BAB 3
METODOLOGI PENELITIAN

Metodologi penelitian yang digunakan adalah metode *waterfall* dari Herbert D. Benington. Metode *waterfall* merupakan metode pengembangan yang menekankan pada langkah-langkah sistematis, sehingga proses penciptaan sistemnya harus dilakukan secara berurutan. Metode *waterfall* terdiri dari lima tahapan dan dijabarkan sebagai berikut [9].

3.1 *Requirement Analysis*

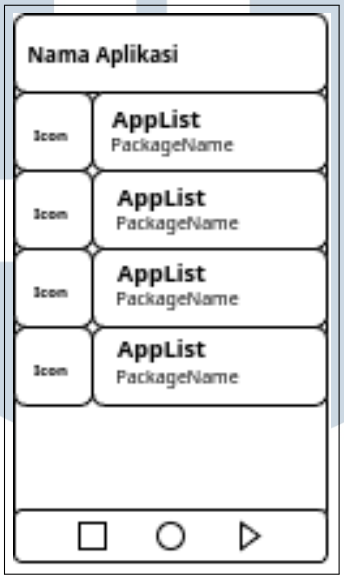
Tahap pertama yang dilakukan adalah menganalisis kebutuhan dari pengguna sistem dengan tujuan dapat memberikan batasan dari *software* yang ingin dikembangkan, sekaligus mengetahui kebutuhan dari pengguna. Metode ini dapat dilakukan melalui metode penelitian kualitatif maupun kuantitatif, menyesuaikan dengan permasalahan yang hendak diselesaikan.

Gambar 3.1. presentasi total data yang terbobol berdasarkan survey dari SoC Radar tahun 2021

hasil analisa yang dilakukan adalah menemukan bahwa terdapat 270 juta warga yang mengalami kebocoran data. tidak lepas juga ancaman Phising yang meningkat dengan 20000 serangan yang terdeteksi menargetkan e-commerce dan SaaS[20].

3.2 *Design*

Tahap kedua dari metode *waterfall* adalah tahap perancangan dan pengembangan. Tahapan ini didasari oleh kebutuhan pengguna, sehingga dapat mempermudah proses pengerjaan sekaligus memperoleh gambaran lebih detail terkait tampilan sistem. Selain itu, tahapan ini juga menggunakan tahapan desain, yaitu merancang desain perangkat lunak dengan tujuan memberi pemahaman mendalam bagi *programmer* terkait tampilan dan antarmuka perangkat lunak.



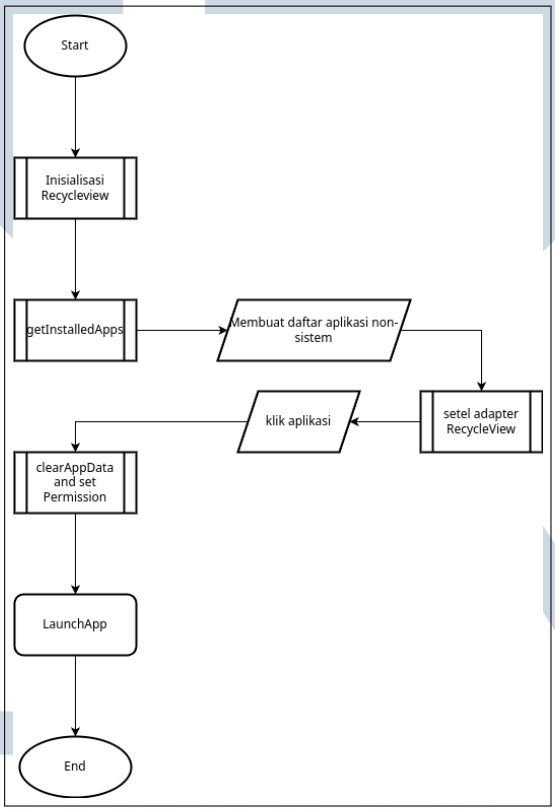
Gambar 3.2. WireFrame Aplikasi Sandboxing

Desain aplikasi ini terdiri dari dua komponen utama, ‘MainActivity‘ dan ‘AppMonitorService‘, yang bekerja bersama untuk menampilkan daftar aplikasi non-sistem yang terinstal dan meluncurkan aplikasi yang dipilih dalam lingkungan terisolasi. ‘MainActivity‘ inisialisasi layout dan RecyclerView untuk menampilkan aplikasi yang diambil dengan memfilter aplikasi sistem menggunakan ‘PackageManager‘. Saat aplikasi dipilih, ‘clearAppDataAndLaunch‘ diaktifkan untuk menghapus data aplikasi melalui perintah shell ‘pm clear‘ dan kemudian meluncurkan aplikasi menggunakan ‘startActivity‘.

