MAKALAH TEORI PEMROGRAMAN MOBILE

"ANDROID LIFE CYCLE, KARAKTER KOTLIN, FLUTTER, DAN PERBANDINGAN SISTEM OPERASI MOBILE"

SEMESTER V

Dosen Pengampu :

Rika Idmayanti, S.T., M.Kom.

Disusun Oleh :

Jesica Sanditia Putri 2111083014

PROGRAM STUDI TEKNOLOGI REKAYASA PERANGKAT LUNAK

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI PADANG

2023

KATA PENGANTAR

Puji syukur kehadirat Allah Ta'ala atas segala nikmat dan karunia Nya yang selalu dilimpahkan kepada kita semua. Atas berkat_Nya saya dapat membuat makalah ini untuk memenuhi tugas mata kuliah Teori Pemrograman Mobile, dengan judul: "Android Life Cycle, Karakter Kotlin, Flutter, dan Perbandingan Sistem Operasi Mobile" .

Penulis menyadari bahwa penyusunan makalah ini masih jauh dari kesempurnaan, dan dengan kerendahan hati, Penulis mengharapkan saran dan kritik yang bersifat membangun dari pembaca agar bisa membuat karya makalah yang lebih baik pada kesempatan berikutnya

Dengan rendah hati, Penulis juga ingin mengungkapkan terima kasih atas dukungan, bimbingan, dan inspirasi yang diberikan oleh dosen dan teman-teman selama proses penulisan makalah ini. Semoga makalah ini dapat memberikan wawasan yang berguna dan mendalam tentang siklus hidup Android, karakter Kotlin, dan Flutter, serta perbandingan sistem operasi mobile. Di tengah perkembangan teknologi yang pesat, pemahaman yang mendalam terhadap aspek-aspek ini diharapkan dapat menjadi kontribusi bermakna dalam pemahaman lebih lanjut tentang ekosistem mobile, khususnya di Indonesia.

Padang, 17 November 2023

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perangkat seluler telah menjadi bagian yang penting dari kehidupan sehari-hari di era digital yang semakin maju. Platform seluler seperti Android, iOS, dan Windows Mobile telah berkembang secara signifikan dan menawarkan beragam fitur yang memengaruhi cara kita berinteraksi dengan teknologi. Selain itu, pemahaman mendalam tentang siklus hidup aplikasi Android, karakteristik Kotlin dan Flutter sebagai bahasa pemrograman seluler, dan perbandingan antara sistem operasi seluler penting untuk mengoptimalkan programmeran aplikasi dan memahami kebutuhan pengguna.

Siklus hidup aplikasi Android menjelaskan bagaimana aplikasi berkembang mulai dari pembuatan hingga penghentian, dengan berbagai fase yang memengaruhi daya tanggap dan kinerja aplikasi. Pemahaman yang baik tentang siklus hidup Android adalah landasan paling penting bagi programmer untuk mengoptimalkan aplikasi, meningkatkan efisiensi, dan memberikan pengalaman pengguna yang lancar.

Kotlin dan Flutter menjadi alternatif menarik untuk programmeran aplikasi seluler. Kedua bahasa pemrograman ini menawarkan fitur unik seperti variabel, kelas, dan fungsi yang memengaruhi cara programmer merancang dan mengimplementasikan solusi. Memahami perbedaan dan keunggulan masing-masing bahasa membantu programmer membuat keputusan yang tepat berdasarkan kebutuhan proyek mereka.

Sistem operasi seluler seperti Android, iOS, Windows Mobile, Symbian, dan BlackBerry memiliki fitur dan spesifikasi yang berbeda. Analisis mendetail tentang perbandingan antara sistem operasi ini memberikan wawasan tentang kekuatan dan kelemahan masing-masing sistem operasi dan membantu programmer serta pemangku kepentingan memutuskan platform terbaik untuk tujuan aplikasi yang dibuat.

Dengan memahami kompleksitas siklus hidup aplikasi Android, karakteristik Kotlin dan Flutter, serta perbandingan sistem operasi seluler, kami berharap penelitian ini akan berkontribusi secara signifikan dalam memenuhi kebutuhan aplikasi seluler yang selalu dinamis dalam programmeran aplikasi mobile di era digital ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka adapun rumusan masalah yang diangkat oleh penulis yaitu :

1. Bagaimana siklus hidup aplikasi Android?
2. Apa saja tahapan-tahapan dalam siklus hidup aplikasi Android?
3. Bagaimana variabel, kelas, dan fungsi diimplementasikan dalam bahasa pemrograman Kotlin dan Flutter?
4. Bagaimana karakteristik Android, iOS, Windows Mobile, Symbian, dan BlackBerry?
5. Apa spesifikasi utama dari masing-masing sistem operasi mobile?

