LANGKAH MUDAH MEMBANGUN APLIKASI ANDROID PROFESIONAL DENGAN JAVA

Ari Peryanto | Dwi Susanto

LANGKAH MUDAH MEMBANGUN APLIKASI ANDROID PROFESIONAL DENGAN JAVA

Ari Peryanto
Dwi Susanto

Sanksi Pelanggaran Pasal 113

Undang-undang Republik Indonesia Nomor 28 Tahun 2014

Tentang Hak Cipta

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).
3. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (l) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/ atau pidana denda paling Rp1.000.000.000,00 (satu miliar rupiah).
4. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah).

LANGKAH MUDAH MEMBANGUN APLIKASI ANDROID PROFESIONAL DENGAN JAVA

Ari Peryanto
Dwi Susanto

YAYASAN PUTRA ADI DHARMA

LANGKAH MUDAH MEMBANGUN APLIKASI ANDROID PROFESIONAL DENGAN JAVA

Penulis :
Ari Peryanto
Dwi Susanto

ISBN : 978-634-295-150-7

IKAPI : No.498/JBA/2024

Editor :
Annida Muthi'ah

Penyunting :
Yayasan Putra Adi Dharma

Desain sampul dan Tata letak
Yayasan Putra Adi Dharma

Penerbit :
Yayasan Putra Adi Dharma

Redaksi :
Wahana Pondok Ungu Blok B9 no 1,Bekasi
Office Marketing Jl. Gedongkuning, Banguntapan Bantul, Yogyakarta
Office Yogyakarta : 087777899993
Marketing : 088221740145
Instagram : @ypad_penerbit
Website : https://ypad.store
Email : teampenerbit@ypad.store

Cetakan Pertama Mei 2026

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

PRAKATA

Segala puji bagi Allah SWT atas tuntasnya penyusunan buku **"Langkah Mudah Membangun Aplikasi Android Profesional dengan Java"**. Buku ini lahir dari pengamatan penulis terhadap tantangan yang sering dihadapi mahasiswa saat pertama kali terjun ke dunia pengembangan aplikasi *mobile*.

Banyak yang menganggap bahwa membangun aplikasi Android itu sulit karena kompleksitas *tools* dan bahasanya. Namun, melalui buku ini, penulis ingin membuktikan bahwa dengan langkah-langkah yang terstruktur—mulai dari persiapan *environment*, pemahaman siklus hidup aktivitas, hingga pengelolaan database lokal—siapa pun dapat menciptakan aplikasi yang fungsional dan profesional.

Buku ini disusun secara praktis agar pembaca tidak hanya memahami teori, tetapi langsung mempraktikkannya melalui proyek nyata aplikasi pengelola catatan. Penulis berharap buku ini dapat menjadi kawan belajar yang menyenangkan bagi para mahasiswa maupun otodidak yang ingin berkontribusi di industri digital.

Terima kasih kepada semua pihak yang telah mendukung proses penulisan buku ini. Kritik dan saran yang membangun sangat penulis harapkan demi kesempurnaan di edisi mendatang.

Selamat belajar dan teruslah berkarya!

Februari 2026

Penulis

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas selesainya buku "**Langkah Mudah Membangun Aplikasi Android Profesional dengan Java**".

Di era *mobile-first* ini, kemahiran membangun aplikasi Android merupakan kompetensi wajib bagi mahasiswa Informatika dan pengembang perangkat lunak. Buku ini hadir sebagai panduan praktis yang membimbing Anda secara bertahap, mulai dari pengenalan ekosistem Android Studio, penguasaan *Activity* dan *Intent*, hingga manajemen database SQLite.

Keunggulan buku ini terletak pada pendekatan "belajar dengan berbuat" (*learning by doing*), yang memuncak pada proyek akhir pembuatan aplikasi pengelola catatan yang aplikatif. Tidak hanya teknis pengodean, buku ini juga menekankan logika pemrograman agar pembaca mampu beradaptasi dengan perkembangan teknologi mobile di masa depan.

Semoga buku ini menjadi jembatan yang efektif bagi pembaca dalam mewujudkan ide-ide inovatif ke dalam aplikasi Android yang profesional. Selamat berkarya!

Jakarta, Februari 2025

Penulis

Bab 1
Persiapan Lingkungan dan Dasar Android

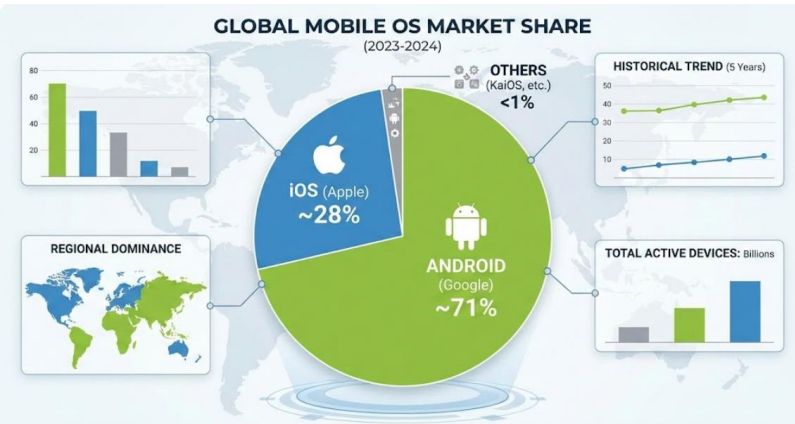
Selamat datang di dunia pengembangan aplikasi *mobile*. Bab ini dirancang sebagai gerbang pembuka bagi mahasiswa untuk memasuki ekosistem Android yang luas dan dinamis. Sebelum kita menyelami baris-baris kode yang rumit, sangat penting untuk membangun fondasi pemahaman yang kokoh mengenai lingkungan kerja (*environment*) yang akan kita gunakan. Pada tahap ini, kita tidak hanya sekadar menginstal perangkat lunak, tetapi juga memahami bagaimana komponen-komponen tersebut berinteraksi satu sama lain untuk mengubah kode Java menjadi aplikasi yang dapat berjalan di saku jutaan pengguna. Kita akan memulai perjalanan ini dengan memahami urgensi teknologi *mobile*, mempersiapkan "dapur" pengembangan kita (Android Studio), hingga akhirnya berhasil menyapa dunia melalui aplikasi pertama Anda.

1.1. Mengapa Pemrograman Mobile?

Dalam era transformasi digital saat ini, paradigma komputasi telah bergeser drastis dari *desktop-centric* menjadi *mobile-first*. Perangkat seluler bukan lagi sekadar alat komunikasi, melainkan komputer saku dengan kemampuan pemrosesan tinggi yang menjadi pusat aktivitas manusia, mulai dari transaksi perbankan, pendidikan, hingga hiburan. Bagi seorang sarjana Informatika, menguasai pemrograman *mobile* adalah sebuah keharusan strategis untuk menjawab kebutuhan industri yang menuntut aksesibilitas sistem di mana saja dan kapan saja.

Android, sebagai sistem operasi yang dikembangkan oleh Google dan *Open Handset Alliance*, menawarkan ekosistem yang sangat unik. Dibangun di atas kernel Linux, Android bersifat terbuka (*open*

source), yang memungkinkan adopsi massal oleh berbagai vendor perangkat keras. Hal ini menciptakan pangsa pasar yang sangat besar dan variatif. Berbeda dengan pengembangan aplikasi *desktop* atau *web*, pemrograman *mobile* menuntut pemahaman khusus terkait manajemen sumber daya (baterai, memori terbatas), siklus hidup aplikasi (*lifecycle*), dan interaksi sentuh (*touch interface*).



Gambar 1.1. Statistik pangsa pasar Sistem Operasi *Mobile* Global

Sebagai perbandingan mendasar agar Anda memahami posisi Android dalam peta teknologi global, mari kita tinjau perbedaannya dengan pesaing utamanya, iOS. Tabel berikut menyajikan perbedaan kunci yang perlu dipahami pengembang:

Tabel 1 Perbandingan Karakteristik Android vs iOS

Aspek	Android	iOS
Filosofi	**Open Source.** Kode inti AOSP terbuka, bisa dimodifikasi vendor manapun.	**Closed Source.** Ekosistem tertutup, dikontrol penuh oleh Apple.
Bahasa	**Java & Kotlin.** Java (Legasi/Klasik) dan Kotlin (Modern).	**Swift & Obj-C.** Swift menggantikan Objective-C.

Perangkat	**Fragmentasi Tinggi.** Beragam merk, ukuran layar, dan spesifikasi hardware.	**Terstandarisasi.** Hanya perangkat iPhone/iPad buatan Apple.
Distribusi	**Fleksibel.** Play Store, toko pihak ketiga, atau instal manual (APK).	**Ketat.** Praktis hanya melalui Apple App Store.

Memahami karakteristik di atas sangat krusial karena fragmentasi perangkat pada Android menuntut kita untuk merancang antarmuka yang adaptif (*responsive*) agar aplikasi tetap terlihat bagus baik di layar kecil maupun tablet.

1.2. Instalasi Android Studio

Pengembangan aplikasi Android modern terpusat pada satu *tool* utama: **Android Studio**. Ini adalah *Integrated Development Environment* (IDE) resmi yang dibangun di atas IntelliJ IDEA, sebuah IDE Java yang sangat cerdas. Android Studio menyediakan semua alat yang dibutuhkan untuk membuat aplikasi, mulai dari editor kode, desainer antarmuka visual, hingga alat analisis performa (*profiler*).

Prasyarat Sistem dan Perangkat Lunak

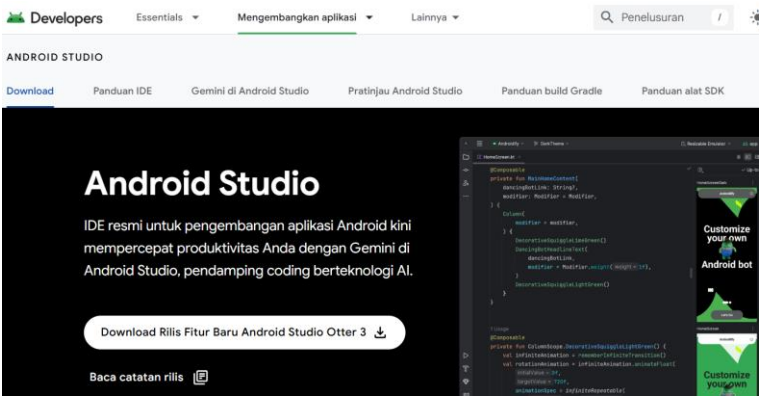
Sebelum melakukan instalasi, kita harus memahami ketergantungan utama Android Studio, yaitu **JDK (Java Development Kit)**. Meskipun Android tidak menjalankan *bytecode* Java standar (melainkan format *Dex bytecode* yang dijalankan di atas Android Runtime/ART), proses kompilasi awal tetap membutuhkan kompiler Java.

- **JDK,** Android Studio versi terbaru biasanya telah mem-bundling OpenJDK di dalamnya. Namun, sebagai pengembang Java, pastikan konfigurasi JAVA_HOME pada sistem operasi Anda sudah tepat jika Anda menggunakan versi JDK tertentu.

- **Spesifikasi Hardware,** Pengembangan Android tergolong berat. Untuk kenyamanan, disarankan menggunakan RAM minimal 8GB (16GB lebih direkomendasikan) dan wajib menggunakan penyimpanan jenis SSD untuk mempercepat proses *build* yang melibatkan ribuan file kecil.

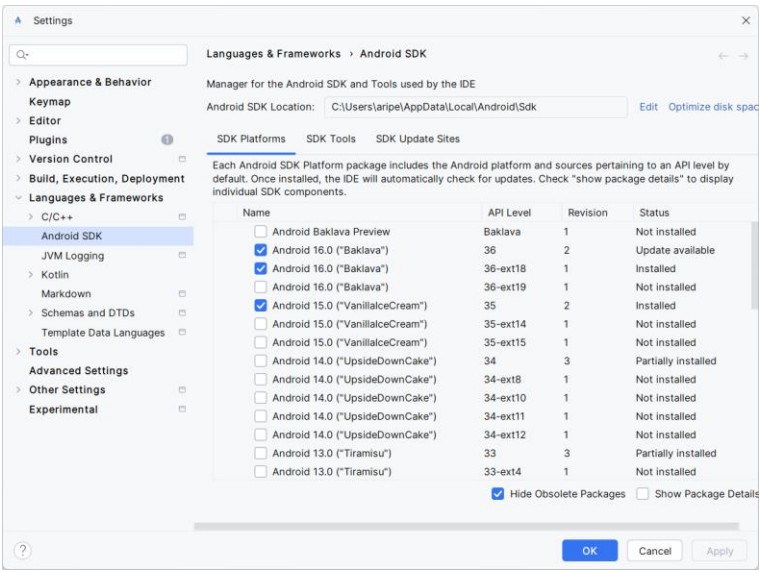
Langkah-Langkah Instalasi Detail

1. **Pengunduhan,** Unduh instalator dari situs resmi developer.android.com. Pastikan Anda mengunduh versi stabil (*Stable Channel*) untuk pembelajaran agar minim *bug*.



Gambar 1.2. Tampilan halaman unduhan resmi Android Studio

2. **Android SDK (Software Development Kit),** Saat proses instalasi, *wizard* akan meminta mengunduh komponen SDK. SDK adalah kumpulan pustaka (*libraries*), *debugger*, emulator, dan dokumentasi yang diperlukan untuk mengembangkan aplikasi Android. Jangan lewatkan tahap ini.



