PERANCANGAN APLIKASI PENGOLAHAN DATA SISTEM KEWASPADAAN DINI DAN RESPON MENGGUNAKAN *CLEAN ARCHITECTURE* DAN *DEPENDENCY INJECTION* BERBASIS ANDROID (STUDI KASUS: UPTD PUSKESMAS NATAR)

Habib Rizky.A.Siregar[1)], A. Ferico Octaviansyah Pasaribu[2)], Ade Surahman[3)]

1 Teknologi Informasi, Universitas Teknokrat Indonesia
2 Teknologi Informasi, Universitas Teknokrat Indonesia
3 Teknologi Informasi, Universitas Teknokrat Indonesia
1,2,3 Jalan ZA Pagar Alam No 9-11 Labuhan Ratu, Kedaton, Bandar Lampung
Email: [1]regarrezky@gmail.com, [2]fericopasaribu@teknokrat.ac.id, , [2]adesurahman@teknokrat.ac.id

Abstrak

Penelitian ini dilakukan atas dasar kebutuhan akan adanya sebuah aplikasi yang dapat digunakan sebagai media atau alat untuk mengelola data untuk kebutuahan sistem kewaspadaan dini dan respon yang dilakukan oleh petugas survailans epidemiologi di UPTD Puskesmas Natar. Dengan aplikasi ini diharapkan survailans dapat dengan mudah melakukan proses pengolahan data. Pada dasarnya aplikasi pengolahan data untuk kebutuhan sistem kewaspadaan dini dan respon dibangun sesuai dengan kebutuhan yang dibutuhkan oleh survailans dalam melakukan pengolahan data penyakit mingguan yang diderita masyarakat diwilayah UPTD Puskesmas Natar. Aplikasi yang dibuat berbasiskan sistem operasi Android, dengan menerapkan *clean architecture* dan *dependency injection*. Mekanisme dari penerapan 2 (dua) prinsip SOLID yaitu *Single Responsibility*, dan *Dependency Inversion* sebagai representasi dari kode yang bersih (*clean code*). Menggunakan metodologi pengembangan sistem *Extreme Programming* (*XP*). Pengujian fungsionalitas aplikasi yang dikembangkan diuji menggunakan Android Unit Testing. Keluaran dari penelitian ini adalah aplikasi pengolah data untuk keperluan sistem kewaspadaan dini dan respon yang dapat mengoptimalisasi pengolahan data berdasarkan kebutuhan untuk mencegah terjadinya kekeliruan data yang dihasilkan

Kata Kunci: SKDR, Android, *SOLID*, *Clean Architecture*, *Dependency Injection*, Kode Bersih, *XP*

1. Pendahuluan

Teknologi Informasi telah menjadi bagian dari kehidupan masyarakat dunia dengan perkembangannya yang sangat cepat dan terbukti berperan dalam berbagai kegiatan. Pengimplementasinya mendukung kinerja dan juga peningkatan efisiensi, efektivitas dan produktivitas bagi berbagai instansi, baik itu instansi pemerintah negeri, swasta atau bahkan perorangan, serta mendorong masyarakat untuk beradaptasi dengan kemajuan teknologi sehingga membentuk perwujudan masyarakat yang maju dan sejahtera.

Android menjadi sistem operasi yang paling banyak digunakan diantara sekian banyak pilihan sistem operasi untuk perangkat seluler. sifatnya yang terbuka untuk siapa saja, baik menggunakan atau berkontribusi dalam pengembangan aplikasi Android atau disebut dengan *open source* memudahkan pengembang untuk membuat aplikasi Android. Menurut *Statcounter* (2022), Pada bulan Oktober 2022 sebanyak 89,82% konsumen indonesia memilih *platfrom* Android untuk sistem operasi peranti *mobile* mereka, ini menandakan bahwa kebutuhan akan aplikasi berbasis android sangat tinggi[1]

Sistem Kewaspadaan Dini dan Respon adalah sebuah sistem yang memiliki fungsi penting dalam mendeteksi adanya ancaman indikasi (Kejadian Luar Biasa) KLB penyakit menular. Data yang digunakan untuk sistem tersebut harus akurat, karena hasil data yang akurat sangat penting untuk dapat membuat keputusan yang sesuai[2]

Pola arsitektur (*Architectural Pattern*) dalam pengembangan perangkat lunak seperti *Clean Architecture* (Robert C. Martin, 2017), *Onion Architecture* (Jeffrey Palermo, 2008). *Onion Architecture* diperuntukan untuk pengembangan website bersekala besar dan memiliki keberlanjutan proses bisnis dan juga memiliki prilaku yang kompleks. *Clean Architecture* mempunyai kelebihan ketika dihadapkan dengan proyek aplikasi berbasis Android dikarenakan kemudahan dalam penerapan aturan ketergantungan (*dependency rule*)[3]–[5].