1.3 Tujuan

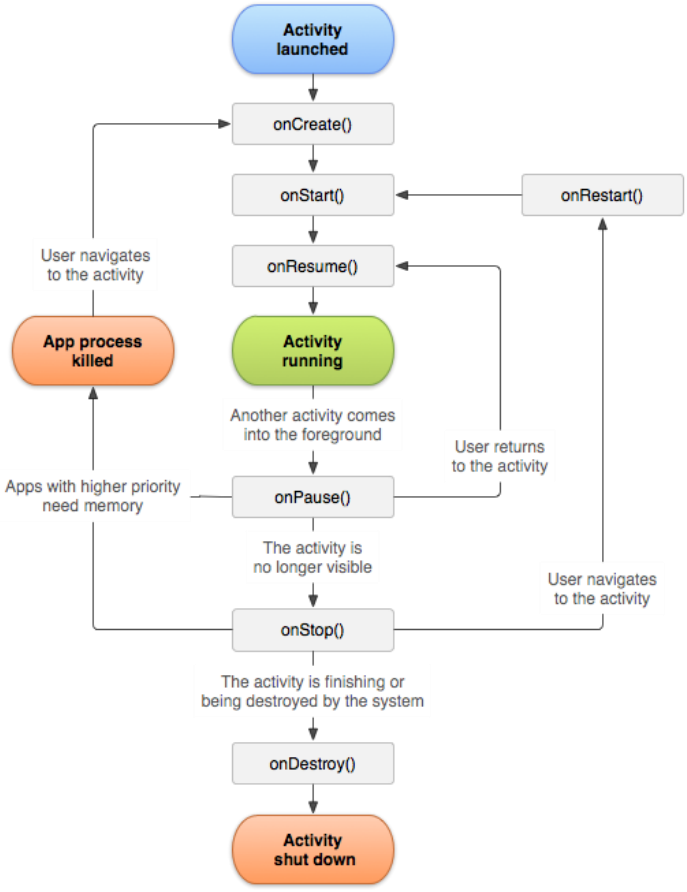
1. Menjelaskan tahapan-tahapan siklus hidup aplikasi Android dengan rinci.
2. Menganalisis dampak siklus hidup terhadap respons dan kinerja aplikasi.
3. Menyajikan gambaran tentang penggunaan variabel, kelas, dan fungsi dalam bahasa pemrograman Kotlin.
4. Menyajikan gambaran tentang penggunaan variabel, kelas, dan fungsi dalam Flutter.
5. Mendeskripsikan karakteristik utama dari sistem operasi Android, iOS, Windows Mobile, Symbian, dan BlackBerry.
6. Mendeskripsikan spesifikasi teknis yang membedakan masing-masing sistem operasi mobile.

BAB II

PEMBAHASAN

2.1 Android Life Cycle

Android Life Cycle adalah serangkaian metode yang dipanggil oleh sistem Android saat suatu aktivitas berpindah dari satu keadaan ke keadaan lainnya. Siklus hidup ini terdiri dari beberapa metode yang memungkinkan programmer untuk mengelola aktivitas, seperti inisialisasi, memulai, menjeda, dan menghentikan aktivitas sehingga programmer dapat mengetahui di metode mana suatu kode perlu diletakkan.



Gambar 1 Ilustrasi sederhana dari siklus proses aktivitas.

Beberapa metode penting / callback dalam siklus hidup aktivitas Android adalah:

1. onCreate()

Dipanggil saat aktivitas pertama kali dibuat namun layoutnya belum terlihat oleh pengguna. Pada pembuatan aktivitas, aktivitas memasuki status Dibuat. metode onCreate() menunjukkan penyiapan dasar untuk aktivitas, seperti mendeklarasikan antarmuka pengguna (didefinisikan dalam file tata letak XML), mendefinisikan variabel anggota, dan mengonfigurasi beberapa UI. Dalam metode onCreate(). Setelah metode onCreate() menyelesaikan eksekusi, aktivitas memasuki status Dimulai, dan sistem memanggil metode onStart() dan onResume() dalam urutan cepat. Bagian selanjutnya menjelaskan callback onStart().

```
package com.jesica.bahanmakalahutsjesica

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Gambar 2 Contoh onCreate()

2. onStart()

   Dipanggil setelah onCreate(), callback onStart() membuat aktivitas terlihat oleh pengguna, namun pengguna belum bisa melakukan interaksi. Metode ini dipanggil ketika aktivitas mulai terlihat oleh pengguna. Misalnya, metode ini adalah tempat aplikasi menginisialisasi kode yang mengelola UI. Saat aktivitas berpindah ke status dimulai, komponen berbasis siklus proses apa pun yang terkait dengan siklus proses aktivitas akan menerima peristiwa ON_START.

```
class MainActivity : AppCompatActivity() {
    lateinit var myTextView: TextView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        myTextView = findViewById(R.id.txtView)
    }
    override fun onStart() {
        super.onStart()

        // Inisialisasi kode yang mengelola UI pada metode onStart()
        updateUI("Aktivitas sedang dimulai.")
    }

    private fun updateUI(message: String) {
        // Mengupdate TextView dengan pesan tertentu
        myTextView.text = message
    }
}
```

Gambar 3 Contoh onStart()

3. onResume()

   Dipanggil saat aktivitas yang terlihat mulai kembali berinteraksi dengan pengguna. Contohnya saat user membuka aplikasi Instagram kemudian user membuka aplikasi Whatsapp setelah itu Kembali lagi membuka Instagram, maka Instagram yang dibuka Kembali itu mengalami onResume(). Jadi Saat aktivitas berpindah ke status dilanjutkan, komponen berbasis siklus proses apa pun yang terkait dengan siklus proses aktivitas akan menerima peristiwa ON_RESUME. Di sinilah komponen siklus proses dapat mengaktifkan fungsi apa pun yang perlu dijalankan saat komponen terlihat dan berada di latar depan, seperti memulai pratinjau kamera. Jika terjadi suatu peristiwa interupsi, aktivitas memasuki status Dijeda, dan sistem memanggil callback onPause().