Gambar 1.3. Jendela konfigurasi komponen SDK saat instalasi

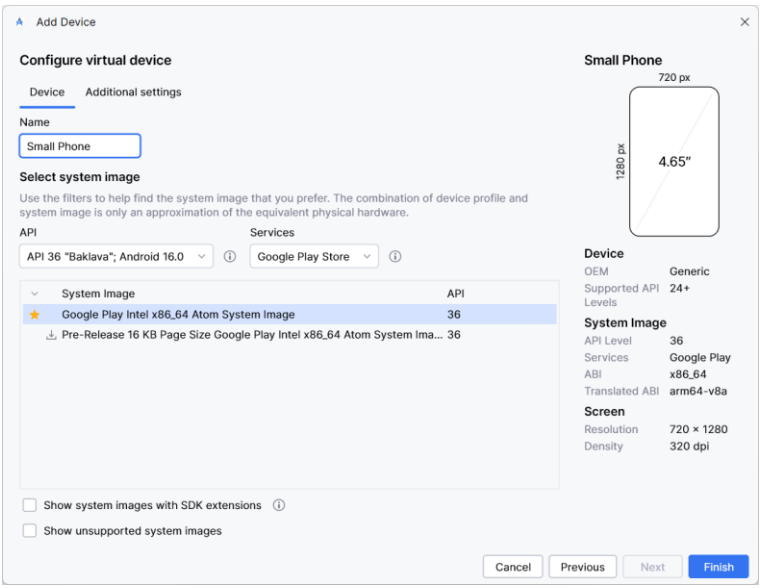
3. **SDK Manager,** Setelah terinstal, Anda bisa mengakses **SDK Manager**. Di sini Anda wajib mengunduh *SDK Platforms* (versi Android target, layer5 Android 14) dan *SDK Tools* (seperti Android SDK Build-Tools, Android Emulator, dan *Intel x86 Emulator Accelerator* untuk mempercepat emulator di prosesor Intel).

Konfigurasi Emulator (AVD)

Menjalankan aplikasi di perangkat fisik adalah yang terbaik, namun emulator sangat penting untuk menguji aplikasi di berbagai ukuran layer tanpa membeli banyak perangkat.

1. Buka **Device Manager** di Android Studio.
2. Pilih **Create Device**. Anda akan dihadapkan pada pilihan profil perangkat keras (layer5: Pixel 7). Pilihlah profil yang memiliki logo “Play Store” agar emulator tersebut memiliki layanan Google Play.

3. Pilih **System Image** (Versi OS). Gunakan versi x86_64 untuk performa terbaik di layer66r.
4. Fitur emulator ini mensimulasikan layer6 semua kemampuan perangkat asli, termasuk lokasi GPS, panggilan telepon masuk, hingga rotasi layer.



Gambar 1.4. Pengaturan Virtual Device pada Device Manager

1.3. Proyek Pertama "Hello, Android!"

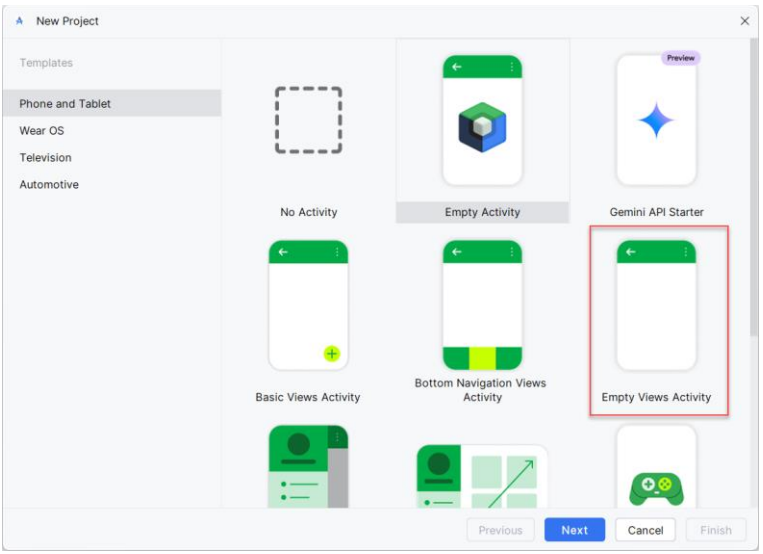
Setelah lingkungan siap, saatnya membuat proyek pertama. Dalam tradisi pemrograman, kita akan membuat aplikasi "Hello World". Tujuan sub-bab ini bukan sekadar menampilkan teks, melainkan memahami struktur direktori proyek Android yang cukup kompleks.

Pembuatan Proyek (Wizard)

1. Pilih **New Project**.
2. Pilih *template* **Empty Views Activity**. *Catatan penting*,

Android Studio modern menyediakan template "Empty Activity" yang menggunakan *Jetpack Compose* (toolkit UI deklaratif modern). Karena buku ini fokus pada pendekatan klasik menggunakan Java dan XML, pastikan Anda memilih template yang mendukung **Views** (biasanya bernama *Empty Views Activity*).

3. Isi **Name** dengan "HelloAndroid".
4. Pada kolom **Language**, pilih **Java**.
5. **Minimum SDK,** Pilih API 24 (Android 7.0 Nougat). Pemilihan ini menentukan batas bawah perangkat yang bisa menginstal aplikasi Anda.



Gambar 1.5. Pemilihan template Empty Views Activity

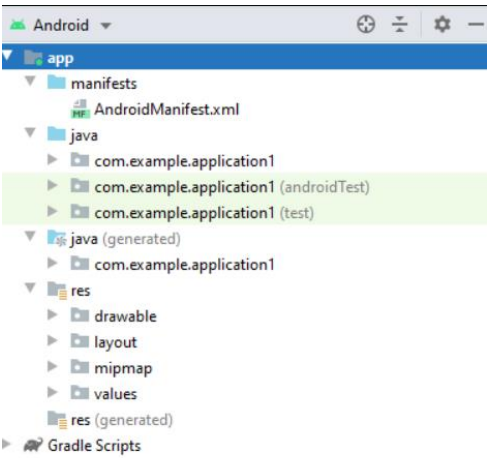
Membedah Struktur Proyek

Setelah *Gradle* (sistem otomatisasi *build*) selesai memproses, Anda akan melihat struktur folder di panel kiri:

1. **Manifests (AndroidManifest.xml),** Ini adalah file deskriptor penting. Sistem Android membaca file ini sebelum kode

apapun dijalankan. Di sini dideklarasikan nama aplikasi, ikon, dan *Activity* mana yang akan muncul pertama kali (*launcher*).

2. **Java (com.example.application1),** Direktori ini berisi seluruh kode logika (*source code*) Java. Di sinilah otak aplikasi berada.
3. **Res (Resources),** Direktori ini berisi sumber daya non-kode.
   - layout: Berisi file XML untuk desain antarmuka (UI).
   - drawable: Berisi gambar atau ikon.
   - values: Berisi konstanta seperti teks (strings.xml) dan warna (colors.xml).
4. **Gradle Scripts,** Berisi file build.gradle (Module dan Project). File ini mengatur dependensi pustaka pihak ketiga dan konfigurasi versi aplikasi.



Gambar 1.6. Struktur direktori proyek Android (Manifest, Java, Res)

Implementasi Kode

Buka file MainActivity.java. Anda akan melihat kode berikut:

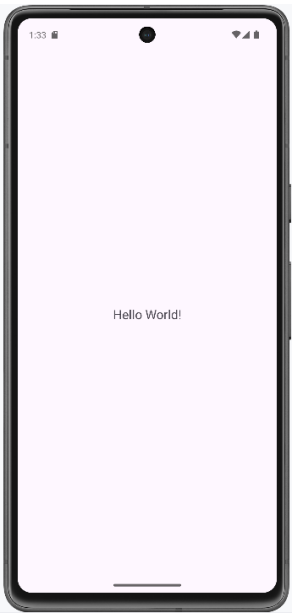
Java

```java
package com.example.application1;

// Import pustaka kelas dasar Android
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    // Metode onCreate adalah titik awal siklus hidup Activity
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Perintah ini 'menempelkan' desain XML ke layar Java
        setContentView(R.layout.activity_main);
    }
}
```

Analisis Kode:

- AppCompatActivity, Kelas induk yang menyediakan kompatibilitas mundur (*backward compatibility*) untuk fitur UI modern di versi Android lama.
- setContentView(R.layout.activity_main), Di sinilah jembatan antara logika (Java) dan tampilan (XML). Kelas R adalah kelas khusus yang dihasilkan otomatis oleh Android Studio yang berisi referensi ke semua *resources* di folder res.

Jalankan aplikasi dengan menekan tombol **Run**. Gradle akan mengkompilasi kode Java menjadi *bytecode*, memaketkannya menjadi APK, dan menginstalnya ke emulator.



Gambar 1.7. Hasil keluaran aplikasi Hello World pada Emulator

1.4. Mengenal Komponen Dasar Aplikasi

Aplikasi Android tidak memiliki satu titik masuk tunggal seperti fungsi main() pada program C++ atau Java Desktop standar. Sebaliknya, aplikasi Android terdiri dari komponen-komponen yang dapat dipanggil secara independen oleh sistem. Tiga komponen visual paling fundamental yang wajib dipahami pemula adalah:

1. **Activity (Aktivitas)**
   Activity adalah komponen yang menyediakan layar tempat pengguna berinteraksi. Satu aplikasi bisa memiliki satu atau banyak *Activity*. Misalnya, aplikasi email mungkin memiliki

satu *Activity* untuk daftar email, satu untuk membaca pesan, dan satu untuk menulis pesan. Setiap *Activity* berdiri sendiri namun bekerja sama.

2. **View (Tampilan/Widget)**
   View adalah blok bangunan dasar untuk antarmuka pengguna. Dalam istilah teknis, ini adalah kelas android.view.View. Segala sesuatu yang tampak di layar adalah *View*.
   - **Widget,** Komponen interaktif seperti tombol (Button), kolom isian teks (EditText), atau gambar (ImageView).
   - **Hierarki,** View disusun dalam struktur pohon (*tree structure*), di mana elemen induk mengatur elemen anak.
3. **Layout (Tata Letak)**

   Layout adalah jenis khusus dari *View* (disebut ViewGroup) yang tugas utamanya bukan menampilkan konten, melainkan mengatur posisi *View* lain di dalamnya. Tanpa *Layout*, komponen UI akan bertumpuk tidak beraturan di pojok kiri atas layar. Contohnya adalah LinearLayout (linear vertikal/horizontal) dan ConstraintLayout (fleksibel).

Latihan

Lakukan eksplorasi pada file res/values/colors.xml. Tambahkan definisi warna baru, misalnya warna oranye. Kemudian, buka activity_main.xml dan aplikasikan warna tersebut sebagai latar belakang (background) dari layar aplikasi Anda. Ini akan melatih pemahaman Anda tentang hubungan antar *resource*.

Tugas

Buatlah proyek baru bernama "BiodataDiri". Aplikasi ini harus menampilkan:

1. Nama lengkap Anda menggunakan komponen TextView.

2. Nomor Induk Mahasiswa (NIM).
3. Sebuah foto diri (gunakan komponen ImageView dan letakkan file foto di folder drawable). Pastikan struktur proyek rapi dan aplikasi dapat dijalankan di perangkat fisik Anda. Dokumentasikan proses instalasi di HP Anda dalam sebuah laporan singkat.

Referensi

1. Android Developers. (2025). *Application Fundamentals*. Google Developers. https://developer.android.com/guide/components/fundamentals
2. Android Developers. (2024). *Meet Android Studio*. Google Developers. https://developer.android.com/studio/intro
3. Griffiths, D., & Griffiths, D. (2022). *Head First Android Development: A Brain-Friendly Guide* (3rd ed.). O'Reilly Media.
4. Nuraeni, F., & Albar, M. (2021). *Implementasi Arsitektur MVVM pada Aplikasi Android Berbasis Java*. Jurnal Teknologi Informasi dan Ilmu Komputer, 8(2), 233-242.
5. Smyth, N. (2023). *Android Studio Iguana Essentials - Java Edition: Developing Android Apps Using Android Studio and Java*. Payload Media.
6. Supardi, Y. (2022). *Semua Bisa Menjadi Programmer Android Studio (Case Study: Java)*. Jakarta: Elex Media Komputindo.

Bab 2
Memahami Antarmuka Android Studio

Pernahkah Anda masuk ke dalam kokpit pesawat terbang dan merasa pusing melihat ratusan tombol yang bertebaran di mana-mana? Perasaan yang sama mungkin Anda alami saat pertama kali membuka Android Studio. Jangan khawatir, itu reaksi yang wajar.

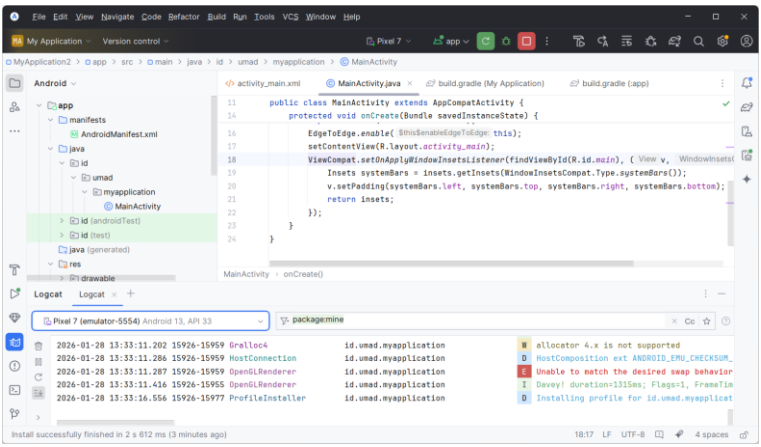
Android Studio memang sebuah *Integrated Development Environment* (IDE) "monster". Ia dibekali segudang fitur canggih untuk menangani segala kebutuhan pengembangan, mulai dari menulis kode, mendesain tampilan, hingga melacak *bug* yang bersembunyi. Namun, kabar baiknya: Anda tidak perlu menekan semua tombol itu sekaligus.