Setelah aplikasi diluncurkan, ‘AppMonitorService‘ dimulai untuk memantau apakah aplikasi tersebut masih berjalan. ‘AppMonitorService‘ secara periodik mengecek status aplikasi, dan jika aplikasi berhenti berjalan, data aplikasi akan dihapus kembali menggunakan perintah shell sebelum layanan dihentikan. Dengan cara ini, aplikasi memastikan bahwa setiap kali aplikasi target dijalankan,

ia selalu dimulai dalam keadaan bersih tanpa data tersimpan sebelumnya.

Pada tahap ini, dibuatlah *wireframe* berupa menu yang berisikan daftar aplikasi yang diunduh ke dalam Android melalui *Play Store* atau pun tempat pengunduhan lainnya, beserta dengan nama *package*-nya. *wireframe* ini dibuat dengan tujuan memberikan arahan terkait penempatan fitur yang dapat diakses melalui Android. Ikon dari setiap aplikasi yang diunduh ditampilkan pada bagian kiri dengan tujuan mempermudah pengguna dalam menemukan aplikasi yang sedang dicari dalam wujud visual yang dikenali, sedangkan daftar aplikasi beserta dengan nama *package*-nya diletakkan di sebelah kanan ikon dengan tujuan memberikan penjelasan terkait aplikasi yang terdaftar.



Gambar 3.3. Desain flowchart MainActivity

Wireframe yang dibuat menyesuaikan pada pembuatan *flowchart* berupa alur navigasi penggunaan aplikasi *Sandbox* tersebut. Berdasarkan *flowchart* yang dibuat, alur navigasi dimulai dari tampilan daftar aplikasi yang diunduh ke dalam Android, baik yang diunduh secara langsung oleh pengguna maupun aplikasi yang diunduh secara tidak langsung atau dibawa dari aplikasi lain secara otomatis.

3.3 *Implementation and Unit Testing*

Tahap ketiga dari metode ini mencakup proses implementasi *Package Manager* yang memungkinkan aplikasi Sandboxing daapat mengatur *permission* yang ada pada aplikasi target. Proses tersebut dapat dijelaskan dalam *coding* sebagai berikut.

3.3.1 MainActivity.Kt

Berikut adalah implementasi *MainActivity.Kt* dalam aplikasi yang telah dibuat.

```
1
2     Declare recyclerView
3     Declare appsAdapter
4
5     Method onCreate(bundle)
6         Set up main activity layout
7         Initialize recyclerView
8         Get list of installed apps
9         Set up appsAdapter with the list
10        Assign appsAdapter to recyclerView
11    End Method
12
13    Method getInstalledApps returns list
14        Get package manager
15        Get list of all installed apps
16        Create empty list for non-system apps
17
18        For each app in the list
19            If the app is not a system app
20                Add app info (name, package, icon) to the list
21            End If
22        End For
23
24        Return list of non-system apps
25    End Method
26
27    Method isSystemApp(app) returns boolean
28        Check if app is a system app using app flags
29    End Method
30
31    Method launchAppInIsolatedEnvironment(packageName)
```

```
        Get package manager
        Get launch intent for the app

        If launch intent exists
            Start the app
            Start service to monitor the app
        Else
            Show error message
        End If
    End Method

End Class
```

Kode 3.1: MainActivity.Kt

pada bagian *MainActivity* terdapat fungsi sebagai berikut:

1. onCreate: Menginisialisasi tampilan dan RecyclerView untuk menampilkan daftar aplikasi.
2. getInstalledApps: Mendapatkan daftar aplikasi yang terinstal, mengabaikan aplikasi sistem.
3. isSystemApp: Memeriksa apakah aplikasi adalah aplikasi sistem.
4. clearAppDataAndLaunch: Membersihkan data aplikasi dan meluncurkannya.
5. clearAppData: Menggunakan perintah shell untuk membersihkan data aplikasi.
6. launchApp: Meluncurkan aplikasi yang dipilih dan memulai AppMonitorService untuk memantau aplikasi.