```
class CameraComponent : LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    fun initializeCamera() {
        if (camera == null) {
            getCamera()
        }
    }

    ...
}
```

Gambar 4 Contoh komponen berbasis siklus proses yang mengakses kamera ketika komponen menerima peristiwa ON_RESUME

4. onPause()

   Dipanggil saat aktivitas tidak lagi dalam fokus tetapi masih terlihat oleh pengguna. Keadaan onPause() ketika activity diberhentikan sementara sehingga tidak terlihat sebagian tapi di sistem dia masih ada. Contohnya saat user membuka aplikasi Instagram kemudian user membuka aplikasi Whatsapp maka Instagram dalam status onPause(). Pada Android 7.0 (API level 24) atau lebih tinggi, beberapa aplikasi berjalan dalam mode multi-jendela. Karena hanya satu aplikasi (jendela) yang memiliki fokus di setiap waktu, sistem menjeda semua aplikasi lain. Jadi komponen siklus proses dapat menghentikan semua fungsi yang tidak perlu dijalankan saat komponen tidak ada di latar depan, seperti menghentikan pratinjau kamera.

```
class CameraComponent : LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
    fun releaseCamera() {
        camera?.release()
        camera = null
    }

    ...
}
```

Gambar 5 sumber daya yang digunakan oleh suatu komponen (seperti kamera) dilepaskan ketika komponen tersebut tidak lagi aktif atau tidak terlihat oleh pengguna.

5. onStop()

   Dipanggil saat aktivitas tidak lagi terlihat oleh pengguna dan activity disembunyikan oleh sistem. Metode onStop() juga dipanggil jika operasi terlalu berat untuk onPause(). Contohnya saat aplikasi di close untuk membuka aplikasi lain.

```
override fun onStop() {
    super.onStop()
    // Tindakan yang perlu dilakukan ketika aktivitas tidak lagi terlihat oleh pengguna

    // Contoh: Menyimpan data atau melakukan pelepasan sumber daya
    saveDataToDatabase()
}
```

Gambar 6 Contoh data disimpan ke database terlebih dahulu sebelum aktivitas dihentikan

6. onRestart()

   Metode ini dipanggil ketika sebuah aktivitas dilanjutkan setelah berhenti (setelah metode onStop() dipanggil) dan sebelum metode onStart() dipanggil kembali. Siklus hidup aktivitas biasanya melibatkan urutan seperti berikut (seperti gambar 1):

   1) onCreate(): Dipanggil saat aktivitas pertama kali dibuat.
   2) onStart(): Dipanggil ketika aktivitas terlihat oleh pengguna.
   3) onResume(): Dipanggil ketika aktivitas dapat mulai berinteraksi dengan pengguna.
   4) onPause(): Dipanggil ketika aktivitas kehilangan fokus, tetapi masih terlihat.
   5) onStop(): Dipanggil ketika aktivitas tidak lagi terlihat oleh pengguna.
   6) onRestart(): Dipanggil ketika aktivitas akan dilanjutkan setelah berhenti.
   7) onStart(): Dipanggil ketika aktivitas terlihat lagi setelah dilanjutkan.
   8) onResume(): Dipanggil ketika aktivitas dapat mulai berinteraksi dengan pengguna lagi.

7. onDestroy()

   Dipanggil saat aktivitas dihentikan atau dimusnahkan oleh sistem karena suatu aktivitas tidak lagi diperlukan jadi di tahap ini aplikasi sudah mencapai finish dan hilang dari memori. Metode onDestroy()

memberikan kesempatan terakhir untuk melakukan pembersihan atau pelepasan sumber daya sebelum aktivitas dihapus dari memori.

Perubahan keadaan state dipicu oleh Tindakan pengguna, perubahan konfigurasi atau Tindakan sistem. Kemudian dengan memahami Life Cycle memungkinkan programmer untuk mengelola sumber daya dan status aktivitas dengan tepat, serta memberikan pengalaman pengguna yang mulus dan responsif.

2.2 Kotlin

Kotlin merupakan sebuah bahasa pemrograman yang bisa ditargetkan untuk berbagai macam platform (Multiplatform) dan juga memiliki beberapa paradigma (Multiparadigm). Multiplatform berarti mendukung lebih dari 1 (satu) platform. Kemudian aplikasi multiplatform, bisa dikatakan aplikasi tersebut tersedia pada lebih dari 1 (satu) platform atau bahkan bermacam-macam platform. Programming paradigm adalah sebuah cara untuk mengklasifikasikan bahasa pemrograman berdasarkan fitur yang dimilikinya. Paradigma disini berkaitan dengan bagaimana kode dalam sebuah bahasa pemrograman diatur, seperti mengelompokkan kode atau memodifikasinya.

A. Variabel

Umumnya variabel digunakan untuk menyimpan informasi atau nilai yang akan dikelola di dalam sebuah program. Sebuah variabel dalam kotlin membutuhkan kata kunci var atau val, identifier, type dan initialization.

Sebuah variabel dapat memiliki nama yang singkat (seperti x dan y) atau nama yang lebih deskriptif (usia, jumlah, totalVolume). Aturan umum untuk variabel Kotlin adalah:

1) Nama dapat berisi huruf, angka, garis bawah, dan tanda dolar.
2) Nama harus dimulai dengan huruf.

3) Nama juga dapat dimulai dengan $ dan _
4) Nama variable bersifat case sensitive ("myVar" dan "myvar" adalah variabel yang berbeda).
5) Nama harus dimulai dengan huruf kecil dan tidak boleh mengandung spasi.
6) Kata-kata yang dicadangkan (seperti kata kunci Kotlin, seperti var atau String) tidak dapat digunakan sebagai nama variable.

1. Variabel Mutable (Var)

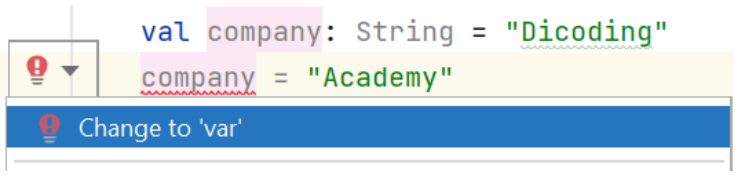
   Kata kunci var digunakan untuk variable yang dapat diubah Kembali nilai yang sudah di inisialisasikan, contoh :