Di bab ini, kita akan berkenalan pelan-pelan dengan "dapur" baru kita ini. Kita akan membedah anatomi proyek Android agar Anda tidak tersesat, belajar cara cepat menggunakan pintasan (*shortcuts*), dan yang tak kalah penting: menyiapkan "mesin waktu" menggunakan Git agar Anda tidak panik saat kode yang ditulis tiba-tiba *error* parah. Mari kita mulai tur kita.

2.1. Tour Antarmuka Android Studio

Saat Android Studio terbuka, layar monitor Anda terbagi menjadi beberapa area utama. Bayangkan ini seperti meja kerja seorang montir profesional; ada tempat untuk meletakkan mesin (editor kode), ada rak perkakas (tool windows), dan ada monitor diagnostik (Logcat).

Agar lebih mudah, mari kita fokus pada bagian-bagian yang paling sering kita sentuh sehari-hari:



Gambar 2.1. Peta area kerja utama pada Android Studio

1. **Editor Window (Tengah)**
   Inilah panggung utama kita. Di sinilah Anda akan menghabiskan 80% waktu Anda untuk mengetik kode Java atau merancang layout XML. Jika Anda membuka file MainActivity.java, di sinilah isinya muncul.
2. **Project Tool Window (Kiri)**
   Ini adalah penjelajah file Anda. Secara *default*, tampilannya diatur ke mode "Android" yang menyederhanakan struktur folder asli agar lebih enak dilihat. Kalau Anda bingung mencari di mana file gambar disimpan, di sinilah tempat mencarinya.
3. **Toolbar (Atas)**
   Berisi tombol-tombol pintas untuk aksi krusial, seperti tombol "Run" (ikon segitiga hijau) untuk menjalankan aplikasi, atau tombol untuk memilih perangkat emulator.
4. **Logcat & Build (Bawah)**
   Pernah merasa aplikasi tiba-tiba menutup sendiri (*Force Close*)? Jawabannya ada di **Logcat**. Ini adalah area diagnostik

yang mencatat semua kejadian sistem secara *real-time*. Jika aplikasi *error*, Logcat akan "berteriak" dengan teks berwarna merah memberi tahu di baris mana kesalahan terjadi.

Senjata Rahasia: Pintasan Keyboard

Seorang pemrogram yang andal jarang menyentuh *mouse* jika tidak terpaksa. Menggunakan *keyboard shortcuts* akan mempercepat kerja Anda berkali-kali lipat. Berikut adalah "jurus" yang wajib Anda hafalkan di luar kepala:

Tabel 2.1. Pintasan Keyboard Wajib Tahu

Aksi	**Windows/Linux**	**Mac**
Mencari Segalanya	Shift (tekan 2x)	Shift (tekan 2x)
Pencarian di File	Ctrl + F	Command + F
Perbaiki Error (Show Intent)	Alt + Enter	Option + Return
Format Kode Otomatis	Ctrl + Alt + L	Cmd + Opt + L
Duplicate Line	Ctrl + D	Command + D

Cobalah tekan **Shift** dua kali sekarang. Kotak pencarian sakti akan muncul, memungkinkan Anda mencari nama file, nama kelas, atau bahkan menu pengaturan yang tersembunyi.

2.2. Struktur Proyek Android

Salah satu hal yang paling membingungkan bagi pemula adalah struktur folder Android. "Kenapa banyak sekali foldernya? Kenapa tidak dijadikan satu saja?"

Jawabannya adalah: **Pemisahan Tanggung Jawab**. Android memaksa kita untuk rapi sejak awal. Kita tidak boleh mencampuradukkan logika (Java) dengan tampilan (XML) atau data mentah (Gambar/Suara). Mari kita bedah anatominya satu per satu.

1. Manifests (AndroidManifest.xml)

Anggap file ini sebagai **KTP** atau Akta Kelahiran aplikasi Anda. Sistem Android tidak akan menjalankan aplikasi tanpa membaca file ini terlebih dahulu. Di sini tertulis:

- Apa nama aplikasinya?
- Apa ikonnya?
- *Activity* mana yang pertama kali muncul?
- Izin apa yang diminta? (Misal: Izin internet atau kamera).

2. Java (java/)

Ini adalah **Otak** aplikasi. Di dalam folder ini (tepatnya di dalam nama paket/package), tersimpan seluruh logika pemrograman. File MainActivity.java yang kita bahas di Bab 1 tinggal di sini.

3. Res (res/)

Ini adalah **Wajah dan Dekorasi** aplikasi. Semua yang bersifat visual dan non-kode ada di sini.

- drawable: Tempat menyimpan gambar (PNG, JPG) atau ikon vektor.
- layout: Tempat mendesain antarmuka (UI) dalam format XML.
- mipmap: Khusus untuk ikon peluncur aplikasi (*launcher icon*) dengan berbagai resolusi.
- values: Gudang data sederhana. Teks disimpan di strings.xml (agar mudah diterjemahkan ke bahasa lain), warna di colors.xml.

4. Gradle Scripts

Ini adalah **Tukang Bangunan** (*Builder*). Android Studio menggunakan sistem bernama **Gradle** untuk "memasak" semua folder di atas menjadi satu file APK. File yang paling sering Anda edit adalah build.gradle (Module: app). Di file inilah kita mengatur versi Android minimal atau menambahkan pustaka tambahan (*library*) dari internet.

2.3. Menggunakan Emulator dan USB Debugging

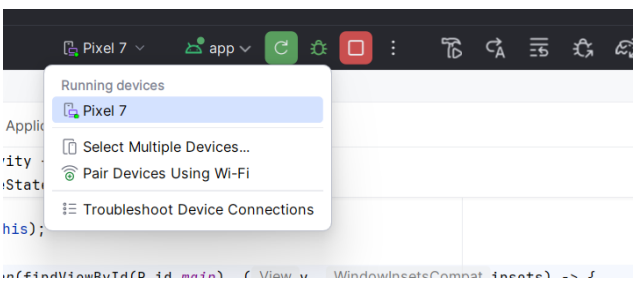
Kode sudah ditulis, tapi bagaimana cara melihat hasilnya? Kita punya dua pilihan: menggunakan simulasi (Emulator) atau barang nyata (HP Fisik).

Menggunakan Emulator (AVD)

Emulator ibarat memiliki HP virtual di dalam komputer Anda. Praktis, tapi "berat". Emulator memakan RAM komputer cukup besar.

- **Kelebihan:** Bisa tes berbagai ukuran layar tanpa beli HP banyak.
- **Kekurangan:** Lambat jika spesifikasi komputer pas-pasan.

Cara menjalankannya cukup pilih jenis perangkat di *Toolbar* atas, lalu klik tombol **Run**.



Gambar 2.2. Memilih target device pada Toolbar

USB Debugging (Perangkat Fisik)

Ini adalah cara favorit saya. Menjalankan aplikasi langsung di HP sendiri terasa lebih nyata, lebih cepat, dan ringan bagi komputer.

Namun, HP Android memiliki fitur keamanan yang memblokir instalasi aplikasi mentah dari komputer. Anda harus membuka kuncinya dulu:

1. Buka **Settings** di HP -> **About Phone**.
2. Cari **Build Number**, ketuk 7 kali dengan cepat sampai muncul tulisan *"You are now a developer!"*.
3. Kembali ke menu utama, cari menu **Developer Options**.
4. Aktifkan opsi **USB Debugging**.
5. Colokkan kabel data ke laptop. Jika muncul *pop-up* izin di HP, klik **Allow**.

Sekarang, nama HP Anda akan muncul di Android Studio. Rasakan sensasi aplikasi buatan sendiri berjalan di genggaman tangan Anda!

2.4. Manajemen Kode dengan Git

Pernahkah Anda mengalami kejadian horor ini: Kode sudah jalan bagus, lalu Anda edit sedikit untuk eksperimen, tiba-tiba semuanya rusak. Anda tekan Ctrl+Z, tapi sudah terlambat. Kode hancur lebur.

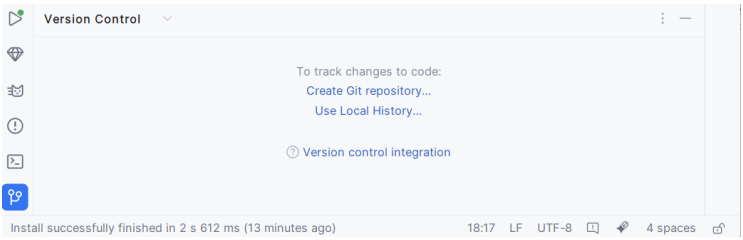
Di sinilah **Git** berperan sebagai "Mesin Waktu".

Git adalah sistem pengontrol versi (Version Control System). Bayangkan Git seperti fitur Save Game pada permainan video. Sebelum melawan bos (baca: melakukan perubahan besar), Anda melakukan Save (baca: **Commit**). Jika kalah (baca: kode rusak), Anda bisa Load Game kembali ke titik sebelum perang terjadi.

Integrasi Git di Android Studio

Anda tidak perlu menghafal perintah terminal yang rumit. Android Studio sudah menyediakan antarmuka visual (GUI) untuk Git.

1. **Enable Version Control,** Menu VCS -> Enable Version Control Integration -> Pilih Git.
2. **Commit (Simpan Titik Aman),** Klik tab **Commit** di panel kiri (atau tekan Ctrl+K). Pilih file yang ingin disimpan, beri pesan (misal: "Menambahkan layout login"), lalu klik **Commit**.



Gambar 2.3. Jendela GIT pada Android Studio

Jika Anda bekerja dalam tim atau ingin menyimpan kode di awan (*cloud*) agar aman dari laptop rusak, Anda bisa menghubungkannya ke **GitHub**.

Latihan dan Tugas

Agar teori di atas menempel di kepala, mari kita praktikkan. Jangan takut salah, karena kita sedang belajar.

Latihan

1. Buka proyek "Hello World" atau "BiodataDiri" Anda.
2. Lakukan perubahan kecil, misalnya ubah warna teks di colors.xml menjadi merah.
3. Aktifkan Git (VCS).

4. Lakukan **Commit** dengan pesan: "Mengubah warna teks utama menjadi merah".
5. Coba ubah lagi warnanya menjadi biru, lalu lihat di panel Git, apakah Anda bisa melihat perbedaan (*diff*) antara versi merah dan biru?

Tugas

Seorang *developer* modern wajib punya portofolio *online*.

1. Buat akun di GitHub.com (jika belum punya).
2. Di Android Studio, hubungkan akun GitHub Anda (Menu File -> Settings -> Version Control -> GitHub).
3. Bagikan proyek Anda ke GitHub (Menu Git -> GitHub -> Share Project on GitHub).
4. Pastikan kode Anda muncul di situs GitHub. Tautan (*link*) repositori ini bisa Anda kumpulkan sebagai bukti tugas.

Referensi

1. Android Developers. (2025). *Android Studio User Guide*. Google Developers. https://developer.android.com/studio/intro
2. Chacon, S., & Straub, B. (2024). *Pro Git* (2nd ed.). Apress. (Tersedia gratis di git-scm.com).
3. Haryanto, E. (2023). *Manajemen Proyek Perangkat Lunak dengan Git dan GitHub*. Bandung: Informatika.
4. Marsicano, K., et al. (2022). *Android Programming: The Big Nerd Ranch Guide* (5th ed.). Big Nerd Ranch Guides.
5. Schwaber, C. (2024). *Mastering Android Studio: Build, Debug, and Test Android Applications*. Packt Publishing.

Bab 3
Merancang Tata Letak Antarmuka (Layout)

Pernahkah Anda menggunakan aplikasi yang fungsinya hebat, tapi tombolnya berantakan, warnanya menabrak mata, dan teksnya sulit dibaca? Kemungkinan besar, Anda akan segera menghapusnya.

Di dunia pengembangan aplikasi, "Jangan menilai buku dari sampulnya" tidak berlaku. Pengguna menilai kualitas kode Anda dari apa yang mereka lihat pertama kali: antarmuka pengguna atau *User Interface* (UI).

Jika Bab 1 dan 2 adalah tentang mempersiapkan "dapur" dan "peralatan masak" kita, maka Bab 3 ini adalah tentang "seni menata piring" (*plating*). Kode Java yang canggih tidak akan berarti apa-apa jika pengguna bingung bagaimana cara menekannya.

Dalam bab ini, kita akan belajar bagaimana menjadi arsitek bagi layar aplikasi kita. Kita akan memahami perbedaan antara komponen dan wadahnya, serta menjajal berbagai strategi penyusunan elemen—mulai dari menyusun balok secara lurus (Linear), menyusun bebas (Relative), hingga menggunakan pegas canggih ala modern (Constraint).

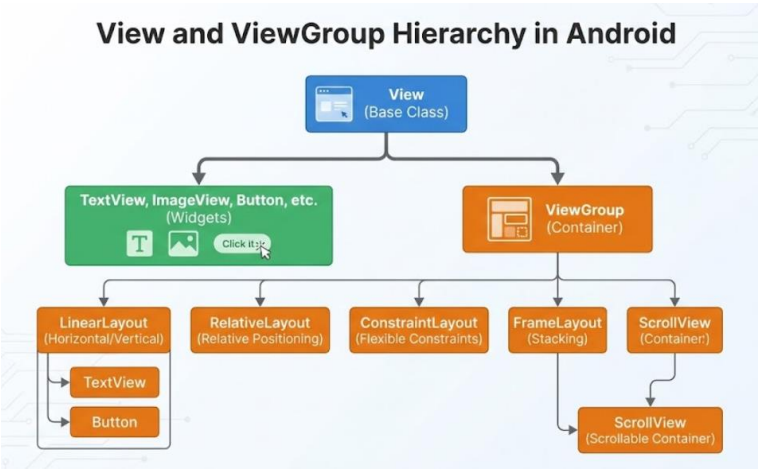
3.1. Konsep Dasar View dan ViewGroup

Sebelum kita mulai menarik garis dan menempatkan tombol, kita harus paham dulu hierarki dasar di Android. Semua yang Anda lihat di layar Android berasal dari satu nenek moyang yang sama, yaitu kelas android.view.View.