3.3.2 AppMonitorService.Kt

Berikut adalah implementasi *AppMonitorService.Kt* dalam aplikasi yang telah dibuat.

```
Class AppMonitorService extends Service

    Declare packageName
    Declare handler
    Set checkInterval to 2000 milliseconds

```

```
    Method onStartCommand(intent, flags, startId) returns Int
        Get package name from intent
        If package name is null, return START_NOT_STICKY

        Start checking if the app is running
        Return START_STICKY
    End Method

    Runnable checkAppRunning
        Method run
            If app is not running
                Clear app data
                Stop the service
            Else
                Check again after checkInterval
            End If
        End Method
    End Runnable

    Method isAppRunning(packageName) returns Boolean
        Get activity manager
        Get list of running app processes
        Check if any process name matches package name
    End Method

    Method clearAppData(packageName)
        Try
            Run command to clear app data
            Log success message
        Catch exception
            Log error message
        End Try
    End Method

    Method onBind(intent) returns IBinder
        Return null
    End Method

End Class
```

Kode 3.2: AppMonitorService.Kt

pada bagian *AppMonitorService* terdapat fungsi sebagai berikut:

1. onStartCommand: Mengambil nama paket aplikasi yang akan dipantau dan

memulai tugas pengecekan.

2. checkAppRunning: Runnable yang memeriksa apakah aplikasi masih berjalan. Jika tidak, data aplikasi akan dihapus dan layanan dihentikan.
3. isAppRunning: Memeriksa apakah aplikasi dengan packageName masih berjalan menggunakan ActivityManager.
4. clearAppData: Menggunakan perintah shell untuk menghapus data aplikasi setelah aplikasi ditutup.
5. onBind: Mengembalikan null karena layanan ini tidak mengikat.

3.3.3 Cara Kerja

Berikut adalah deskripsi dari komponen utama dan interaksi aplikasi dan target aplikasi:

MainActivity:

1. Menampilkan daftar aplikasi yang terinstal.
2. Menghapus data aplikasi dan meluncurkan aplikasi yang dipilih.
3. Memulai `AppMonitorService` untuk memantau aplikasi yang diluncurkan.

AppMonitorService:

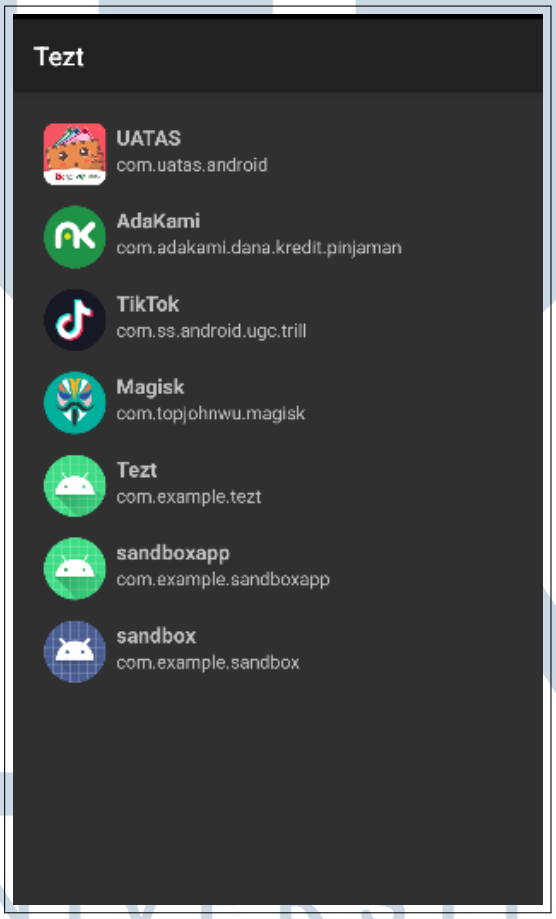
1. Memantau apakah aplikasi yang diluncurkan masih berjalan.
2. Menghapus data aplikasi saat aplikasi ditutup.

Catatan:

1. **Root Access:** Pastikan perangkat telah di-root untuk mengeksekusi perintah `su`.
2. **Permissions:** Aplikasi memerlukan izin untuk mengakses data aplikasi lain dan menghapusnya.
3. **Monitoring:** Layanan memeriksa status aplikasi setiap 2 detik dan menghapus data saat aplikasi ditutup.

3.4 *Integration and System Testing*

Tahap keempat mencakup proses pengintegrasian hasil dari *coding* yang dibuat untuk mewujudkan aplikasi yang mampu menjawab permasalahan dari penggunanya. Aplikasi yang digunakan untuk *testing* adalah aplikasi UATAS, Tiktok, dan KSP Dompet Gajah. UATAS merupakan aplikasi Pinjaman *Online* yang tidak diunduh melalui *play store*, melainkan melalui sebuah *website* resmi. TikTok adalah media sosial yang diunduh secara langsung dari *Google play store*. Sedangkan, aplikasi KSP Dompet Gajah diunduh melalui *website* tidak resmi.