```
var company: String = "Perusahaan"
company = "Academy"
```

   Variabel company yang awalnya memiliki nilai "Perusahaan" sekarang sudah diubah menjadi "Academy".

2. Variabel Immutable (Val)

   Dengan kata kunci val, programmer tidak bisa mengubah nilai yang sebelumnya sudah diinisialisasi. Jika memaksa untuk mengubahnya, maka akan terjadi error.



   Kata kunci val berguna ketika ingin membuat sebuah variabel yang selalu menyimpan nilai yang sama, seperti PI (3.14159...):

3. Inferensi Tipe (Type Inference):

   Kotlin mendukung inferensi tipe, yang berarti tidak selalu perlu menyebutkan tipe data secara eksplisit.

```
var nama = "Jesica"
var umur = 21

println("Nama: $nama, Umur: $umur")
```

4. Variabel Nullable

   Merupakan variabel yang dapat berisi nilai null dengan menambahkan tanda tanya (?) setelah tipe data.

```
var nama: String? = null
var umur: Int? = null

println("Nama: $nama, Umur: $umur")
```

5. Late-Initialized (var dengan lateinit)

   Untuk variabel yang diinisialisasi nanti (late-initialized) dengan menggunakan kata kunci lateinit.

```
lateinit var nama: String
// Inisialisasi variabel nanti
nama = "John"

println("Nama: $nama")
```

B. Class

Gunakan kata kunci ‘class’ dan tentukan nama kelas dengan diawali huruf besar(disarankan). Contoh : membuat class Car dan di dalamnya terdapat property atau variable yang ada di dalam class.

```
class Car {
    var brand = ""
    var model = ""
    var year = 0
}
```

C. Fungsi

Fungsi adalah blok kode yang hanya berjalan ketika dipanggil. Programmer dapat melakukan pass data, yang dikenal sebagai parameter, ke dalam fungsi. Fungsi digunakan untuk melakukan tindakan tertentu, dan juga dikenal sebagai metode.

1. Membuat fungsi

   Untuk membuat fungsi, gunakan kata kunci 'fun', dan tuliskan nama fungsi, diikuti dengan tanda kurung (). Contoh :

```
fun tampilkan(){
    println("--- Mobil ---")
    println("Brand: " + brand)
    println("Model Mobil: " + model)
    println("Tahun: " + year)
}
```

2. Memanggil Fungsi

   Untuk memanggil sebuah fungsi di Kotlin, tuliskan nama fungsi tersebut diikuti dengan dua tanda kurung (). Pada contoh berikut, judul() akan mencetak teks (aksi), ketika fungsi tersebut dipanggil:

```
class Car(var brand: String, var model: String, var year: Int) {
    // Class function
    fun drive() {
        println("Wrooom!")
    }
}

fun main() {
    val car1 = Car( brand: "Ford",  model: "Mustang",  year: 2002)

    // Call the function
    car1.drive()
}
```

2.3 Flutter

Flutter adalah platform yang digunakan para developer untuk membuat aplikasi multiplatform hanya dengan satu basis coding (codebase). Artinya, aplikasi yang dihasilkan dapat dipakai di berbagai platform,

baik mobile Android, iOS, web, maupun desktop. Flutter menggunakan bahasa pemrograman Dart, yang juga dikembangkan oleh Google. Flutter memiliki dua komponen penting, yaitu :

Software Development Kit (SDK) merupakan sekumpulan tools yang berfungsi untuk membuat aplikasi supaya bisa dijalankan di berbagai platform.Framework UI merupakan komponen UI, seperti teks, tombol, navigasi, dan lainnya, yang dapat di kustomisasi sesuai kebutuhan.

A. Variabel

1. Variabel Lokal (Local Variable)

   Variabel lokal hanya dapat diakses di dalam fungsi atau blok di mana mereka dideklarasikan.