Namun, dalam praktiknya, kita membagi mereka menjadi dua kubu besar:

1. **View (Si Objek Nyata)**
   Ini adalah komponen yang terlihat dan berinteraksi dengan pengguna. Contohnya: Tombol (Button), Gambar (ImageView), atau Kolom Teks (EditText). Bayangkan ini sebagai perabotan rumah: kursi, meja, televisi.
2. **ViewGroup (Si Wadah)**
   Ini adalah komponen yang tidak terlihat (transparan), tapi tugasnya sangat krusial: menampung dan mengatur posisi *View* di dalamnya. Contohnya: LinearLayout, RelativeLayout, ConstraintLayout. Bayangkan ini sebagai ruangannya: Ruang Tamu, Kamar Tidur.

Aturan Emas: Anda tidak bisa meletakkan kursi (*View*) melayang di udara. Ia harus berada di dalam sebuah ruangan (*ViewGroup*).



Gambar 3.1. Ilustrasi pohon hierarki View dan ViewGroup

3.2. Layout Populer: Linear & Relative

Meskipun Android Studio modern sangat mempromosikan *ConstraintLayout*, memahami dua layout legendaris ini—Linear dan Relative—sangat penting karena Anda akan sering menemukannya di kode-kode referensi atau proyek lama.

3.2.1. LinearLayout (Si Baris Berbaris)

Ini adalah layout paling sederhana dan paling sering saya gunakan untuk struktur yang simpel. Filosofinya seperti antrean: elemen disusun satu per satu dalam **satu arah saja**.
Kunci utama LinearLayout adalah atribut android:orientation:

- **Vertical:** Menyusun elemen dari atas ke bawah (seperti tumpukan buku).
- **Horizontal:** Menyusun elemen dari kiri ke kanan (seperti gerbong kereta).

Contoh Kode XML:

XML

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:text="Tombol 1 (Atas)"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <Button
        android:text="Tombol 2 (Bawah)"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

```
</LinearLayout>
```

Salah satu fitur "sakti" dari LinearLayout adalah layout_weight. Ini memungkinkan kita membagi ruang layar berdasarkan proporsi persentase, bukan ukuran piksel yang kaku.

3.2.2. RelativeLayout (Si Relatif)

Jika LinearLayout itu kaku seperti tentara baris-berbaris, RelativeLayout itu fleksibel seperti menata furnitur di kamar.

Di sini, posisi sebuah elemen ditentukan berdasarkan hubungannya dengan elemen lain atau dengan wadah induknya (*parent*).

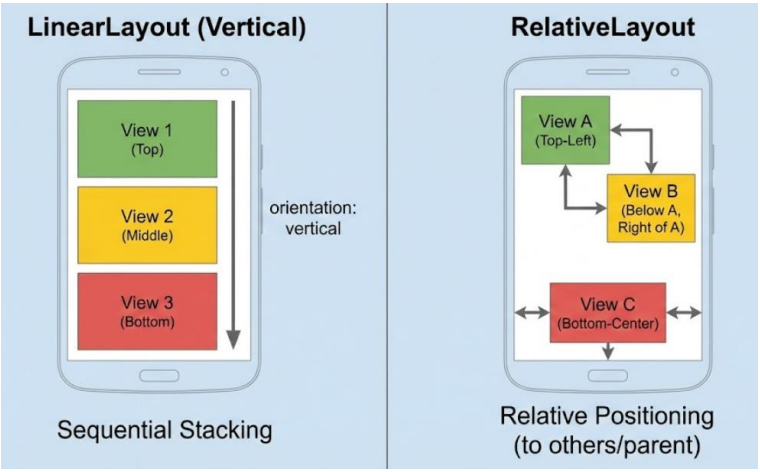
- "Letakkan Tombol A di pojok kanan atas layar."
- "Letakkan Tombol B tepat **di bawah** Tombol A."

Contoh Kode XML:

XML

```
<RelativeLayout ... >

    <Button
        android:id="@+id/btnSimpan"
        android:text="Simpan"
        android:layout_alignParentTop="true"/>

    <Button
        android:text="Batal"
        android:layout_below="@id/btnSimpan"/>

</RelativeLayout>
```

Kelemahan RelativeLayout adalah jika desain terlalu kompleks, performanya bisa menurun karena sistem harus menghitung relasi posisi berkali-kali.



Gambar 3.2. Perbandingan visual susunan LinearLayout vs RelativeLayout

3.3. ConstraintLayout: Layout Modern

Selamat datang di era modern. Sejak 2017, Google menetapkan **ConstraintLayout** sebagai standar *default* pembuatan aplikasi baru.

Mengapa? Karena ConstraintLayout menggabungkan fleksibilitas RelativeLayout dengan performa tinggi, dan dirancang khusus untuk digunakan dengan **Design Editor** (drag-and-drop) di Android Studio.

Filosofi Pegas (Constraints)

Bayangkan Anda memaku sebuah lukisan di dinding, lalu menarik karet gelang (*spring*) dari sisi lukisan ke pinggir dinding.

- Jika Anda menarik karet dari sisi kiri lukisan ke sisi kiri dinding, lukisan akan menempel ke kiri.

- Jika Anda menarik karet dari kiri **dan** kanan sekaligus, lukisan akan berada tepat di tengah (karena gaya tarik seimbang).

Inilah konsep *Centering* di ConstraintLayout. Anda wajib memberikan minimal satu pegangan horizontal dan satu pegangan vertikal agar elemen tidak "terlempar" ke pojok kiri atas (posisi 0,0) saat dijalankan.

Tabel 3.1. Perbandingan Karakteristik Tiga Layout Utama

Layout	**Kelebihan**	**Kekurangan**
Linear	Sangat mudah dipahami, kode rapi untuk susunan sederhana.	Sulit membuat desain tumpang tindih atau kompleks.
Relative	Fleksibel menaruh elemen di mana saja.	Performa lambat jika *nesting* (bersarang) terlalu dalam.
Constraint	Performa tinggi, fitur lengkap, hierarki datar (*flat*).	Kode XML-nya terlihat rumit jika ditulis manual (lebih enak pakai GUI).

3.4. Menggunakan Sumber Daya (Resources)

Seorang *programmer* yang baik itu "malas" mengulang hal yang sama. Dalam Android, kita memisahkan konten (data) dari wadah (layout). Konten-konten ini disimpan dalam folder res (Resources).

Mengapa harus dipisah?

1. **Mudah diubah:** Jika klien minta ubah warna merah jadi biru, Anda cukup ubah 1 baris di colors.xml, dan seluruh aplikasi berubah.

2. **Multi-bahasa:** Memungkinkan aplikasi berubah bahasa otomatis sesuai pengaturan HP pengguna.

File Penting di Folder res/values/:

1. strings.xml (Teks):
   Jangan pernah menulis teks keras (*hardcoded*) di layout seperti android:text="Selamat Datang".
   Tulislah di strings.xml:
   <string name="salam">Selamat Datang</string>
   Lalu panggil di layout: android:text="@string/salam".
2. **colors.xml (Warna):**
   Tempat mendefinisikan palet warna *branding* aplikasi Anda.
   <color name="merah_kampus">#FF0000</color>
3. **dimens.xml (Ukuran):**
   (Opsional) Tempat menyimpan ukuran margin atau besaran *font* agar konsisten.

Latihan dan Tugas

Mari kita latih "otot" desain kita. Jangan khawatir jika hasilnya belum estetik, yang penting strukturnya benar.

Latihan: Kalkulator Sederhana

1. Buatlah tampilan antarmuka kalkulator sederhana tanpa logika Java (hanya tampilan).
2. Gunakan **LinearLayout** dengan orientation="vertical".
3. Baris pertama: Sebuah TextView untuk menampilkan angka "0".
4. Baris kedua hingga kelima: Gunakan LinearLayout horizontal di dalamnya untuk menampung tombol angka (1, 2, 3, +).
5. Gunakan layout_weight="1" pada setiap tombol agar lebar

tombol terbagi rata.

Tugas: Halaman Login

Rancanglah sebuah halaman Login yang rapi menggunakan **ConstraintLayout** atau **RelativeLayout**.

Ketentuan elemen:

1. **Logo:** Gunakan ImageView di bagian atas tengah (silakan pakai gambar bebas di folder drawable).
2. **Input Email:** Gunakan EditText dengan *hint* "Masukkan Email".
3. **Input Password:** Gunakan EditText dengan tipe input *textPassword*.
4. **Tombol Aksi:** Dua tombol berdampingan di bawah input, yaitu "Masuk" dan "Daftar".
5. **Wajib:** Semua teks (judul tombol, hint) harus diambil dari strings.xml.

Referensi

1. Android Developers. (2025). *Layouts*. Google Developers. https://developer.android.com/guide/topics/ui/declaring-layout
2. Dawn, M., & Blake, M. (2021). *Android UI Design: Planning and Building Interfaces*. Pearson Education.
3. Hortasm, M. (2022). *Belajar Desain User Interface Android untuk Pemula*. Bandung: Informatika.
4. Meier, R. (2024). *Professional Android 4 Application Development*. Wrox Press.
5. Phillips, B., et al. (2023). *Android Programming: The Big Nerd Ranch Guide (6th Edition)*. Big Nerd Ranch Guides.

Bab 4
Activity dan Siklus Hidupnya

Pernahkah Anda sedang asyik mengetik pesan WhatsApp, tiba-tiba ada telepon masuk? Layar pesan hilang, digantikan layar panggilan. Setelah telepon ditutup, layar pesan muncul kembali. Ajaibnya, teks yang setengah Anda ketik tadi masih ada, tidak hilang.

Siapa yang mengatur semua "koreografi" layar tersebut? Jawabannya adalah **Siklus Hidup Activity** (*Activity Lifecycle*).

Dalam bab ini, kita akan masuk ke jantungnya aplikasi Android. Jika Layout (Bab 3) adalah wajahnya, maka Activity adalah nyawanya. Kita akan belajar bahwa sebuah layar di Android itu hidup layaknya manusia: ia dilahirkan, ia aktif bekerja, ia bisa "tertidur" sebentar, dan pada akhirnya ia akan dimatikan.

Memahami siklus ini adalah kunci untuk membuat aplikasi yang stabil, tidak mudah *crash*, dan hemat baterai. Mari kita mulai bedah anatominya.

4.1. Apa Itu Activity?

Secara sederhana, **Activity** adalah satu layar tunggal yang dilihat pengguna. Jika aplikasi Anda memiliki layar Login, layar Beranda, dan layar Profil, artinya aplikasi Anda memiliki tiga Activity yang berbeda.

Dalam bahasa teknis pemrograman Java, Activity adalah sebuah kelas (Java Class) yang mewarisi (*extends*) kelas AppCompatActivity.

Namun, Activity bukan sekadar kanvas kosong. Ia adalah komponen cerdas yang tahu kapan harus muncul dan kapan harus sembunyi.

Setiap Activity harus didaftarkan di dalam AndroidManifest.xml. Jika Anda membuat kelas Activity baru tapi lupa mendaftarkannya di Manifest, aplikasi akan *force close* saat mencoba membukanya. Ini seperti punya anak tapi tidak dibuatkan Akta Kelahiran—negara (sistem Android) tidak akan mengakuinya.

4.2. Siklus Hidup Activity

Inilah bagian terpenting. Android tidak membiarkan aplikasi berjalan seenaknya. Sistem operasi mengontrol penuh nasib setiap Activity melalui serangkaian metode yang disebut *Callback Methods*.

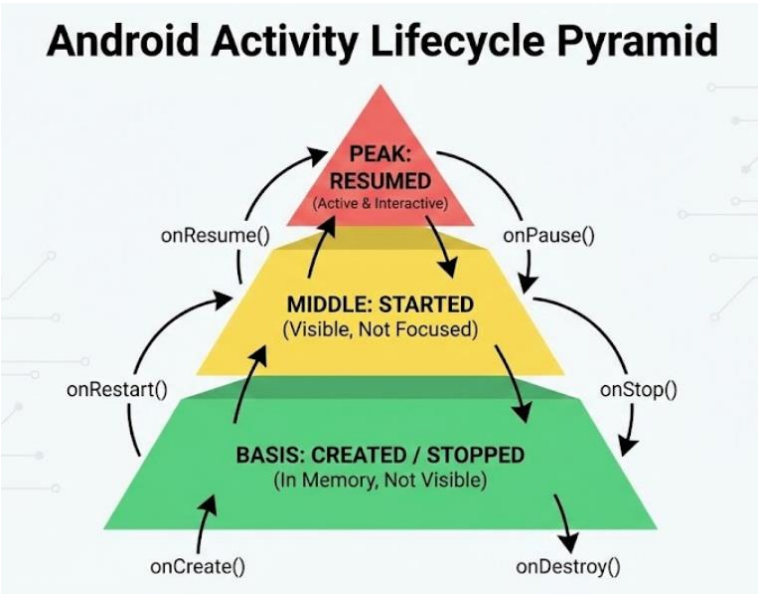
Bayangkan Activity itu seperti **Mahasiswa di Kelas**:

1. **onCreate() (Bangun Tidur & Persiapan):**
   Ini fase kelahiran. Mahasiswa bangun, mandi, dan pakai baju.
   Di sini kita melakukan inisialisasi awal: menghubungkan layout XML (setContentView) dan menyiapkan variabel.
2. **onStart() (Masuk Kelas):**
   Mahasiswa sudah sampai di kelas, sudah terlihat oleh dosen, tapi belum siap belajar (mungkin masih menaruh tas).
   Activity sudah terlihat di layar, tapi belum bisa berinteraksi dengan pengguna.
3. **onResume() (Mulai Belajar):**
   Mahasiswa duduk tegak, pulpen di tangan, siap mencatat.
   Activity berada di lapisan paling atas, terlihat penuh, dan pengguna bisa menekannya. Inilah kondisi aktif.
4. **onPause() (Terganggu Sebentar):**
   Tiba-tiba ada teman mengajak ngobrol, atau dosen keluar sebentar. Fokus terpecah.
   Terjadi jika Activity tertutup sebagian (misal ada pop-up dialog) atau pengguna akan pindah aplikasi. Data harus

segera disimpan di sini.