Pada penelitian ini akan dilakukan perancangan dan pembangunan aplikasi berbasis Android yang dapat mengelola data untuk kebutuhan SKDR (Sistem Kewaspadaan Dini dan Respon) agar terhindar dari kesalahan perhitungan data. *Clean Architecture* dan *Dependency Injection* akan digunakan, sehingga akan menghasilkan aplikasi dengan kode yang bersih dan kokoh.

2. Tinjauan Pustaka

2.1 Pengertian Perancangan

Perancangan adalah proses mendeskripsikan, merencanakan dan mensketsa atau menyusun beberapa elemen independen menjadi satu kesatuan fungsional yang lengkap. Perancangan sistem dapat dirancang dalam bentuk diagram alir sistem (system flowchart), yaitu suatu alat grafis yang dapat digunakan untuk menampilkan proses Urutan sistem. Menurut Jogiyanto (1999)[6], Perancangan mempunyai dua tujuan, yaitu untuk memenuhi kebutuhan pengguna sistem, dan untuk memberikan gambaran yang jelas bagi pemrogram komputer dan pakar teknis terkait lainnya[6]

2.2 Pengertian Pengolahan Data

Pengolahan data dengan mengunakan komputer dikenal dengan sebutan *Pengolahan Data Elektronik* (PDE) atau *Electronic Data Processing* (EDP). Data adalah kumpulan kejadian yang diambil dari suatu fakta dan data dapat berupa angka-angka, huruf-huruf atau simbol- simbol khusus atau gabungan dari kesemuanya. Suatu data mentah adalah merupakan data yang belum dapat memberi informasi yang banyak dan lengkap, sehingga perlu diolah terlebih dahulu.[6]

2.3 Sistem Kewaspadaan Dini dan Respon

Sistem Kewaspadaan Dini dan Respon (SKDR) adalah suatu sistem yang memberikan penyadaran mengenai terjadinya proses atau merebaknya suatu wabah penyakit dan juga faktor-faktor yang mempengaruhinya dengan menerapakan teknologi pengawasan epidemiologi dan digunakan untuk meningkatkan respon, upaya dan penanggulangan wabah penyakit secara cepat dan tepat[2]

2.4 *Clean Architecture*

Dalam proses pengembagan perangkat lunak muncul beberapa arsitekturyang digunakan oleh para pengembang sebagai prinsip dalam pengembangan aplikasi[3]. Robert C. Martin (2012) mengatakan bahwa ide utama penggunaan *Clean Architecture* yaitu untuk menghasilkan sistem yang:

a. *Independent of Framework*, Tidak tergantung dengan implementasi *framewok* yang digunakan.
b. *Testable*, Kode untuk proses bisnis dapat dites tanpa perlu UI, *Database*, atau elemen eksternal lainnya.
c. *Independent of UI*, UI dapat diubah dengan mudah, tanpa perlu mengubah keseluruhan sistem.
d. *Independent of Database*, Tidak bergantung pada *framework* database tertentu dan dapat diganti degan mudah.
e. *Independent of External*, Proses bisnis yang ada tidak perlu apa yang ada diluarnya.

2.5 *Dependency Injection*

Dikutip dari (*Developers Android Documentation) Dependency Injection adalah* teknik yang banyak digunakan dalam pemprograman dan sesuai dengan pengembangan Android. Dengan mengikuti prinsip-prinsip *Dependency Injection*, Anda mengatur dasar arsitektur aplikasi yang baik[7][8].