```dart
void main() {
  // Mendeklarasikan dan menginisialisasi variabel lokal
  String nama = "Jesica";
  int umur = 21;

  // Menggunakan variabel lokal
  print("Nama: $nama, Umur: $umur");
}
```

2. Variabel Instans (Instance Variable)

   Variabel instans adalah variabel yang terkait dengan suatu objek. Mereka dapat diakses melalui objek tersebut.

```dart
class Person {
  String nama;
  int umur;

  // Konstruktor untuk menginisialisasi variabel instans
  Person(this.nama, this.umur);
}

Run | Debug | Profile
void main() {
  // Membuat objek dari kelas Person
  Person person = Person("John", 25);

  // Menggunakan variabel instans
  print("Nama: ${person.nama}, Umur: ${person.umur}");
}
```

3. Variabel Statik (Static Variable)

   Variabel statik terkait dengan kelas, bukan dengan objek. Mereka dapat diakses langsung dari kelas tanpa membuat objek.

```dart
class Counter {
  static int count = 0;
}

Run | Debug | Profile
void main() {
  // Menggunakan variabel statik tanpa membuat objek
  print("Count: ${Counter.count}");

  // Mengubah nilai variabel statik
  Counter.count += 1;
  print("Count: ${Counter.count}");
}
```

4. Variabel Final dan Const

   Variabel final dan const digunakan untuk nilai yang tidak dapat diubah. Nilai final dapat diinisialisasi hanya sekali, sementara const adalah konstanta di waktu kompilasi.

```dart
void main() {
  // Variabel final
  final String nama = "John";
  final int umur = 25;

  // Variabel const
  const double pi = 3.14;

  print("Nama: $nama, Umur: $umur, Nilai Pi: $pi");
}
```

B. Class

   Deklarasikan kelas dengan kata kunci class, diikuti oleh nama kelas. Nama kelas biasanya diawali dengan huruf kapital.

```dart
class Person {
  String name;
  int age;

  // Konstruktor
  Person(this.name, this.age);
}
```

C. Fungsi

1. Fungsi Sederhana

   Fungsi yang tidak mengembalikan nilai.

```dart
void sayHello() {
  print("Hello, Flutter!");
}

Run | Debug | Profile
void main() {
  sayHello();
}
```

2. Fungsi dengan parameter

   Fungsi yang dapat memiliki parameter untuk menerima input.

```dart
void greetUser(String name) {
  print("Hello, $name!");
}

Run | Debug | Profile
void main() {
  greetUser("Jesica");
}
```

3. Fungsi dengan Nilai Kembali

   Fungsi yang dapat mengembalikan nilai dengan menggunakan return.

```dart
int addNumbers(int bil1, int bil2) {
  return bil1 + bil2;
}

Run | Debug | Profile
void main() {
  int result = addNumbers(3, 4);
  print("Result: $result");
}
```

4. Fungsi dengan Parameter Default

   Fungsi yang memiliki parameter dengan nilai default.

```dart
void displayMessage(String message, {String prefix = "Halo"}) {
  print("$prefix: $message");
}

Run | Debug | Profile
void main() {
  displayMessage("This is a message");
  displayMessage("Another message", prefix: "Warning");
}
```

5. Fungsi Anonim

   Fungsi anonim atau lambda.

```dart
void main() {
  // Fungsi anonim untuk menghitung kuadrat
  var square = (int x) => x * x;

  print(square(5));
}
```

6. Fungsi dengan Fungsi sebagai Parameter

   Meneruskan fungsi sebagai parameter ke fungsi lainnya.

```dart
void performOperation(int bil1, int bil2, Function operation) {
  print("Result: ${operation(bil1, bil2)}");
}

Run | Debug | Profile
void main() {
  performOperation(5, 3, (bil1, bil2) => bil1 + bil2);
  performOperation(5, 3, (bil1, bil2) => bil1 * bil2);
}
```

7. Fungsi Rekursif

   Fungsi yang memanggil dirinya sendiri selama proses eksekusi. Pemanggilan diri ini membentuk serangkaian iterasi hingga kondisi terminasi tertentu terpenuhi.