5. **onStop() (Kelas Selesai/Pulang):**
   Mahasiswa keluar ruangan. Dia masih ada di kampus, tapi tidak terlihat di kelas.
   Activity tidak lagi terlihat (misal pengguna menekan tombol Home). Sistem bisa mematikan Activity ini jika butuh memori.

6. **onDestroy() (Tidur Malam):**
   Hari berakhir. Semua energi direset.
   Activity dihancurkan sepenuhnya dari memori.



Gambar 4.1. Diagram Piramida Siklus Hidup Activity (Lifecycle)

Memahami urutan ini vital. Jangan sampai Anda memuat data berat (seperti unduh video) di onResume(), karena itu akan membuat aplikasi macet setiap kali layar menyala.

4.3. Praktik Menangani Siklus Hidup

Teori tanpa praktik hanya akan jadi angin lalu. Mari kita buktikan kapan metode-metode di atas dipanggil oleh sistem. Kita akan menggunakan alat diagnostik yang sudah kita pelajari di Bab 2: **Logcat**.

Buatlah proyek baru, lalu buka MainActivity.java. Kita akan melakukan *Override* (menimpa) metode-metode siklus hidup dan menyisipkan "pesan jejak".

Contoh Kode:

Java

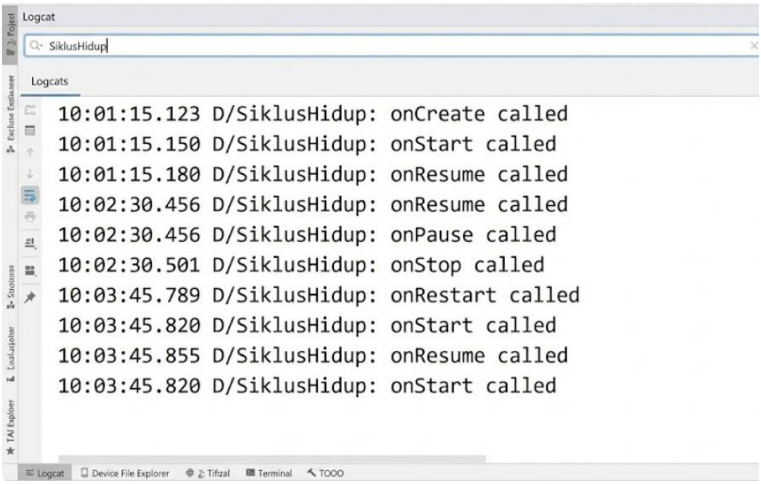
```
public class MainActivity extends AppCompatActivity {

    // Tag untuk memudahkan pencarian di Logcat
    String TAG = "SiklusHidup";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "Sedang di fase: onCreate");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, "Sedang di fase: onStart");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d(TAG, "Sedang di fase: onResume");
    }
```

```
    @Override
    protected void onPause() {
        super.onPause();
        Log.d(TAG, "Sedang di fase: onPause");
    }

    // ... tambahkan juga onStop dan onDestroy
}
```

Jalankan aplikasi, lalu buka panel **Logcat** di bagian bawah Android Studio. Ketik "SiklusHidup" di kolom pencarian.

1. Saat aplikasi dibuka, Anda akan melihat urutan: onCreate -> onStart -> onResume.
2. Tekan tombol **Home** di HP. Anda akan melihat: onPause -> onStop.
3. Buka kembali aplikasi dari *Recent Apps*. Anda akan melihat: onRestart -> onStart -> onResume.

Perhatikan bahwa onCreate tidak dipanggil lagi saat Anda membuka kembali aplikasi yang belum "mati". Inilah efisiensi Android.



Gambar 4.2. Tampilan Logcat yang menunjukkan urutan metode dipanggil

4.4. Menyimpan State Activity

Ada satu "cacat" unik (atau fitur?) di Android yang sering membuat pemula frustrasi: **Rotasi Layar**.

Saat Anda memutar HP dari *Portrait* ke *Landscape*, Android sebenarnya **menghancurkan** (onDestroy) Activity tersebut, lalu **membangun ulang** (onCreate) dari nol agar layout menyesuaikan dengan lebar layar baru.

Masalahnya: Data yang sedang diketik pengguna akan hilang!

Bayangkan pengguna sedang mengisi formulir panjang, lalu tidak sengaja memutar HP, dan *poof!* Isiannya kosong kembali. Itu pengalaman pengguna yang buruk.

Solusi: onSaveInstanceState

Kita harus menyelamatkan data sesaat sebelum Activity dihancurkan. Kita gunakan metode onSaveInstanceState. Bayangkan

ini seperti menitipkan tas di loker sebelum gedung direnovasi.

Langkah 1: Simpan Data

Java

```java
@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    super.onSaveInstanceState(outState);
    // Simpan teks ke dalam "loker" (Bundle)
    // Formatnya: KUNCI, NILAI
    outState.putString("data_nama", "Budi Santoso");
}
```

Langkah 2: Ambil Kembali Data (di onCreate)

Java

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Cek apakah ada barang titipan di loker?
    if (savedInstanceState != null) {
        String namaTersimpan =
savedInstanceState.getString("data_nama");
        // Tampilkan kembali ke layar...
    }
}
```

Dengan mekanisme Bundle ini, data penting akan selamat melewati badai rotasi layar.

Latihan dan Tugas

Untuk mengasah pemahaman Anda tentang "kehidupan" sebuah Activity, kerjakanlah tantangan berikut.

Latihan

1. Gunakan kode praktik di sub-bab 4.3.
2. Jalankan aplikasi di emulator atau HP fisik.
3. Lakukan skenario berikut dan catat urutan log yang muncul:
    - Tekan tombol *Back*.
    - Tekan tombol *Home*, lalu buka aplikasi lain (misal YouTube), lalu kembali ke aplikasi Anda.
    - Matikan layar HP (kunci layar), lalu nyalakan lagi.
    - Putar layar (*Rotate*).

Tugas

Buatlah sebuah aplikasi sederhana yang memiliki:

1. Satu EditText untuk input komentar.
2. Satu TextView untuk menampilkan jumlah karakter yang diketik (misal: "10/100 karakter").
3. **Tantangan:** Pastikan saat layar diputar (*rotate*), teks di dalam EditText **DAN** angka penghitung karakter di TextView tidak kembali ke 0. Gunakan onSaveInstanceState untuk menyimpan kedua data tersebut.

Referensi

1. Android Developers. (2025). *The Activity Lifecycle*. Google Developers. https://developer.android.com/guide/components/activities/activity-lifecycle
2. Big Nerd Ranch. (2023). *Android Programming: The Big*

Nerd Ranch Guide (7th ed.). Big Nerd Ranch Guides.

3. Hidayat, A. (2022). *Fundamental Android: Memahami Siklus Hidup Aplikasi*. Jakarta: Elex Media Komputindo.
4. Joshi, P. (2024). *Android 14 Application Development for Beginners*. Packt Publishing.
5. Meier, R. (2021). *Professional Android 4 Application Development*. Wrox (Wiley).

Bab 5
Intent: Menghubungkan Antar Layar

Bayangkan Anda memiliki rumah yang sangat besar, tetapi tidak memiliki pintu penghubung antar ruangan. Anda terjebak di ruang tamu selamanya. Membosankan, bukan?

Begitu juga dengan aplikasi Android. Aplikasi yang hanya memiliki satu layar (*Activity*) itu ibarat rumah tanpa pintu. Padahal, aplikasi yang baik adalah aplikasi yang dinamis: dari layar *Login*, pengguna pindah ke *Beranda*, lalu memilih produk masuk ke *Detail Produk*, dan berakhir di *Keranjang Belanja*.

Lalu, bagaimana cara berpindah dari satu layar ke layar lainnya? Di Android, kita tidak bisa memanggil Activity lain secara langsung seperti memanggil fungsi biasa. Kita membutuhkan perantara khusus. Perantara atau "kurir" inilah yang disebut **Intent**.

Dalam bab ini, kita akan belajar cara "membuka pintu" antar Activity, bahkan cara menyuruh aplikasi lain (seperti Browser atau Kamera) untuk bekerja demi aplikasi kita. Mari kita mulai.

5.1. Pengertian Intent

Secara harfiah, *Intent* berarti "Niat" atau "Maksud".

Dalam istilah teknis Android, Intent adalah objek pesan yang digunakan untuk meminta aksi dari komponen aplikasi lain. Bayangkan Intent sebagai sebuah **Amplop Surat**.

1. Anda menulis tujuan di amplop tersebut (Mau ke Activity mana?).
2. Anda memasukkan data tambahan ke dalam amplop (Data apa yang mau dibawa?).

3. Anda menyerahkan amplop itu ke sistem operasi Android.
4. Sistem Android membaca tujuannya dan meluncurkan komponen yang diminta.

Intent tidak hanya digunakan untuk berpindah layar (*startActivity*), tetapi juga untuk menjalankan layanan di latar belakang (*startService*) atau menyebarkan pengumuman ke seluruh sistem (*broadcast*). Namun, fokus kita saat ini adalah untuk navigasi layar.

5.2. Explicit Intent (Intent Eksplisit)

Sesuai namanya, *Explicit* berarti "Jelas" atau "Terang-terangan".

Kita menggunakan **Explicit Intent** ketika kita tahu persis alamat tujuan kita. Ini biasanya digunakan untuk navigasi di dalam aplikasi kita sendiri. Misalnya, dari MainActivity ingin pindah ke ProfileActivity. Kita tahu nama kelas tujuannya secara spesifik.

Analogi:

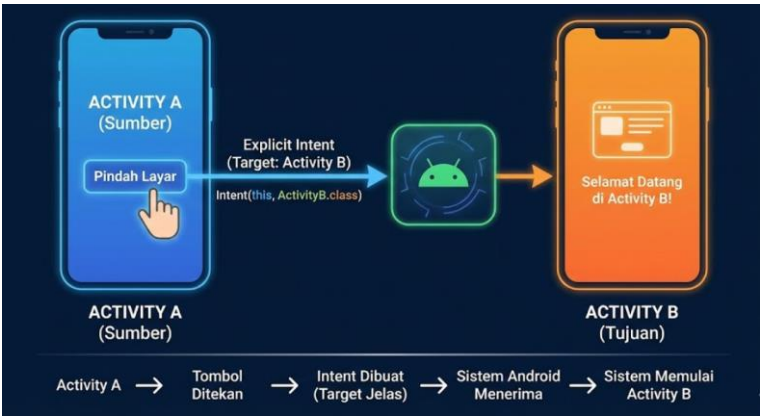
Seperti Anda memesan ojek online dan berkata: *"Tolong antar saya ke Gedung Fakultas Informatika Universitas X."* Tujuannya jelas dan spesifik.

Contoh Kode:

Java

```
// 1. Membuat objek Intent
// Parameter: (Context saat ini, Kelas Tujuan)
Intent intentPindah = new Intent(MainActivity.this,
ProfileActivity.class);

// 2. Menjalankan Intent
startActivity(intentPindah);
```

Kode di atas memerintahkan sistem: "Saya sekarang ada di MainActivity, tolong luncurkan ProfileActivity."



Gambar 5.1. Ilustrasi perpindahan layar menggunakan Explicit Intent

Perlu diingat, setelah startActivity dijalankan, Activity lama (MainActivity) tidak dihancurkan, melainkan hanya ditumpuk di bawah Activity baru. Jika pengguna menekan tombol *Back*, mereka akan kembali ke Activity lama.

5.3. Implicit Intent (Intent Implisit)

Bagaimana jika kita ingin membuka sebuah halaman web, tapi kita malas membuat browser sendiri? Atau kita ingin menelepon seseorang, tapi tidak mau membuat fitur *dialer* dari nol?

Di sinilah **Implicit Intent** berperan. Kita tidak menunjuk "siapa" yang mengerjakannya, tapi kita hanya menyebutkan "apa" yang ingin kita lakukan.

Analogi:

Seperti Anda berteriak di kerumunan: *"Tolong, ada yang dokter tidak?"*. Anda tidak peduli siapa dokternya (bisa Dokter A, Dokter B, atau Dokter C), yang penting dia bisa mengobati.

Sistem Android akan mengecek HP pengguna: "Aplikasi mana saja yang bisa membuka website?". Jika ada Chrome dan Firefox, Android akan memunculkan dialog pemilihan (*Chooser*) kepada pengguna.

Contoh Kode (Membuka Website):

Java

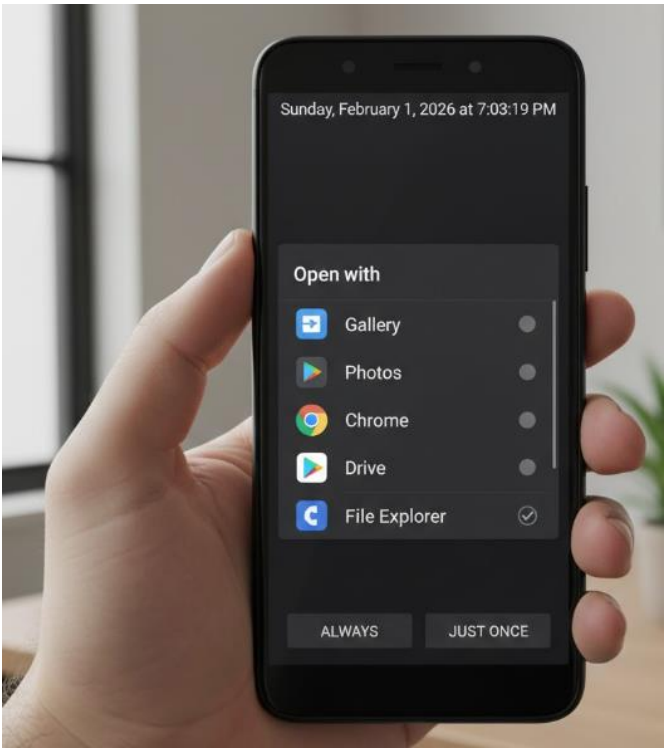
```
// Menentukan aksi: ACTION_VIEW (Melihat sesuatu)
Intent intentWeb = new Intent(Intent.ACTION_VIEW);