Didalam pengembangan aplikasi Android dan perangkat lunak secara umum, beberapa permasalahan yang bersifat berulang dapat diatasi dengan menggunakan *Design Patterns*. *Dependency Injection* termasuk kedalam bagian *Creational Pattern* dalam *Design Pattern*, *Creational Pattern* digunakan supaya pengembang dapat membuat *instance* dari sebuah objek tanpa perlu mengetahui bagaimana proses pembuatannya secara detail. Sehingga dapat membuat objek secara cepat dan mudah, *Creational Pattern* mengedepankan fleksibilitas dan penggunaan kembali (*reusability*)[7], [8]

3.Analisis Dan Perancangan Sistem

3.1 Deskripsi Umum Sistem

Pada tugas akhir ini akan dikembangkan suatu aplikasi yang dapat mengelola data untuk keperluan sistem kewaspadaan dini dan respon yang berlajan pada perangkat seluler berbasis android. Dengan adanya aplikasi ini, pengguna yaitu survailans di UPTD Puskesmas Natar dapat melakukan proses pengolahan data-data penyakit mingguan dan mempermudah dalam pembuatan dan pengiriman laporan baik melalui pesan digital whatsapp atau pesan manual menggunakan sms (*short message service*) melalui fitur intent.

Aplikasi dibangun dengan menerapkan desain dari *clean architecture* yang bertujuan agar aplikasi yang dikembangkan dapat dibangun menggunakan prinsip pengkodean yang mengikuti kaidah *single Responsibillity* dan *dependency inversion* yang merupakan bagian dari paradigma SOLID. Penerapan *clean architecture* mengarah pada prinsip *single responsibillity* yaitu pemisahan konsentrasi atau tanggung jawab antar bagian pada aplikasi[7]–[9].

Sedangkan *dependency injection* digunakan untuk mengurangi terjadinya ketergantungan yang berlebih antar bagian dalam aplikasi sehingga sulit dalam proses pembangunan dan perawatan seperti proses instansiasi dan *constactor*. Untuk mempermudah proses penerapan dari dependency injection pada fungsionalitas aplikasi akan digunakan sebuah *library* bernama koin yang dapat membantu penerapan *dependency injection* menjadi lebih mudah[7], [8]

3.2 Metode Pengumpulan Data

a. Observasi

Dalam teknik ini penulis mengamati secara langung bagaimana proses pengumpulan data yang dilakukan oleh petugas surveilans UPTD Puskesmas Natar.

b. Wawancara

Merupakan jenis-jenis pengumpulan data yang berbentuk tanya jawab antara peneliti dan narasumber.

Pada penelitian ini, pengumpulan data dengan wawancara dilakukan dengan petugas surveilans sendiri

c. Studi Literatur

Teknik ini disebut juga dengan studi pustaka yaitu cara menelusuri kepustakaan yang berisi teori-teori daru karya ilmiah yang ada pada buku-buku (*e-books*), makalah, jurnal *online*

3.3 Analisa Kebutuhan Sistem

Data yang dibutuhkan dalam penelitian ini berupa panduan kriteria, jenis dan atribut dari data yang akan dikelola oleh aplikasi yang akan digunakan sebagai data dasar dari pembentukan basis data dan juga pembentukan dari *pre-populate database* yaitu suatu mekanisme ketika dilakuikan penginstalan aplikasi terhadap perangkat Android, maka sudah ada data dasar yang tersimpat pada data base, agar user tidak perlu melakukan proses memasukan data dasar secara manual. Berikut merupakan data jenis-jenis penyakit dan format laporan mingguan yang akan digunakan

KODE SMS	PENYAKIT
A	Diare Akut
B	Malaria Konfirmasi
C	Tersangka Demam Dengue
D	Pneumonia
E	Diare Berdarah ATAU Disentri
F	Tersangka Demam Tifoid
G	Sindrom Jaundis Akut
H	Tersangka Chikungunya
J	Tersangka Flu Burung pada Manusia
K	Tersangka Campak
L	Tersangka Difteri
M	Tersangka Pertussis
N	AFP (Lumpuh Layuh Mendadak)
P	Kasus Gigitan Hewan Penular Rabies
Q	Tersangka Antraks
R	Tersangka Leptospirosis
S	Tersangka Kolera
T	Klaster Penyakit yang tidak lazim
U	Tersangka Meningitis/Ensefalitis
V	Tersangka Tetanus Neonatorum
W	Tersangka Tetanus
Y	ILI (*Influenza Like Illness*)
Z	Tersangka HFMD
AC	Tersangka Covid-19
X	**TOTAL (JUMLAH KUNJUNGAN)****

Gambar 1. Data Jenis Penyakit

Untuk pembuatan laporan mingguan akan menggunakan kaidah dan aturan seperti format pembuatan laporan dan cara pengiriman laporan, sebagai berikut:

Format Penulisan SMS: MANUAL#2,A10,B15,H3,T4,X110, artinya:
Minggu epidemiologi ke 2, jumlah kasus diare= 10, jumlah kasus malaria = 15, jumlah kasus tersangka Chikungunya = 3, jumlah kasus klaster penyakit yang tidak lazim = 4, Jumlah kunjungan = 110
Kirim ke **0812-9610-0884; 0822-9800-0620; 0857-1486-8413; 0818-0681-8190**

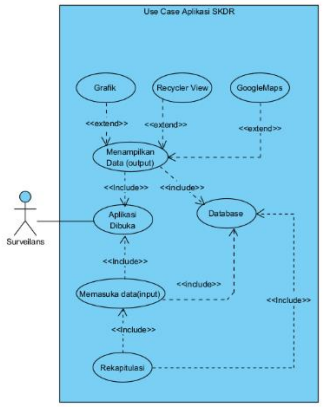
Format Penulisan WA: SKDR minggu#tahun#data laporan,total kunjungan
Contoh:
SKDR 19#2022#A0,B2,R3,AC6,X230

Gambar 2. Format Laporan Mingguan

3.4 Perancangan Sistem

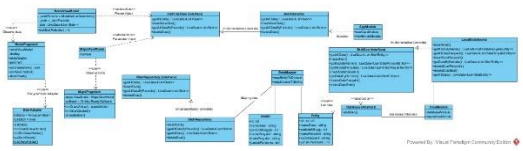
Perancangan sistem dapat memodelkan kebutuhan dengan menggambarkannya menggunakan *Unifield Modeling Language* yaitu menggunakan class diagram, use case diagram dan menggunakan class responsibillity collaborator card dalam melakukan permodelan kelas dengan menggunakan metode analisis sitem extreme programming (XP)[10], [11]

1. Use Case Diagram



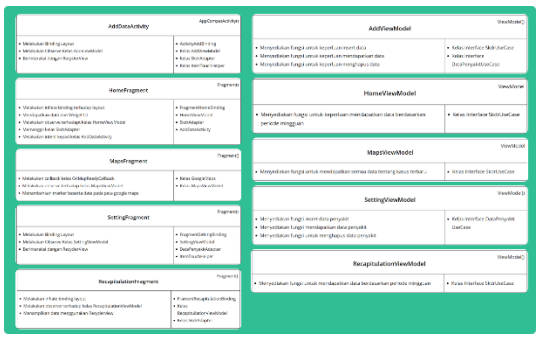
Gambar 3. Use Case Diagram

2. Class Diagram

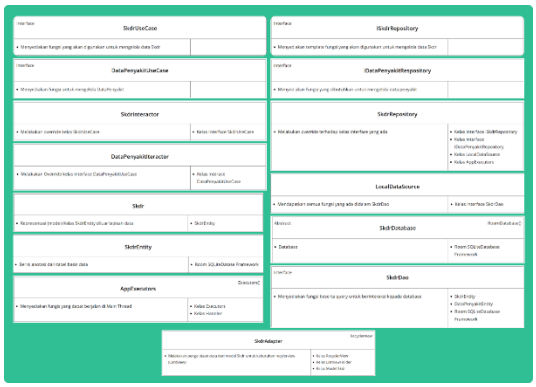


Gambar 4 Class Diagram

3. Class Diagram Collaborator Card



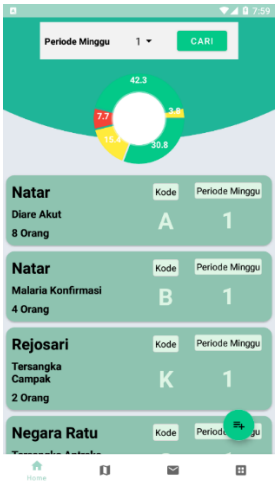
Gambar 5 CRC Card



Gambar 6 CRC Card (lanjutan)