```dart
int factorial(int n) {
  if (n == 0 || n == 1) {
    return 1;
  } else {
    return n * factorial(n - 1);
  }
}

Run | Debug | Profile
void main() {
  print("Factorial of 5: ${factorial(5)}");
}
```

2.4 Sistem Operasi Mobile

1. Android

   OS Android adalah sistem operasi seluler berbasis Linux yang terutama berjalan pada smartphone dan tablet. Platform Android mencakup sistem operasi berdasarkan kernel Linux, GUI, browser web, dan aplikasi pengguna akhir yang dapat diunduh.

   a. Karakteristik

      1) Lengkap (Complete Platform)

         Android merupakan sistem operasi yang aman dan banyak menyediakan tools dalam membangun software dan memungkinkan untuk peluang pengembangan aplikasi.

      2) Terbuka (Open Source Platform)

         Platform Android disediakan melalui lisensi open source. Pengembang dapat dengan bebas untuk mengembangkan aplikasi. Android sendiri menggunakan Linux Kernel 2.6.

      3) Free (Free Platform)

         Android adalah platform/Aplikasi yang bebas untuk develop. Tidak ada lisensi atau biaya royalty untuk dikembangkan pada platform Android. Tidak ada biaya keanggotaan diperlukan. Tidak diperlukan biaya pengujian. Tidak ada kontrak yang

diperlukan. Aplikasi untuk android dapat didistribusikan dan diperdagangkan dalam bentuk apa pun.

4) Dukungan Multi-Tasking

Android mendukung multitasking, memungkinkan pengguna untuk menjalankan beberapa aplikasi secara bersamaan dan beralih di antara aplikasi dengan mudah.

5) Google Play Store

Google Play Store adalah platform distribusi resmi untuk aplikasi Android. Pengguna dapat mengunduh, memperbarui, dan mengelola aplikasi mereka melalui Play Store.

6) Update Sistem Otomatis

Android memiliki fitur pembaruan sistem otomatis yang memungkinkan pembaruan sistem operasi dan perangkat lunak secara berkala untuk meningkatkan kinerja dan keamanan.

7) Near Field Communication (NFC)

Sebagian besar perangkat Android mendukung NFC, yang memungkinkan perangkat elektronik untuk berinteraksi dalam jarak dekat dengan mudah.

8) Infrared Transmission

Membuat pengguna dapat menggunakan android nya sebagai remote control.

b. Spesifikasi

1) Kernel Linux

Android berjalan di atas kernel Linux, yang menyediakan dasar sistem operasi dan interaksi dengan perangkat keras.

2) Arsitektur Prosesor

Android mendukung berbagai arsitektur prosesor, termasuk ARM dan x86. Ini memungkinkan penggunaan pada berbagai perangkat keras.

3) Penyimpanan Internal

Android dapat digunakan pada perangkat dengan kapasitas penyimpanan internal yang berbeda. Versi Android terbaru dapat

mendukung teknologi penyimpanan yang lebih cepat seperti UFS (Universal Flash Storage).

4) Pengelolaan Energi

Android memiliki fitur pengelolaan daya baterai untuk mengoptimalkan penggunaan daya dan meningkatkan daya tahan baterai.

5) Konektivitas

Android mendukung berbagai protokol konektivitas, termasuk Wi-Fi, Bluetooth, NFC, dan data seluler (3G, 4G, 5G).

6) Keamanan

Android memiliki fitur keamanan seperti enkripsi data, model izin aplikasi, pemindaian aplikasi berbahaya, dan pembaruan keamanan berkala.

2. Ios

Apple iOS adalah singkatan dari sistem operasi iPhone dan didesain untuk digunakan dengan perangkat multisentuh Apple. OS seluler ini mendukung input melalui manipulasi langsung dan merespons berbagai gerakan pengguna, seperti mencubit, mengetuk, dan mengusap.

a. Karakteristik

1) Eksklusivitas Perangkat

iOS dirancang khusus untuk perangkat Apple, seperti iPhone, iPad, dan iPod Touch. Ini menciptakan ekosistem tertutup yang terintegrasi secara menyeluruh.

2) Kontrol Penuh Apple

Apple memiliki kontrol penuh atas perangkat keras dan perangkat lunak iOS, memungkinkan pengoptimalan yang baik antara keduanya dan memberikan pengalaman yang konsisten di seluruh perangkat.

3) Antarmuka Pengguna yang Konsisten

iOS memiliki antarmuka pengguna yang konsisten di seluruh perangkatnya. Hal ini menciptakan pengalaman yang serupa

bagi pengguna ketika beralih dari satu perangkat iOS ke perangkat lainnya.

4) App Store Terintegrasi

App Store pada iOS adalah platform resmi untuk distribusi aplikasi. Pengguna dapat mengunduh dan menginstal aplikasi dari App Store, yang diawasi ketat oleh Apple untuk memastikan kualitas dan keamanan.

5) Integrasi Dengan Ekosistem Apple

iOS terintegrasi erat dengan produk-produk dan layanan Apple lainnya, seperti iCloud, iMessage, FaceTime, dan Siri, untuk memberikan pengalaman yang terkoordinasi di seluruh ekosistem Apple.

6) Dukungan ARKit