// Menentukan data: Alamat URL
intentWeb.setData(Uri.parse("https://www.google.com"));

// Jalankan
startActivity(intentWeb);
```

Tabel berikut merangkum perbedaan mendasar kedua jenis Intent ini:

Tabel 5.1. Perbedaan Explicit vs Implicit Intent

Fitur	**Explicit Intent**	**Implicit Intent**
Tujuan	**Spesifik.** Menyebutkan	**Umum.** Menyebutkan

	nama kelas tujuan.	aksi (*Action*) dan data.
Penggunaan	Navigasi internal dalam satu aplikasi.	Interaksi dengan aplikasi lain (Kamera, Map, Browser).
Keamanan	Lebih aman karena target terkunci.	Memerlukan pengecekan apakah ada aplikasi yang sanggup menangani.



Gambar 5.2. Dialog "Open With" yang muncul akibat Implicit Intent

5.4. Mengirim Data dengan Intent

Berpindah layar saja seringkali tidak cukup. Kita sering perlu membawa "oleh-oleh" atau data ke layar tujuan. Misalnya, dari layar *Login*, kita ingin mengirimkan "Nama User" untuk ditampilkan di layar *Beranda*.

Intent menyediakan metode putExtra() untuk menyimpan data. Konsepnya mirip dengan struktur data *Key-Value Pair* (Pasangan Kunci-Nilai).

Analogi:

Seperti menitipkan barang di loker. Anda menaruh barang (Data) dan mendapatkan nomor kunci (Key). Untuk mengambil barang itu nanti, Anda harus menggunakan nomor kunci yang sama persis.

Langkah 1: Pengirim (MainActivity)

Java

```
Intent intent = new Intent(MainActivity.this,
HomeActivity.class);

// Menyisipkan data ("KUNCI", "NILAI")
intent.putExtra("EXTRA_NAMA", "Budi Santoso");
intent.putExtra("EXTRA_UMUR", 20);

startActivity(intent);
```

Langkah 2: Penerima (HomeActivity)

Di Activity tujuan, kita membongkar "paket" tersebut, biasanya di dalam metode onCreate.

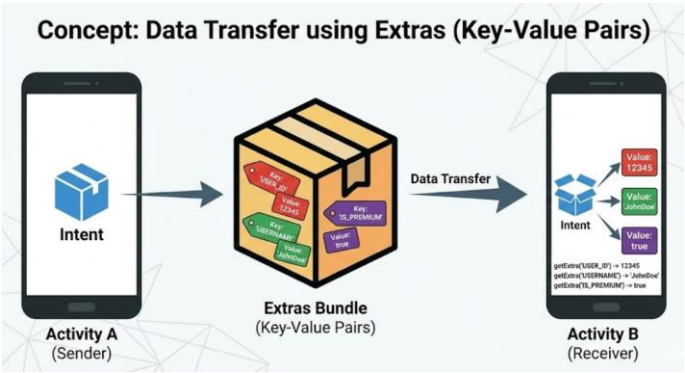
Java

```
// Ambil objek Intent yang mengirim kita ke sini
Intent intentDiterima = getIntent();

// Ambil datanya sesuai tipe data & kuncinya
String nama = intentDiterima.getStringExtra("EXTRA_NAMA");
int umur = intentDiterima.getIntExtra("EXTRA_UMUR", 0); // 0 adalah nilai default

// Tampilkan ke TextView...
```

Mengirim Objek Kompleks (Parcelable)

Bagaimana jika kita ingin mengirim satu objek Mahasiswa utuh (yang berisi nama, NIM, IPK)? Kita tidak bisa menggunakan putExtra biasa. Kita harus mengubah objek tersebut menjadi **Parcelable**.

Parcelable adalah mekanisme membongkar objek menjadi serpihan kecil agar bisa dikirim lewat Intent, lalu dirakit kembali di tujuan. Ini jauh lebih cepat daripada metode *Serializable* milik Java standar.



Gambar 5.3. Konsep pengiriman data menggunakan Extras (Key-Value)

Latihan dan Tugas

Mari kita praktikkan peran kita sebagai "kurir data" antar layar.

Latihan: Pintu Ajaib

1. Buatlah proyek baru dengan dua Activity: FirstActivity (sebagai launcher) dan SecondActivity.
2. Di FirstActivity, tambahkan sebuah tombol bertuliskan "Buka Layar Kedua".
3. Berikan kode pada tombol tersebut agar saat diklik, aplikasi berpindah ke SecondActivity menggunakan Explicit Intent.

Tugas: Formulir Pendaftaran Mini

Kembangkan latihan di atas menjadi aplikasi pendaftaran sederhana.

1. **Layar 1 (Input):**
   - Memiliki EditText untuk Nama.
   - Memiliki EditText untuk Usia (tipe *number*).
   - Tombol "Kirim Data".
2. **Layar 2 (Hasil):**
   - Memiliki TextView besar di tengah layar.
3. **Alur:**
   - Pengguna mengetik nama dan usia di Layar 1.
   - Saat tombol diklik, data dikirim menggunakan putExtra.
   - Di Layar 2, tangkap data tersebut dan tampilkan kalimat: *"Halo [Nama], usia Anda [Usia] tahun."*
4. **Tantangan:** Pastikan aplikasi tidak *crash* jika pengguna lupa mengisi salah satu kolom (gunakan validasi sederhana).

Referensi

1. Android Developers. (2025). *Intents and Intent Filters*. Google Developers. https://developer.android.com/guide/components/intents-

filters

2. Haryanto, T. (2022). *Pemrograman Android Modern dengan Java dan Kotlin*. Jakarta: Informatika.
3. Neil, T. (2022). *Beginning Android Programming with Android Studio* (4th ed.). Wrox.
4. Smyth, N. (2023). *Android Studio Iguana Essentials - Java Edition*. Payload Media.
5. Supardi, Y. (2021). *Koleksi Program Tugas Akhir dan Skripsi dengan Android*. Jakarta: Elex Media Komputindo.

Bab 6
RecyclerView

Bayangkan Anda membuka aplikasi Kontak di HP Anda. Di sana ada ratusan, mungkin ribuan nama teman dan kerabat. Saat Anda menggulir (*scroll*) layar ke bawah dengan cepat, gerakannya terasa mulus, bukan? Tidak ada kemacetan (*lag*) dan HP Anda tidak mendadak panas.

Tahukah Anda apa rahasianya? Apakah HP Anda memuat ribuan baris tampilan itu sekaligus ke dalam memori? Jawabannya: **Tidak**. Jika aplikasi memuat 1.000 tampilan sekaligus, memori HP akan habis dan aplikasi akan *crash*.

Rahasia di balik efisiensi ini bernama **RecyclerView**.

Sesuai namanya, komponen ini melakukan *Recycling* atau "Daur Ulang". Ia tidak menciptakan tampilan baru untuk setiap data. Ia hanya menciptakan sejumlah tampilan yang muat di layar (misal 10 baris), dan ketika Anda menggulir ke bawah, baris yang hilang di atas akan diambil, dibersihkan, dan digunakan kembali untuk menampilkan data baru di bawah.

Di bab ini, kita akan belajar membuat daftar canggih seperti WhatsApp atau Instagram. Siapkan fokus Anda, karena kita akan bermain dengan sedikit lebih banyak kode Java daripada biasanya.

6.1. Konsep RecyclerView

Dulu, Android menggunakan komponen bernama ListView. Itu adalah versi kuno yang boros memori. RecyclerView adalah versi modern yang jauh lebih efisien dan fleksibel.

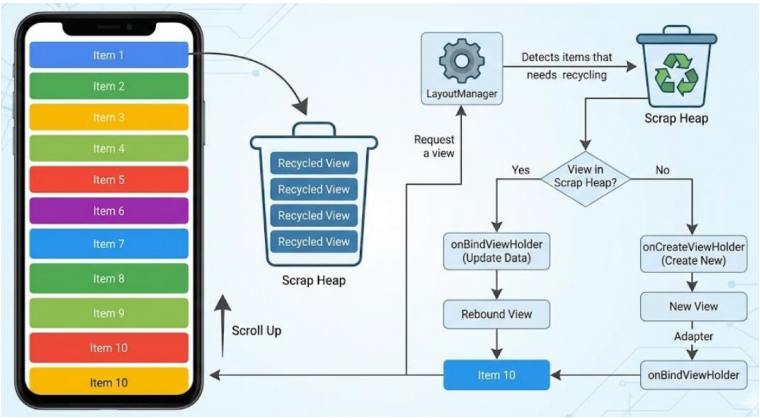
Untuk membangun sebuah RecyclerView, kita membutuhkan kerjasama tim dari tiga komponen utama. Mari kita gunakan analogi sebuah **Restoran**:

1. **Data (Menu Makanan):**
   Ini adalah daftar informasi yang ingin ditampilkan (misal: List<String> berisi nama teman).
2. **RecyclerView (Ruang Makan):**
   Area di layar tempat daftar ditampilkan.
3. **Adapter (Pelayan):**
   Ini adalah komponen paling kerja keras. Tugasnya menjembatani data dengan tampilan. Pelayan mengambil data dari dapur, menaruhnya di piring, dan menyajikannya ke meja.
4. **ViewHolder (Piring/Wadah):**
   Objek yang memegang referensi ke elemen visual (seperti TextView atau ImageView). Kenapa disebut *Holder*? Agar sistem tidak perlu "mencari piring" berulang kali setiap kali mau menyajikan makanan.
5. **LayoutManager (Manajer Tata Letak):**
   Dia yang mengatur susunan meja. Apakah mau disusun berbaris ke bawah (*Linear*)? Atau kotak-kotak seperti papan catur (*Grid*)?

Tabel 6.1. Perbedaan ListView (Lama) vs RecyclerView (Baru)

Fitur	**ListView**	**RecyclerView**
Pola View	Selalu buat baru (Boros).	Daur ulang View (Hemat).
Kustomisasi	Sulit dan kaku.	Sangat fleksibel.

Animasi	Minim dukungan.	Dukungan animasi bawaan.
Susunan	Hanya vertikal.	Vertikal, Horizontal, Grid.



Gambar 6.1. Ilustrasi mekanisme daur ulang (Recycling) tampilan

6.2. Implementasi RecyclerView Sederhana

Jujur saja, membuat RecyclerView pertama kali memang terasa agak panjang langkahnya dibandingkan sekadar membuat Button. Tapi percayalah, hasilnya sepadan.

Berikut adalah resep langkah demi langkahnya:

Langkah 1: Tambahkan RecyclerView di Layout

Buka activity_main.xml, dan tambahkan wadahnya.

XML

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvDaftarNama"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Langkah 2: Desain Tampilan Per Item

Kita perlu mendesain bagaimana bentuk **satu baris** data. Buat file layout baru item_nama.xml.

XML

```
<LinearLayout                          ...                          >
    <TextView
        android:id="@+id/tvNama"
        android:textSize="18sp"
        ...                                                         />
</LinearLayout>
```

Langkah 3: Membuat Adapter (Bagian Tersulit)

Buat kelas Java baru, misal NamaAdapter.java. Kelas ini harus mewarisi RecyclerView.Adapter. Anda wajib mengimplementasikan 3 metode sakti:

1. **onCreateViewHolder**: "Koki, tolong ambilkan piring kosong baru!" (Membuat tampilan baris baru jika belum ada).
2. **onBindViewHolder**: "Pelayan, tolong isi piring ini dengan Masakan A!" (Mengisi data ke dalam tampilan).
3. **getItemCount**: "Berapa total pesanan?" (Menghitung jumlah data).

Java

```
public class NamaAdapter extends
RecyclerView.Adapter<NamaAdapter.ViewHolder> {

    private List<String> dataNama;

    // Konstruktor untuk menerima data
    public NamaAdapter(List<String> data) {
        this.dataNama = data;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        // Mengubah XML menjadi Objek View (Inflate)
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_nama, parent, false);
        return new ViewHolder(view);
    }


    @Override
    public void onBindViewHolder(ViewHolder holder, int
position) {
        // Mengisi data ke view
        String nama = dataNama.get(position);
        holder.tvNama.setText(nama);
    }

    @Override
    public int getItemCount() {
        return dataNama.size();
    }

    // Kelas Penampung (ViewHolder)
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView tvNama;
        public ViewHolder(View itemView) {
            super(itemView);
```

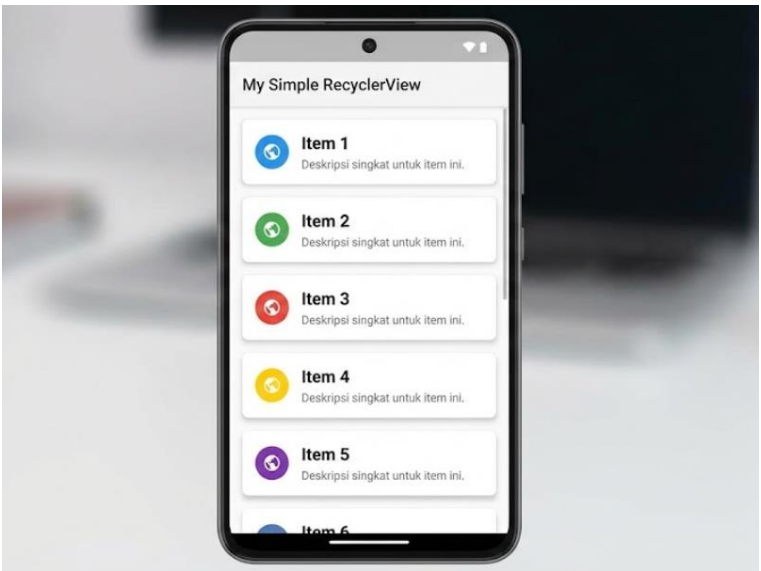
```
            tvNama = itemView.findViewById(R.id.tvNama);
        }
    }
}
```

Langkah 4: Pasang di Activity

Terakhir, kita hubungkan semuanya di MainActivity.java.

Java