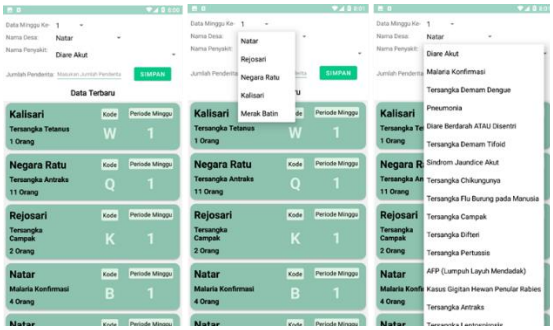
4. Implementasi

4.1 Halaman Home



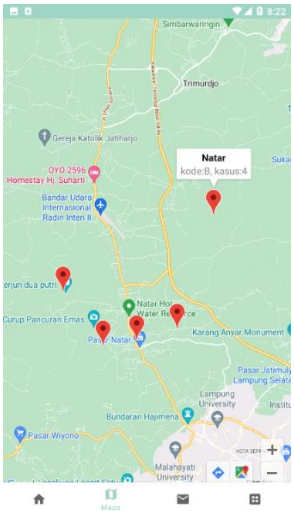
Gambar 7 Halaman Home

4.2 Tambah Data



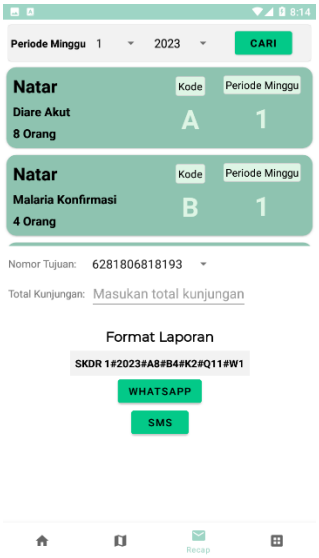
Gambar 8 Halamaan Tambah Dat

4.3 Halaman Maps



Gambar 9 Halaman Maps

4.4 Halaman Rekaputilasi

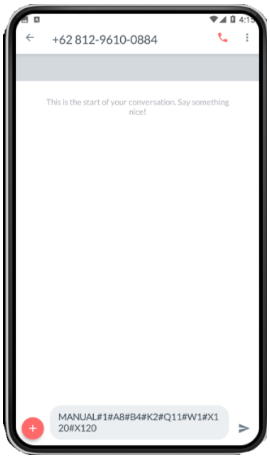


Gambar 10 Halaman Rekapitulasi

4.5 Halaman Intent WhatsApp

Gambar 11 Intent WhatsApp

4.6 Halaman Intent SMS



Gambar 12 Intent SMS

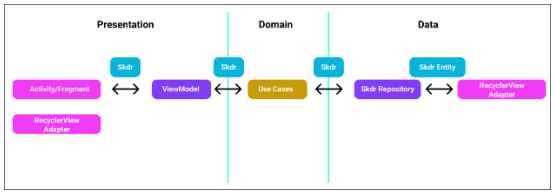
5. Hasil Penelitian dan Pembahasan

5.1 *Clean Architecture*

Penerapan arsitektur ini berfokus pada salah satu prinsip dari paradigma pemprograman SOLID yaitu *single responsibillity*, dimana pada prinsip tersebut aplikasi akan diklasifikasikan menjadi 3 (tiga) *layer* atau lapisan yaitu lapisan *Presentation*, *Domain*, dan *Data* yang memiliki pembagian tugas yang jelas, lapisan *presentation* berisi *user interface* seperti *activity* dan *fragment* juga *viewmodel*, lalu untuk lapisan *domain* berisi kelas *interface* atau abstraksi yang menghubungkan antara lapisan *presentation* dan *data*, untuk lapisan data sendiri berisi fungsionalitas yang berhubungan dengan penyimpanan data seperti *database* dan komponen yang diperlukan[3]

Gambar 13 Struktur Aplikasi

Penerapan dari *clean architecture* membuat dapat dilakukannya pemisahan model untuk data dan model untuk domain. Sehingga bagian kode yang ada pada lapisan data seperti *database* dapat dipisahkan dari bagian *presentation.* seperti yang digambarkan pada gambar dibawah ini



Gambar 14 Pemisahan model data dan model domain

Pada gambar diatas dijelaskan bahwa kelas yang barkaitan dengan *database* yaitu *SkdrEntity* tidak secara langsung dipanggil pada bagian yang membutuhkan penggunaan *database* seperti pada bagian presentation yaitu ketika menambahkan data, memasukan data, menghapus data. Melainkan dibuat kelas baru bernama Skdr yaitu merupakan kelas *model* yang berisi variabel yang sama seperti kelas SkdrEntity, yang menjadi pembeda adalah kelas Skdr bersih dari penggunaan *framework* seperti *Room* dan tidak secara langsung berelasi dengan kelas penyimpanan data yaitu SkdrDatabase, berikut merupakan perbedaan kode yang ada pada kelas Skdr dan *SkdrEntity*