iOS menyertakan ARKit, platform realitas tambahan (augmented reality) yang memungkinkan pengembang membuat aplikasi AR yang inovatif.

7) Keamanan yang Tinggi

iOS dikenal dengan tingkat keamanan yang tinggi. Apple menerapkan berbagai langkah keamanan, termasuk enkripsi data, izin aplikasi, dan pembaruan keamanan berkala.

8) Pembaruan Sistem Otomatis

Pengguna mendapatkan pemberitahuan untuk pembaruan sistem operasi, dan perangkat dapat diatur untuk mengunduh dan menginstal pembaruan otomatis.

b. Spesifikasi

1) Antarmuka Pengguna (UI)

Antarmuka pengguna iOS didesain untuk kegunaan yang mudah dan intuitif, dengan tata letak ikon yang dikenal sebagai Springboard.

2) App Store

Platform distribusi aplikasi tempat pengguna dapat men-download, menginstal, dan memperbarui aplikasi pihak ketiga.

3) Integrasi dengan Layanan Apple

iOS terintegrasi dengan berbagai layanan Apple, seperti iCloud untuk penyimpanan dan sinkronisasi data, iMessage untuk pesan teks, dan FaceTime untuk panggilan video.

4) AirDrop dan AirPlay

Fitur untuk berbagi file dan konten multimedia antara perangkat Apple, seperti foto, video, dan musik.

5) Metal

Antarmuka pemrograman aplikasi (API) grafis yang dikembangkan oleh Apple untuk kinerja grafis yang tinggi.

3. Windows Mobile

Windows Mobile adalah sistem operasi seluler yang dikembangkan oleh Microsoft, pertama kali dirilis pada tahun 2000 sebagai Pocket PC, juga dilabeli sebagai Windows Mobile Classic dan Windows Mobile Professional. Sistem operasi ini memungkinkan pengguna untuk mengakses banyak fitur yang sama dengan komputer desktop, seperti Office Outlook dan Internet Explorer.

a. Karakteristik

1) Integrasi Microsoft

Windows Mobile secara alami terintegrasi dengan ekosistem Microsoft, termasuk Exchange untuk sinkronisasi email dan kalender.

2) Keamanan

Windows Mobile menyertakan fitur keamanan seperti enkripsi data dan opsi untuk mengelola perangkat jarak jauh.

3) Multi-Tasking

Kemampuan untuk menjalankan beberapa aplikasi secara bersamaan dan beralih di antara mereka.

4) Pembaruan dan Upgrade

Sistem operasi ini mendukung pembaruan perangkat lunak dan perbaikan keamanan melalui pembaruan yang dapat diunduh.

b. Spesifikasi

1) Antarmuka Pengguna (UI)

   Windows Mobile memiliki antarmuka pengguna yang berbasis sentuhan (touchscreen) dengan tata letak ikon dan menu yang mirip dengan sistem operasi Windows pada komputer.

2) Aplikasi dan Software

   Windows Mobile mendukung berbagai aplikasi, termasuk versi mobile dari aplikasi Microsoft Office seperti Word, Excel, dan PowerPoint. Selain itu, ada banyak aplikasi pihak ketiga yang dapat diinstal.

3) Pengaturan Ponsel

   Windows Mobile menyediakan pengaturan yang cukup rinci untuk penggunaan ponsel, termasuk manajemen koneksi, pengaturan keamanan, dan opsi sinkronisasi data.

4. Symbian

   OS Symbian adalah sistem operasi yang dirancang untuk perangkat seluler, terutama ponsel cerdas. Sistem operasi ini dikembangkan oleh Symbian Ltd, sebuah konsorsium dari beberapa perusahaan telepon seluler, termasuk Nokia, Ericsson, dan Motorola. OS Symbian banyak digunakan pada awal tahun 2000-an.

   a. Karakteristik

      1) Kualitas multimedia

         Symbian dirancang untuk multimedia, sehingga hampir semua aplikasi yang kompatibel dapat dijalankan di dalamnya.

      2) Tidak open source secara penuh

         Meskipun terdapat API dan dokumentasi yang membantu pengembang aplikasi, Symbian OS bukanlah software open source secara penuh.

      3) Multithreading dan multitasking

         Symbian OS mampu melakukan operasi secara multithreading, multitasking, dan pengamanan terhadap memori, mirip dengan sistem operasi desktop.

      4) Antarmuka pemrograman aplikasi (API)

Symbian OS memiliki kategori API yang tersedia, seperti API Umum, API Opsional, dan API Opsional Tergantikan, yang memungkinkan pengembang aplikasi untuk membuat software yang berjalan di atas sistem operasi ini.

b. Spesifikasi

1) Antarmuka Pengguna (UI)

Symbian memiliki antarmuka pengguna yang berbasis layar sentuh dengan ikon dan tata letak menu yang khas.

2) Browser dan Konektivitas

Peramban web bawaan di Symbian dapat digunakan untuk menjelajahi internet. Konektivitas termasuk Wi-Fi, Bluetooth, dan dukungan untuk koneksi seluler.

3) Aplikasi dan Software

Symbian mendukung berbagai aplikasi pihak ketiga, baik yang diunduh dari toko aplikasi atau diinstal secara langsung. Aplikasi bawaan meliputi browser web, aplikasi pesan, dan aplikasi produktivitas.

5. BBOS / Blackberry

