pada RiwayatAdapter.kt, tujuan memperbaiki kode disini untuk meningkatkan fleksibilitas kode.

```
if (tvBerat.text.toString().toInt() < 5) {
    tvStatus.text = "Masih dalam proses"
} else {
    tvStatus.text = "Sudah di konfirmasi"
}
```

Gambar 4.6 RiwayatAdapter.kt sebelum diperbaiki

```
val statusText = if (model.weight < 5) {
   binding.root.context.getString(R.string.masih_dalam_proses)
} else {
  binding.root.context.getString(R.string.sudah_di_konfirmasi)
}
```

Gambar 4.7 RiwayatAdapter.kt sesudah diperbaiki

## 4.4 Sprint Ketiga

Pada sprint ketiga ini ialah implementasi prinsip *clean architecture* pada aplikasi buang.in untuk meningkatkan pemeliharaaan kode.

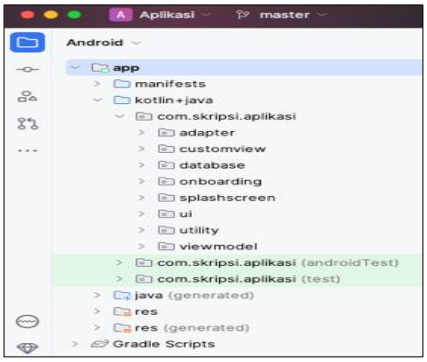
Tabel. 4.5. Sprint Ketiga Implmentasi Clean Architecture

| No | Product Backlog | Narasi | Prioritas | Penyelesaian |
|---|---|---|---|---|
| 1 | Implementasi Lapisan Data | Akses data secara langsung menyebabkan ketergantungan tinggi | Tinggi | Membuat dao dan mengonfigurasi database dan menambahkan entity |
| 2 | Implementasi Lapisan Domain | Logika bisnis tercampur dengan UI | Tinggi | Membuat repository dan usecase |

| | | | | |
|---|---|---|---|---|
| 3 | Implementasi Lapisan Presentation | Memisahkan logika bisnis dari UI agar tidak bergantung | Tinggi | Membuat ViewModel untuk kelola data UI dan adapter mengkonfigurasi dengan UI |

Pada bagian ini, penerapan clean architecture akan dilakukan dengan memisahkan logika bisnis dari UI, memisahkan akses data dari logika bisnis, dan memisahkan model dari domain. Proses ini melibatkan penerapan *data layer*, *domain layer*, dan *presentation layer*. Sebelumnya, aplikasi Buang.in belum menerapkan *clean architecture*.

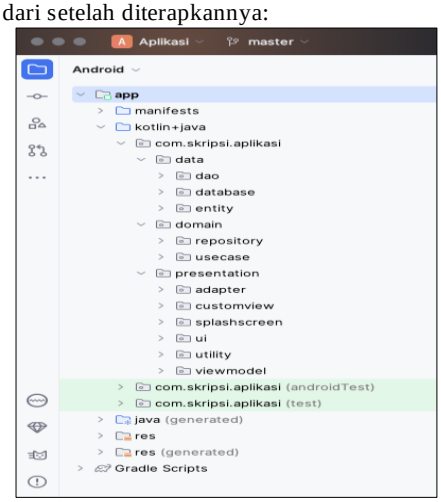
Berikut merupakan gambar struktur sebelum dilakukannya implementasi *clean architecture*




Gambar 4.8 Struktur sebelum implementasi *clean architecture*

Aplikasi ini memiliki struktur direktori yang mengelompokkan komponen-komponen seperti adapter UI, custom view, file database, fitur onboarding, splash screen, dan file UI berdasarkan fitur-fitur aplikasi seperti home, jemput sampah, jenis sampah, login, profile, register, dan riwayat. Selain itu, terdapat juga direktori untuk file utilitas dan ViewModel yang mengelola data UI. Namun, aplikasi ini belum menerapkan clean architecture, sehingga logika bisnis, UI, dan akses data masih tercampur dalam satu struktur.

Setelah menerapkan 3 layer *clean architecture,* berikut adalah hasil dari setelah diterapkannya:




Gambar 4.9 setelah implementasi *clean architecture*

Aplikasi ini mengadopsi prinsip clean architecture dengan membagi komponen-komponen aplikasi ke dalam direktori yang terstruktur dengan baik. Struktur tersebut mencakup:

***Data*:** Menyimpan file yang berhubungan dengan akses data, seperti DAO, konfigurasi database, dan model entitas. Ini memastikan bahwa pengelolaan data dan penyimpanan dipisahkan dari lapisan lain.

***Domain*:** Fokus pada logika bisnis aplikasi melalui repository dan use case. Repository memberikan abstraksi untuk akses data, sedangkan use case mengelola operasi bisnis utama.

***Presentation*:** Mengatur semua aspek terkait tampilan dan interaksi pengguna, termasuk adapter, ViewModel, serta elemen UI seperti activity dan fragment. Ini memisahkan logika tampilan dari logika bisnis dan pengelolaan data.

***Utility:*** Menyediakan file untuk kebutuhan utilitas umum, seperti pengaturan bitmap, fungsi bantuan, dan preferensi.

Jadi telah di implementasi *clean architecture* dengan memisahkan tanggung jawab antara lapisan data, domain, dan presentasi, yang berkontribusi pada peningkatan modularitas dan kemudahan pemeliharaan.