Gambar 15 Kelas Skdr Entity

5.2 *Dependency Injection*

Penerapan *dependency injection* atau *dependency inversion principal* yang termasuk dalam poin terakhir pada prinsip SOLID merupakan *design pattern* yang populer dalam pemprogramaan berorientasi objek. *Dependency injection* termasuk dalam *creational pattern* yaitu suatu cara yang digunakan untuk mempermudah pembuatan suatu objek. Pada dasarnya *dependency* adalah suatu objek yang dibutuhkan untuk melakukan suatu proses. Pada penelitian ini digunakan sebuah library yang bernama koin, dengan begitu penerapan dependency injection akan menjadi lebih mudah.

Gambar 19 Dependencies gradle

Penggunakan *dependency injection* dapat di lihat dari kode instansiasi kelas *ViewModel* kedalam UI *interface* baik itu *activity* atau *fragment*. Berikut merupakan kode yang digunakan untuk menggunakan kelas *AddViewModel* pada kelas *AddDataActivity*

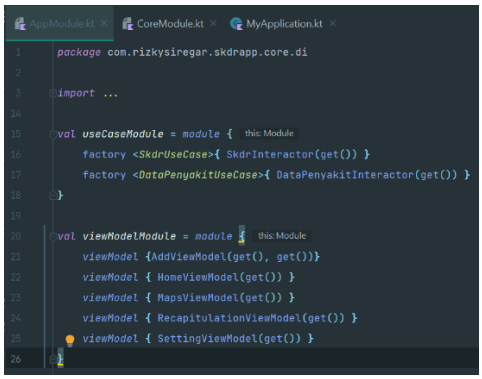
Gambar 20 Instansiasi Kelas *ViewModel*

Gambar 21 Kelas *ViewModel*

Berbeda dengan penggunaan dari *dependency injection* semua fungsionalitas yang dibutuhkan telah disiapkan pada suatu kelas yang bernama *AppModule* dan

CoreModule sehingga pembuatan kelas viewmodel menjadi jauh lebih mudah untuk dibangun. Berikut merupakan kode yang ada pada kelas *AppModule* dan *CoreModule*

Gambar 22 *CoreModule.kt*

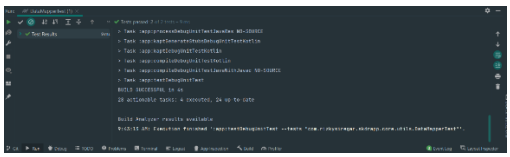


Gambar 23 *AppModule.kt*

Baik *CoreModule.kt* dan *AppModule.kt* berisi variabel kotlin yang telah dilakukan inisialisasi modul-modul (*module*) yang dibutuhkan oleh fungsionalitas yang ada pada aplikasi.

5.3 Pengujian atau Testing

Dengan menerapakan *unit testing* pada fungsionalitas aplikasi sebelum dikembangkan akan memudahkan penulisan kode pada tahap implementasi, tahap pengujian penelitian ini android unit test akan digunakan pada beberapa fungsi yang ada pada aplikasi



Gambar.26 Hasil Pengujian DataMapperTest

Gambar.28 Hasil Pengujian ValidatorTest

6. Kesimpulan dan Saran

6.1 Kesimpulan

Pembuatan aplikasi pengelola data penyakit untuk keperluan sistem kewaspadaan dini dan respon bertujuan untuk mempermudah kegiatan dari survailans di UPTD Puskesmas Natar dalam mengerjakan tugasnya untuk mengelola dan melaporakan data penyakit mingguan.

Dalam perancangan dan pengembangan aplikasi ini menggunakan pola arsitektur dari *clean architecture* dan *dependecy injection* yang merupakan bagian dari prinsip *SOLID* yang berdampak pada terciptanya aplikasi yang dibangun dengan pengkodean yang bersih (*clean code*) sehingga memberikan dampak positif bagi para pengembang/developer dalam melakukan proses perawatan dan perkembangan kedepannya.