```
RecyclerView rv = findViewById(R.id.rvDaftarNama);
// Siapkan data
List<String> teman = new ArrayList<>();
teman.add("Budi");
teman.add("Siti");
// ... tambah data lain

// Pasang Adapter
NamaAdapter adapter = new NamaAdapter(teman);
rv.setAdapter(adapter);

// Tentukan Layout Manager (PENTING!)
rv.setLayoutManager(new LinearLayoutManager(this));
```



Gambar 6.2. Hasil tampilan RecyclerView sederhana

6.3. Mengelola Data Dinamis

Daftar statis itu membosankan. Bagaimana jika ada teman baru? Atau ada yang dihapus?

Kita tidak perlu memuat ulang seluruh layar. Kita cukup memberitahu Adapter bahwa ada perubahan data.

Adapter memiliki metode notifikasi khusus:

- notifyDataSetChanged(): "Perhatian semua! Data berubah total!" (Cek ulang semua, agak berat).
- notifyItemInserted(int posisi): "Ada data baru masuk di posisi X." (Lebih efisien, ada animasi).
- notifyItemRemoved(int posisi): "Data di posisi X dihapus."

Contoh Menambah Data:

```
Java
```

```java
// Tambah data ke List
teman.add("Doni");

// Beritahu adapter ada barang baru di posisi terakhir
adapter.notifyItemInserted(teman.size() - 1);
```

Dengan cara ini, RecyclerView akan menampilkan animasi selip (*slide in*) yang cantik secara otomatis.

6.4. Variasi Tampilan (Grid & Staggered)

Ingin mengubah tampilan daftar baris menjadi kotak-kotak seperti galeri foto Instagram?

Di sinilah kehebatan RecyclerView. Anda tidak perlu menyentuh Adapter atau Data. Anda cukup mengganti **LayoutManager** di Activity.

GridLayoutManager

Membagi layar menjadi kolom-kolom.

```
Java
```

```java
// Membuat Grid dengan 2 kolom
rv.setLayoutManager(new GridLayoutManager(this, 2));
```

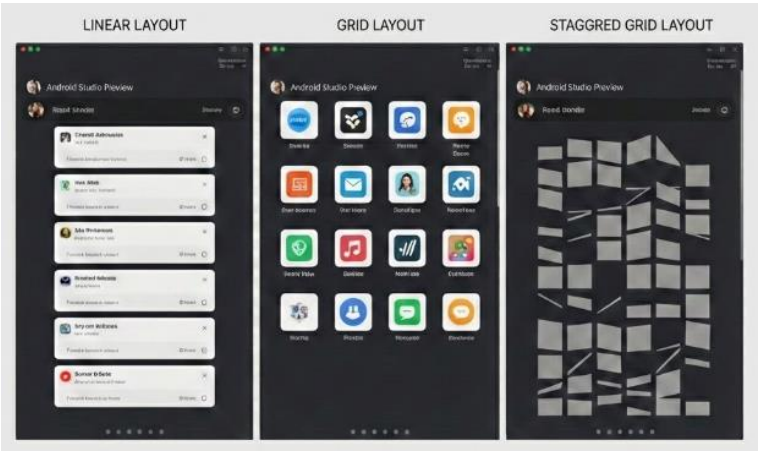
StaggeredGridLayoutManager

Pernah lihat Pinterest? Tinggi setiap kotaknya berbeda-beda (batu bata tidak rata). Itulah *Staggered Grid*.

```
Java
```

```
rv.setLayoutManager(new StaggeredGridLayoutManager(
    2, StaggeredGridLayoutManager.VERTICAL));
```

Hanya dengan mengubah satu baris kode, seluruh nuansa aplikasi Anda berubah drastis.



Gambar 6.3. Perbandingan tampilan Linear, Grid, dan Staggered

Latihan dan Tugas

Untuk menguasai RecyclerView, kuncinya adalah: Mengetik Ulang. Jangan *copy-paste* kode Adapter, tapi ketiklah agar jari Anda hafal strukturnya.

Latihan: Daftar Teman

1. Buatlah proyek baru.
2. Implementasikan RecyclerView sederhana yang menampilkan 10 nama teman dekat Anda.
3. Gunakan LinearLayoutManager.

Tugas: Buku Telepon Interaktif

Tingkatkan latihan di atas menjadi lebih kompleks:

1. **Data:** Jangan hanya String. Buatlah kelas model Kontak yang berisi String nama dan String nomorHp.
2. **Tampilan:** Ubah item_layout agar menampilkan Nama (tebal) dan Nomor HP (kecil di bawahnya).
3. **Interaksi:** Tambahkan *event listener* di Adapter. Ketika salah satu baris diklik, tampilkan **Toast** (pesan pop-up singkat) yang berisi: *"Memanggil [Nama]..."*.

Referensi

1. Android Developers. (2025). *Create dynamic lists with RecyclerView*. Google Developers. https://developer.android.com/develop/ui/views/layout/recyclerview
2. Dawn, M. (2022). *Android UI Design: Planning and Building Interfaces*. Pearson.
3. Haryanto, T. (2023). *Pemrograman Android Modern: Studi Kasus Aplikasi Marketplace*. Jakarta: Informatika.
4. Kurniawan, D. (2021). *Membangun Aplikasi Android Efisien dengan RecyclerView*. Jurnal Sistem Informasi, 5(2), 112-120.
5. Phillips, B., & Stewart, C. (2024). *Android Programming: The Big Nerd Ranch Guide* (8th ed.). Big Nerd Ranch Guides.

Bab 7
Menyimpan Data Lokal SharedPreferences dan SQLite

Pernahkah Anda memainkan sebuah *game*, bersusah payah mencapai level tertinggi, lalu mematikan HP? Saat Anda membukanya kembali besok, level Anda masih tersimpan aman. Bayangkan betapa frustrasinya jika setiap kali membuka *game*, Anda harus mulai lagi dari Level 1.

Itulah perbedaan mendasar antara **Memori (RAM)** dan **Penyimpanan (Storage)**.

Variabel yang kita pelajari di bab-bab sebelumnya disimpan di RAM. Sifatnya sementara, ibarat tulisan kapur di papan tulis kelas; begitu kelas selesai (aplikasi ditutup) atau lampu mati (HP mati), tulisan itu hilang tak berbekas.

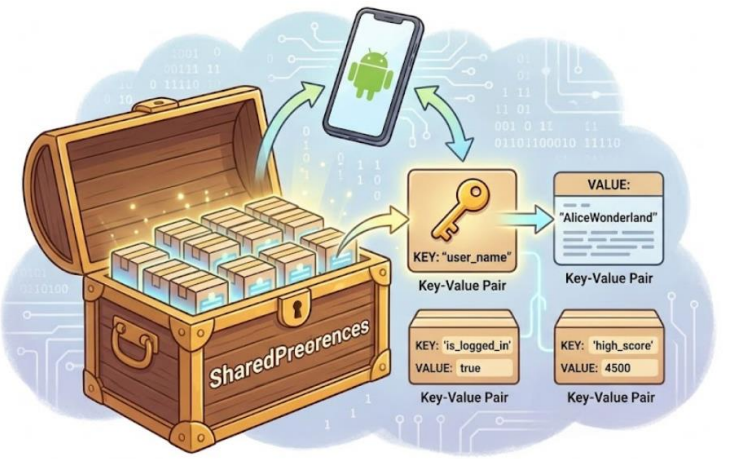
Di bab ini, kita akan belajar cara menulis di "Buku Catatan" yang tintanya permanen. Kita akan mempelajari dua mekanisme penyimpanan lokal paling dasar di Android: **SharedPreferences** untuk catatan kecil (seperti pengaturan suara *on/off*), dan **SQLite** untuk catatan besar yang terstruktur (seperti daftar ribuan kontak).

7.1. SharedPreferences: Si Catatan Tempel

Mari kita mulai dari yang paling sederhana. **SharedPreferences** adalah mekanisme penyimpanan data berjenis *Key-Value Pair* (Pasangan Kunci-Nilai).

Bayangkan SharedPreferences seperti **Catatan Tempel (*Sticky Notes*)** di pintu kulkas Anda. Kapasitasnya kecil, tapi sangat cepat dan mudah diakses. Anda tidak akan menulis novel di kertas *sticky notes*, bukan? Anda hanya menulis hal-hal ringkas seperti:

- "Beli Susu" (Kunci: Belanja, Nilai: Susu)
- "Lampu Ruang Tamu: Nyala" (Kunci: Lampu, Nilai: True)



Gambar 7.1. Ilustrasi konsep Key-Value Pair pada SharedPreferences

Di Android, SharedPreferences sangat cocok untuk menyimpan data sederhana, antara lain:

1. **Status Login,** Apakah user sudah login? Ya/Tidak.
2. **Pengaturan Aplikasi,** Mode Gelap, Ukuran Font, Notifikasi.
3. **Skor Tertinggi,** High Score game sederhana.

Prinsip kerjanya sederhana. Anda menyimpan data dengan label (Kunci), dan saat butuh, Anda panggil labelnya untuk mendapatkan isinya (Nilai).

7.2. Praktik SharedPreferences

Mari kita praktikkan cara menyimpan nama pengguna agar aplikasi bisa menyapa "Halo, Budi!" setiap kali dibuka, meskipun aplikasi sudah pernah dimatikan sebelumnya.

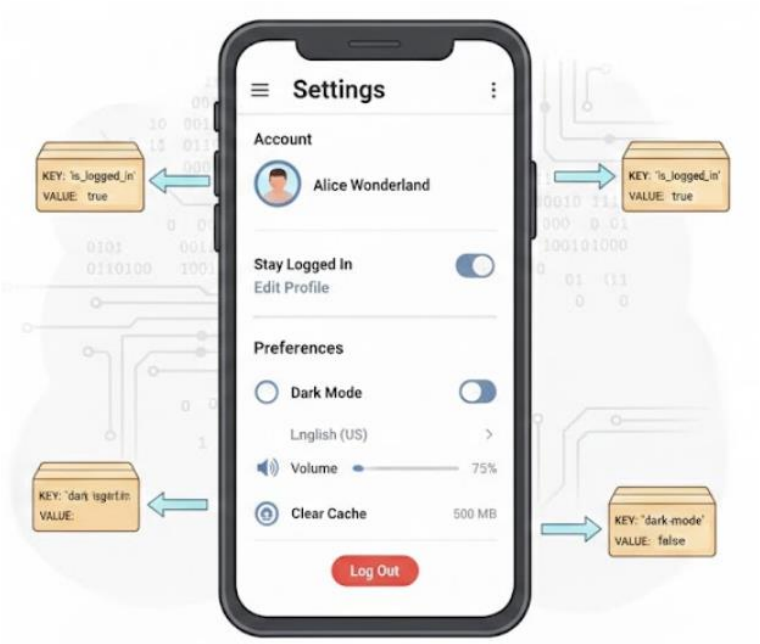
Ada dua tahap utama: **Menulis (Save)** dan **Membaca (Load)**.

Langkah 1: Menulis Data (Save)

Untuk menulis, kita membutuhkan objek bantu bernama Editor.

Java

```
// 1. Panggil file SharedPreferences (misal namanya "DataPengguna")
// MODE_PRIVATE artinya hanya aplikasi ini yang boleh membaca file ini
SharedPreferences sharedPref = getSharedPreferences("DataPengguna", MODE_PRIVATE);

// 2. Siapkan Editor (Pena untuk menulis)
SharedPreferences.Editor editor = sharedPref.edit();

// 3. Masukkan data (Kunci, Nilai)
editor.putString("nama_user", "Budi Santoso");
editor.putBoolean("is_login", true);

// 4. Simpan perubahan (PENTING!)
// apply() menyimpan di latar belakang (aman), commit() memblokir alur utama.
editor.apply();
```



Gambar 7.2. Contoh antarmuka pengaturan (Settings) yang datanya **disimpan di** SharedPreferences

Langkah 2: Membaca Data (Load)

Saat membaca, kita tidak butuh Editor. Kita cukup panggil kuncinya. Kita juga wajib menyediakan "Nilai Default" sebagai cadangan jika data belum pernah disimpan sebelumnya.

Java

```
SharedPreferences sharedPref =
getSharedPreferences("DataPengguna", MODE_PRIVATE);

// Ambil String dengan kunci "nama_user"
// Parameter kedua adalah nilai default jika data tidak
ditemukan
String nama = sharedPref.getString("nama_user", "Pengguna
```

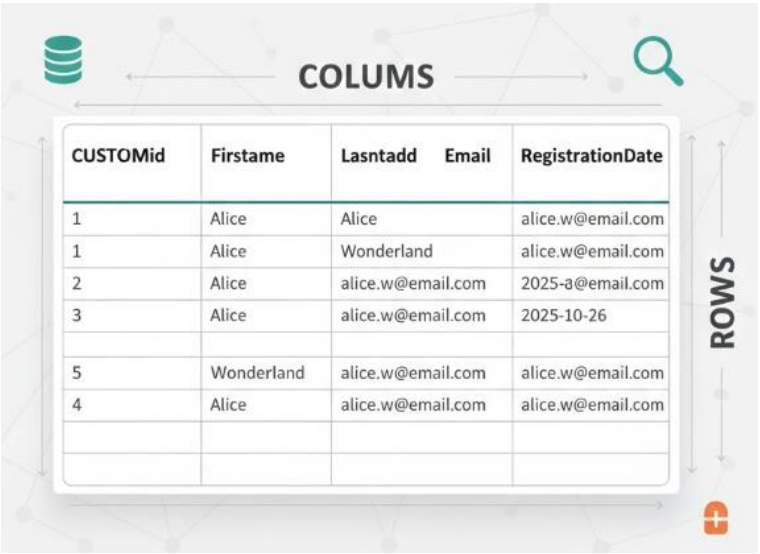
```
Baru");