BlackBerry OS adalah sistem operasi mobile eksklusif yang dikembangkan oleh Research In Motion (RIM) untuk smartphone BlackBerry.

a. Karakteristik

1) Sistem operasi tersendiri

BlackBerry menggunakan sistem operasi yang dirancang khusus untuk mereka, yang mencakup dua sistem BlackBerry Enterprise Server (BES) dan BlackBerry Internet Service (BIS).

2) BBM (BlackBerry Messenger)

Aplikasi pesan instan eksklusif untuk pengguna BlackBerry, memungkinkan pengguna mengirim pesan teks, foto, dan file lain secara instan.

3) Aplikasi MDS

Aplikasi MDS adalah bagian aplikasi BlackBerry yang menyediakan dukungan untuk aplikasi yang memerlukan akses lebih halaman dan fitur yang lebih spesifik.

4) BIS (BlackBerry Internet Service)

   Layanan yang memungkinkan sinkronisasi email dan data dengan server BlackBerry.

5) Aplikasi Pihak Ketiga

   Kemampuan untuk menginstal aplikasi pihak ketiga melalui BlackBerry App World atau sumber lainnya.

6) Update Firmware

   Kemampuan untuk melakukan pembaruan sistem operasi atau firmware.

b. Spesifikasi

1) Antarmuka Pengguna (UI)

   Antarmuka yang bersahabat dan khas BlackBerry, dengan tata letak ikon dan navigasi menggunakan tombol-tombol fisik atau trackpad.

2) BlackBerry Balance

   BlackBerry Balance yang memungkinkan pengguna memisahkan dan mengelola data pribadi dan pekerjaan.

3) Keamanan

   BlackBerry OS dikenal dengan tingkat keamanan yang tinggi, terutama dalam hal enkripsi data dan keamanan email.

4) Sistem File

   BlackBerry OS memiliki sistem file untuk mengelola data dan aplikasi pada perangkat.

BAB III

PENUTUP

3.1 Kesimpulan

Makalah ini membahas beberapa aspek kunci dalam pengembangan aplikasi mobile, terutama berfokus pada lingkup sistem operasi Android. Melalui pembahasan mengenai siklus hidup aktivitas Android, pemahaman terhadap karakteristik bahasa pemrograman Kotlin, eksplorasi framework pengembangan cross-platform seperti Flutter yang menggunakan Bahasa dart, serta karakteristik dan spesifikasi pada berbagai sistem operasi mobile. Makalah ini memberikan wawasan yang berguna dalam dunia pengembangan aplikasi mobile.

3.2 Saran

Perangkat seluler telah menjadi bagian yang penting dari kehidupan sehari-hari di era digital yang semakin maju. Platform seluler seperti Android, iOS, dan Windows Mobile telah berkembang secara signifikan dan menawarkan beragam fitur yang memengaruhi cara kita berinteraksi dengan teknologi. Oleh karena itu, penting untuk mengetahui hal-hal terkait platform ini dengan memahami Android Life Cycle, Karakter Kotlin, Flutter, aan apa saja Sistem Operasi Mobile. Dengan menggabungkan pemahaman mendalam tentang Android Life Cycle, karakter Kotlin, Flutter, dan perbandingan sistem operasi mobile, pembaca akan dibekali dengan pengetahuan yang kokoh untuk menghadapi kompleksitas dan dinamika dalam mengembangkan aplikasi mobile.

DAFTAR PUSTAKA

Android Developers. (n.d.). Memahami Siklus Proses Aktivitas. Diakses dari https://developer.android.com/guide/components/activities/activity-lifecycle?hl=id

W3Schools. (n.d.). Diakses dari https://www.w3schools.com

Niagahoster. (n.d.). Diakses dari https://www.niagahoster.co.id/

Pancabudi Academic Repository. (n.d.). Sistem Operasi: Sebuah Kajian Literatur. Diakses dari https://repository.pancabudi.ac.id/perpustakaan/lokalkonten/16143 73115_71_2_BAB_II.pdf

JavaTpoint. (n.d.). Diakses dari https://www.javatpoint.com

Rahmat Age's Google Sites. (n.d.). Pengertian Sistem Operasi Symbian. Diakses dari https://sites.google.com/a/student.unsika.ac.id/rahmat-age/pengertian-sistem-operasi-symbian

Kompas Tekno. (2021, December 31). Umur Sistem Operasi BlackBerry Tinggal Menghitung Hari. Diakses dari https://tekno.kompas.com/read/2021/12/31/0701 0007/umur-sistem-operasi-blackberry-tinggal-menghitung-hari?page=all

TechTarget. (n.d.). Android OS. Diakses dari https://www.techtarget.com/search mobilecomputing/definition/Android-OS

GSM Arena. (n.d.). BlackBerry OS. Diakses dari https://www.gsmarena.com/gl ossary.php3?term=bb-os

Lenovo. (n.d.). What is Symbian Operating System? Diakses dari https://www.l enovo.com/us/en/glossary/what-is-symbian-operating-system/

TechTarget. (n.d.). Mobile Operating System. Diakses dari https://www.techtarg et.com/searchmobilecomputing/definition/mobile-operating-system

TechTarget. (n.d.). iOS. Diakses dari https://www.techtarget.com/searchmobilec omputing/definition/iOS

LogRocket Blog. (2021, November 19). Understanding the Android activity lifecycle. Retrieved November 23,2023, from https://blog.logrocket.com/underst anding-the-android-activity-lifecycle/