Gambar 3.4. Tampilan Menu Aplikasi

Uji coba yang dilakukan pada tahap ini menggunakan metode pengujian fungsional dari OWASP *Mobile Security Testing Guide* dengan tujuan mengevaluasi keefektifan penggunaan *Sandbox* dalam membatasi akses aplikasi tertentu terhadap fitur dan data pada Android, sesuai dengan batasan yang telah dibuat. Pengujian yang dilakukan melalui metode tersebut mencakup pengujian izin aplikasi, akses

penyimpanan, dan respon terhadap pembatasan. Pengujian yang telah dilakukan dijabarkan sebagai berikut.

1. Pengujian Izin Aplikasi
   Pengujian ini dilakukan pada aplikasi UATAS, TikTok, dan KSP Dompet Gajah. Tujuannya adalah untuk memastikan bahwa ketiga aplikasi tersebut tidak dapat mengakses data ataupun fitur lain yang telah dibatasi melalui *Sandbox*.Pengujian ini dilakukan dengan cara menjalankan aplikasi yang mencoba meminta akses terkait lokasi, kontak, dan kamera dari perangkat yang digunakan. Aplikasi tersebut dibuka melalui pembatasan *Sandbox*, lalu dipantau melalui LogCat yang ada.

2. Pengujian Akses Penyimpanan
   Pengujian ini dilakukan pada ketiga aplikasi sebelumnya dengan tujuan memastikan bahwa ketiga aplikasi tersebut tidak dapat menelurusi isi penyimpanan internal maupun eksternal melebihi dari batasan yang telah ditetapkan dalam *Sandbox*. Pengujian ini dilakukan dengan menjalankan ketiga aplikasi tersebut yang mencoba membuat dan mengakses *file* dalam penyimpanan internal dan eksternal. Pengecekan dilakukan melalui *command* berikut.

```
internal: adb shell touch /storage/emulated/0/Android/data/(
    com.example.target)/files/test_internal.txt
external: adb shell touch /sdcard/test_external.txt

```

3. Pengujian Respon Terhadap Pembatasan
   Pengujian ini dilakukan pada tiga aplikasi yang sama dengan sebelumnya, yaitu UATAS, TikTok, dan KSP Dompet Gajah, dengan tujuan mengecek respon dari ketiga aplikasi tersebut saat mencoba melakukan tindakan yang telah dibatasi oleh *Sandbox*. Setelqah melakukan pengecekan terhadap aplikasi, dilakukan pula pemantauan log sistem terkait respon aplikasi terhadap penolakan izin yang diterapkan. Pengecekan dilakukan melalui *command* berikut.

```
Penolakan Izin: adb shell am start -a android.intent.action.
    VIEW -d geo:0,0
Penolakan Akses Penyimpanan: adb shell touch /storage/
    emulated/0/Android/data/com.example.target/files/denied.
    txt
```

Berdasarkan hasil uji coba tersebut, ditemukan bahwa aplikasi TikTok dan KSP Dompet Gajah tetap dapat dijalankan setelah diisolasi oleh *Sandbox*, sedangkan aplikasi UATAS tidak dapat diisolasi sama sekali sehingga aplikasi langsung melakukan *force close*.

3.5 *Maintenance*

Tahap *maintenance* merupakan tahap terakhir dari metode *waterfall*, setelah sistem didistribusikan dan digunakan oleh pengguna. Tahapan ini dilakukan untuk memastikan bahwa sistem dapat berjalan sesuai dengan fungsinya melalui tahap perbaikan terkait kesalahan yang terjadi pada sistem, sehingga kinerja sistem dapat meningkat menyesuaikan dengan kebutuhan penggunanya. Aplikasi *Sandbox* yang telah dibuat, ditambahkan dengan beberapa fitur baru, seperti fitur *search* dan tema.

Gambar 3.5. Tampilan searchbar dalam aplikasi *Sandbox*

Penambahan fitur dilakukan dengan tujuan mempermudah pengguna untuk menemukan unduhan aplikasi yang ingin dicek dengan lebih cepat, sekaligus memperindah tampilan dengan adanya tambahan tema dalam aplikasi sehingga pengguna menjadi lebih tertarik untuk menggunakannya.

UNIVERSITAS MULTIMEDIA NUSANTARA