## 5. KESIMPULAN DAN SARAN

### 5.1.Kesimpulan

Penelitian ini berfokus pada penerapan Clean Architecture dalam pengembangan aplikasi Android "Buang.in" dengan menggunakan metodologi Agile Scrum. Clean Architecture dipilih untuk meningkatkan struktur, pemeliharaan, dan fleksibilitas kode, sementara Scrum diterapkan untuk mengelola proses pengembangan secara iteratif dan adaptif. Hasil penerapan menunjukkan bahwa struktur yang lebih terorganisir dan pemisahan tanggung jawab antara lapisan-lapisan kode meningkatkan efisiensi dan kualitas aplikasi secara signifikan. Meskipun demikian, tantangan seperti kebutuhan akan pelatihan tambahan dan dokumentasi yang lebih baik masih perlu diatasi.

### 5.2.Saran

**Perbaikan Dokumentasi**: Perbaiki dan tingkatkan dokumentasi proses pengembangan, khususnya terkait dengan penerapan *Clean Architecture* dan Scrum, untuk memastikan efisiensi dan kemudahan dalam pengelolaan proyek di masa depan.

**Penggunaan Alat Pengembangan**: Pertimbangkan untuk mengintegrasikan alat bantu tambahan yang dapat mendukung *Clean Architecture* dan Scrum, seperti alat pemantauan kinerja dan manajemen proyek, guna meningkatkan pengelolaan dan efisiensi proyek.

## UCAPAN TERIMA KASIH

Saya mengucapkan terima kasih yang tulus kepada pembimbing yang telah memberikan arahan, dukungan, dan bimbingan yang sangat berharga selama ini. Terima kasih khusus saya sampaikan kepada semua teman serta rekan yang telah memberikan dukungan dan masukan berharga selama proses ini. Terima kasih juga kepada pihak-pihak yang telah menyediakan data dan sumber daya yang diperlukan. Semoga tulisan ini bermanfaat dan memberikan kontribusi positif di masa depan.

## DAFTAR PUSTAKA


[1] Presiden Republik Indonesia, "Undang-undang (UU) tentang Pengelolaan Sampah," *Peraturan BPK*, 2008.

[2] KEMENKO PMK, "7,2 Juta Ton Sampah di Indonesia Belum Terkelola Dengan Baik," *Kementrian Koordinator Bidang Pembangunan Manusia dan Kebudayaan Republik Indonesia*, 2023. https://www.kemenkopmk.go.id/72-juta-ton-sampah-di-indonesia-belum-terkelola-dengan-baik

[3] N. Adisty, "Mengulik Perkembangan Penggunaan Smartphone di Indonesia," *GoodStats*, 2022. https://goodstats.id/article/mengulik-perkembangan-penggunaan-smartphone-di-indonesia-sT2LA#:~:text=Sementara%2C awal tahun 2022 ini,yang sama di tahun sebelumnya.

[4] M. H. Badrudduja and R. E. Putra, "Penerapan Clean Architecture pada Aplikasi Pemesanan Makanan menggunakan Metode Slope One Algorithm," *J. Informatics Comput. Sci.*, vol. 3, no. 04, pp. 506–514, 2022, doi: 10.26740/jinacs.v3n04.p506-514.

[5] T. B. Duy, "Reactive Programming and Clean Architecturein Android Development," no. April, pp. 1–47, 2017.

[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," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 9, no. 4, pp. 327–335, 2020, doi: 10.22146/jnteti.v9i4.572.

[7] M. Prabowo, *METODOLOGI PENGEMBANGAN SISTEM INFORMASI*. LP2M Press IAIN Salatiga, 2020.

[8] S. P. Budiarto and M. Dedi, "Desain Dan Perancangan Aplikasi Jemput Sampah Online Desa Rejosari Menggunakan Agile Development," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 7, no. 3, pp. 531–545, 2020, doi: 10.35957/jatisi.v7i3.529.

[9] Kementerian Pendidikan Kebudayaan Riset dan Teknologi Republik Indonesia, "Kamus Besar Bahasa Indonesia," *Badan Pengembangan dan Pembinaan Bahasa*, 2016. https://kbbi.kemdikbud.go.id/entri/aplikasi

[10] A. Juansyah, "Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System ( A-GPS ) Dengan Platform Android," *J. Ilm. Komput. dan Inform.*, vol. 1, no. 1, pp. 1–8, 2015.

[11] S. Pratama, S. Ibrahim, and M. A. Reybaharsyah, "Jurnal Penggunaan Metode Scrum Dalam Membentuk Sistem Informasi Penyimpanan Gudang Berbasis Web," *Intech*, vol. 3, no. 1, pp. 27–35, 2022, doi: 10.54895/intech.v3i1.1192.

[12] JetBrains, "FAQ: What is Kotlin," 2023. https://kotlinlang.org/docs/faq.html

[13] R. C. Martin, "Clean Architecture: A Craftsman's Guide to Software Structure and Design.," 2017.

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

[15] Ms. Prof. Dr. Suryana, "Metodologi Penelitian : Metodologi Penelitian Model Prakatis Penelitian Kuantitatif dan Kualitatif," *Univ. Pendidik. Indones.*, pp. 1–243, 2012, doi: 10.1007/s13398-014-0173-7.2.


## BIODATA PENULIS

**Neng Fia Anggita Zalsabila**

Merupakan Lulusan S1 Teknik Informatika Universitas Muhammadiyah Sukabumi. Selama masa kuliahnya, saya aktif dalam berbagai kegiatan organisasi mahasiswa, dan mengembangkan kemampuan kepemimpinan dan kerja sama tim. Dan saya tertarik pada bidang mobile. Profile LinkedIn: Neng Fia Anggita Zalsabila