Dengan dikembangkannya aplikasi ini diharapkan mampu memberikan efisiensi dan optimalisasi terahadap kegiatan laporan mingguan yang dilakukan oleh survailns di UPTD Puskesmas Natar agar data yang dikirimkan dapat terhindar dari kesalahan yang mungkin terjadi karena *human error.*

6.2 Saran

Penelitian tentu saja memiliki batasan-batasan dalam pengerjaannya mulai dari waktu, keterbatasan kemampuan dan lain-lain, maka dengan ini penulis menyadari masih banyak kekurangan dari penelitian yang dilakukan, maka saran untuk pengembangan dan penelitian selanjutnya adalah sebagai berikut:

1. Penerapan 3 (tiga) prinsip SOLID yang tidak diulas dengan detail pada penelitian ini yaitu: *Liskov Subtitution Principle, Interface Segregation Prinsiple dan Open Closed Priciple*
2. Menambahkan fitur lain yang tidak terbahas pada penelitian ini
3. Menerapkan modularization

4. Menggunakan desain dengan menggunakan jetpack compose tidak lagi menggunakan desain dengan bahasa XML

DAFTAR PUSTAKA

[1] Statcounter, "Mobile Operating System Market Share Indonesia," *Startcounter*, 2022. Https://Gs.Statcounter.Com/Os-Market-Share/Mobile/Indonesia (Diakses Jan 29, 2023).

[2] Nurul Haq Arma Putra, "Penerapan Sistem Kewaspadaan Dini Dan Respon Berbasis Puskesmas Di Kabupaten Barru," 2018. Diakses: Jan 29, 2023. [Daring]. Available: Http://Eprints.Stialanmakassar.Ac.Id/Id/Eprint/33/1/3.%20NURULHAQ%20TESIS.Pdf

[3] Robert C. Martin (Uncle Bob), "The Clean Code Blog," Hlm. 1–1, 2012, Diakses: Jan 29, 2023. [Daring]. Available: Https://Blog.Cleancoder.Com/Uncle-Bob/2012/08/13/The-Clean-Architecture.Html

[4] D. Andrian Dwijaya, "Perancangan Aplikasi Untuk Pelanggaran Dan Prestasi Siswa Pada Smp Kartika Ii-2 Bandar Lampung," *Jurnal Informatika Dan Rekayasa Perangkat Lunak (JATIKA)*, Vol. 1, No. 2, Hlm. 127–136, 2020, [Daring]. Available: Http://Jim.Teknokrat.Ac.Id/Index.Php/Informatika

[5] Developer Android Documentation, "Guide To App Architecture," *Developer Android*. 2023. Diakses: Jan 28, 2023. [Daring]. Available: Https://Developer.Android.Com/Topic/Architecture

[6] E. Amalia Dan Y. Supriatna, "Perancangan Sistem Informasi Administrasi Kependudukan Sebagai Pengembangan Egovernment," *Prosiding Seminar Ilmu Komputer Dan Teknologi Informasi*, Vol. 2, No. 1, 2017.

[7] Developer Android Documentation, "Dependency Injection In Android," *Dependency Injection In Android*, 2023, Diakses: Jan 29, 2023. [Daring]. Available: Https://Developer.Android.Com/Training/Dependency-Injection

[8] Koin Dan Kotzilla, "The Pragmatic Kotlin Dependency Injection Framework," 2023, Diakses: Jan 29, 2023. [Daring]. Available: Https://Insert-Koin.Io/

[9] Katarina Dan Apriliansyah, "Analisis Dependency Injection Dan Model-View," 2022, Diakses: Jan 30, 2023. [Daring]. Available: Http://Ojs.Stmik-Banjarbaru.Ac.Id/Index.Php/Progresif/Article/Download/781/520

[10] R. D. Gunawan, R. Napianto, R. I. Borman, Dan I. Hanifah, "Penerapan Pengembangan Sistem Extreme Programming Pada Aplikasi Pencarian Dokter Spesialis Di Bandar Lampung Berbasis Android," 2019.

[11] M. Al, K. Rizki, Dan A. F. Op, "Rancang Bangun Aplikasi E-Cuti Pegawai Berbasis Website (Studi Kasus : Pengadilan Tata Usaha Negara)," *Jurnal Teknologi Dan Sistem Informasi (JTSI)*, Vol. 2, No. 3, Hlm. 1–13, 2021, [Daring]. Available: Http://Jim.Teknokrat.Ac.Id/Index.Php/JTSI