// Hasil:
// Jika sudah disimpan -> keluar "Budi Santoso".
// Jika belum -> keluar "Pengguna Baru".
```

Sangat sederhana, bukan? Namun ingat, jangan simpan data besar (seperti foto atau ribuan teks) di sini, karena akan memperlambat performa aplikasi.

7.3. Pengenalan SQLite

Bagaimana jika kita ingin menyimpan daftar mahasiswa satu universitas lengkap dengan NIM, Jurusan, dan IPK? Menggunakan SharedPreferences akan sangat berantakan dan sulit dikelola. Kita butuh lemari arsip yang rapi. Kita butuh **Database**.

Android sudah dibekali dengan **SQLite**, sebuah mesin database relasional (*RDBMS*) yang sangat ringan namun bertenaga. Database ini tersimpan dalam sebuah file tunggal di dalam memori internal HP.



Gambar 7.3. Struktur tabel database yang terdiri dari Baris (Row) dan Kolom (Column)

Jika Anda pernah belajar MySQL atau basis data di semester awal, konsepnya sama persis:

- **Database:** Gedung arsipnya.
- **Table:** Lemari/Rak arsip (misal: Tabel Mahasiswa).
- **Column (Field):** Jenis data (Nama, NIM, Alamat).
- **Row (Record):** Satu baris data mahasiswa lengkap.

Untuk berinteraksi dengan SQLite di Android, kita tidak mengakses file databasenya langsung. Kita dibantu oleh kelas "Asisten Manajer" bernama SQLiteOpenHelper. Tugas asisten ini adalah:

1. Membuat database jika belum ada (onCreate).
2. Memperbarui struktur tabel jika versi aplikasi naik (onUpgrade).

7.4. Operasi Dasar SQLite (CRUD)

Mari kita buat sistem pengelolaan data teman sederhana. Kita akan melakukan operasi **CRUD** (*Create, Read, Update, Delete*).

Tahap 1: Membuat Helper

Buat kelas Java baru, misal DatabaseHelper.java.

Java

```
public class DatabaseHelper extends SQLiteOpenHelper {

    // Nama Database dan Versi
    private static final String DB_NAME = "TemanDB";
    private static final int DB_VERSION = 1;

    // Konstruktor
    public DatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    // Dipanggil saat DB pertama kali dibuat
    @Override
    public void onCreate(SQLiteDatabase db) {
        // Query SQL: Membuat tabel dengan kolom ID (Otomatis)
dan NAMA
        String sql = "CREATE TABLE teman (id INTEGER PRIMARY
KEY AUTOINCREMENT, nama TEXT)";
        db.execSQL(sql);
    }

    // Dipanggil saat versi DB berubah (misal developer
menambah kolom baru)
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS teman");
        onCreate(db);
```

```
    }
}
```

Tahap 2: Create (Menambah Data)

Kita menggunakan objek ContentValues, yang mirip dengan map (pasangan nama kolom dan isinya).

Java

```
public void tambahTeman(String namaBaru) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    // Masukkan data ke dalam penampung
    values.put("nama", namaBaru);

    // Masukkan ke tabel 'teman'
    db.insert("teman", null, values);
    db.close();
}
```

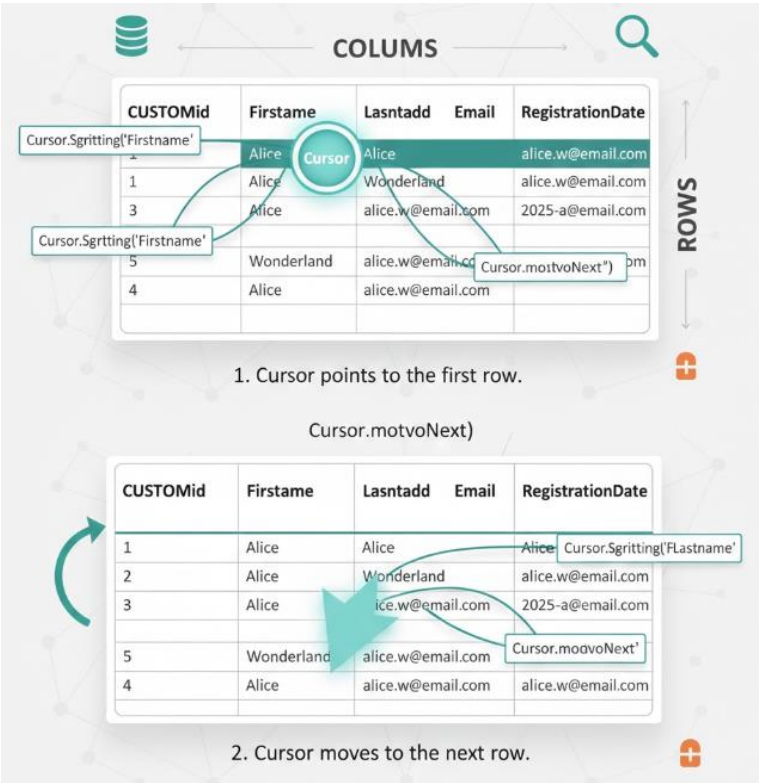
Tahap 3: Read (Membaca Data)

Hasil pembacaan database di Android dikembalikan dalam bentuk objek **Cursor**. Bayangkan Cursor sebagai jari telunjuk yang menunjuk baris demi baris data dari hasil *query*. Jari tersebut akan bergerak dari baris pertama hingga baris terakhir.

Java

```
public ArrayList<String> ambilSemuaTeman() {
    ArrayList<String> daftar = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
```

```
    // Query SQL: SELECT * FROM teman
    Cursor cursor = db.rawQuery("SELECT * FROM teman", null);

    // Pindahkan kursor ke baris pertama, lalu loop sampai
habis
    if (cursor.moveToFirst()) {
        do {
            // Ambil data di kolom index ke-1 (kolom 'nama')
            String nama = cursor.getString(1);
            daftar.add(nama);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return daftar;
}
```



Gambar 7.4. Ilustrasi cara kerja Cursor menelusuri data baris demi baris

Latihan dan Tugas

Materi database memang membutuhkan ketelitian tinggi karena satu kesalahan penulisan nama tabel (*typo*) bisa menyebabkan aplikasi berhenti (*crash*).

Latihan: Pengaturan Mode Malam

Buatlah satu halaman pengaturan sederhana yang berisi satu tombol: **"Aktifkan Mode Gelap"**.

1. Saat tombol ditekan, simpan status boolean (true/false) ke **SharedPreferences**.
2. Saat aplikasi dibuka kembali (di metode onCreate), cek status tersebut. Jika bernilai true, ubah latar belakang aplikasi menjadi warna hitam.

Tugas: Buku Telepon Abadi

Ini adalah tugas lanjutan dari Bab 6 (RecyclerView).

Aplikasi Buku Telepon Anda sebelumnya datanya hilang saat aplikasi ditutup, bukan? Sekarang, ubahlah agar menggunakan **SQLite**.

1. Buat tombol "Simpan" untuk menambah teman ke database SQLite (Panggil metode insert).
2. Ubah Adapter RecyclerView agar mengambil data dari ArrayList yang diisi oleh hasil *query* SQLite (ambilSemuaTeman).
3. **Hasil Akhir:** Tambahkan 3 nama teman, tutup aplikasi (buang dari *Recent Apps*), lalu buka lagi. Nama-nama tersebut harus tetap ada di layar.

Referensi

1. Android Developers. (2025). *Save data using SQLite*. Google Developers. https://developer.android.com/training/data-storage/sqlite
2. Android Developers. (2024). *Save key-value data*. Google Developers. https://developer.android.com/training/data-storage/shared-preferences
3. Haryanto, T. (2023). *Manajemen Data Lokal Android: SQLite dan Room Persistence*. Jakarta: Informatika.

4. Neil, T. (2022). *Beginning Android Programming with Android Studio* (Chapter on Databases). Wrox.
5. Supardi, Y. (2022). *Semua Bisa Menjadi Programmer Android Studio*. Jakarta: Elex Media Komputindo.

Bab 8
Berinteraksi dengan Jaringan: Volley

Coba perhatikan aplikasi-aplikasi yang sering Anda buka di HP saat ini: Instagram, Gojek, Tokopedia, hingga Mobile Banking. Apa kesamaan mendasar dari semua aplikasi tersebut?

Benar, semuanya **membutuhkan internet**.

Aplikasi tanpa internet ibarat perpustakaan pribadi yang pintunya terkunci rapat; isinya mungkin bagus, tapi tidak akan pernah bertambah atau berubah. Sebaliknya, aplikasi yang terhubung ke internet adalah jendela dunia. Ia bisa menampilkan cuaca terkini, harga saham *real-time*, hingga memungkinkan kita mengobrol dengan teman di benua lain.

Namun, menghubungkan aplikasi Android ke internet tidak sesederhana mencolokkan kabel. Ada protokol yang harus dipatuhi, ada antrean data yang harus diatur, dan ada bahasa asing (JSON) yang harus diterjemahkan.

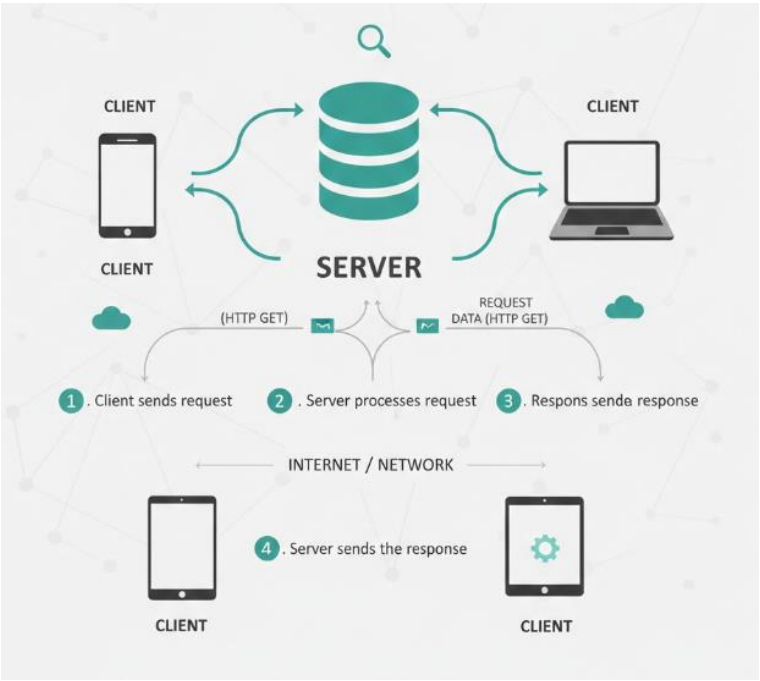
Di bab ini, kita akan berkenalan dengan asisten setia kita bernama **Volley**. Dia adalah pustaka (*library*) buatan Google yang bertugas mengurus semua kerumitan lalu lintas jaringan, sehingga Anda bisa fokus pada urusan tampilan dan logika aplikasi. Mari kita mulai berselancar!

8.1. Konsep Komunikasi Jaringan

Sebelum menulis kode, mari kita pahami dulu bagaimana internet bekerja dalam konteks aplikasi *mobile*.

Kita menggunakan model **Client-Server**.

- **Client (Aplikasi Anda):** Pihak yang meminta informasi. Contoh: "Tolong berikan data cuaca hari ini."
- **Server (Penyedia):** Komputer super kuat di suatu tempat (misal milik Google atau BMKG) yang melayani permintaan tersebut.



Gambar 8.1. Ilustrasi arsitektur Client-Server dalam komunikasi data

Mengapa Volley?

Di Android, melakukan koneksi internet di "Jalur Utama" (*Main Thread*) adalah tindakan ilegal. Jika Anda nekat melakukannya, aplikasi akan macet total dan muncul pesan horor *Application Not Responding* (ANR). Anda wajib melakukannya di jalur latar belakang (*Background Thread*).

Masalahnya, membuat jalur latar belakang secara manual itu rumit dan rawan kesalahan. Di sinilah **Volley** hadir sebagai pahlawan.

Volley ibarat **Manajer Logistik**. Anda cukup berikan perintah: "Ambil data di alamat X!" lalu Volley yang akan:

1. Membuka jalur koneksi.
2. Mengantrekan permintaan jika internet lambat.
3. Mengirimkan hasilnya kembali ke tangan Anda saat sudah siap. Semua terjadi secara otomatis dan sangat cepat.

Persiapan Awal

Sebelum menggunakan Volley, ada dua "tiket masuk" yang wajib disiapkan:

1. Izin Internet (Manifest)

Buka AndroidManifest.xml dan tambahkan izin ini di atas tag <application>:

```
XML
```

```
<uses-permission android:name="android.permission.INTERNET" />
```

2. Tambah Pustaka (Gradle)

Buka build.gradle (Module: app) dan tambahkan di bagian *dependencies*:

```
Gradle
```

```
implementation 'com.android.volley:volley:1.2.1'
```

8.2. Permintaan GET

Dalam dunia web, ada dua cara utama berkomunikasi yaitu, GET dan POST.

GET artinya "**Mengambil**". Kita meminta data dari server tanpa mengubah apapun di server tersebut. Contoh: Melihat daftar berita, cek cuaca, atau mencari produk di toko *online*.

Mari kita coba mengambil data dari sebuah API (Antarmuka Pemrograman Aplikasi). API adalah pelayan restoran yang mencatat pesanan kita dan memberikannya ke dapur (Server).

Langkah-langkah Coding:

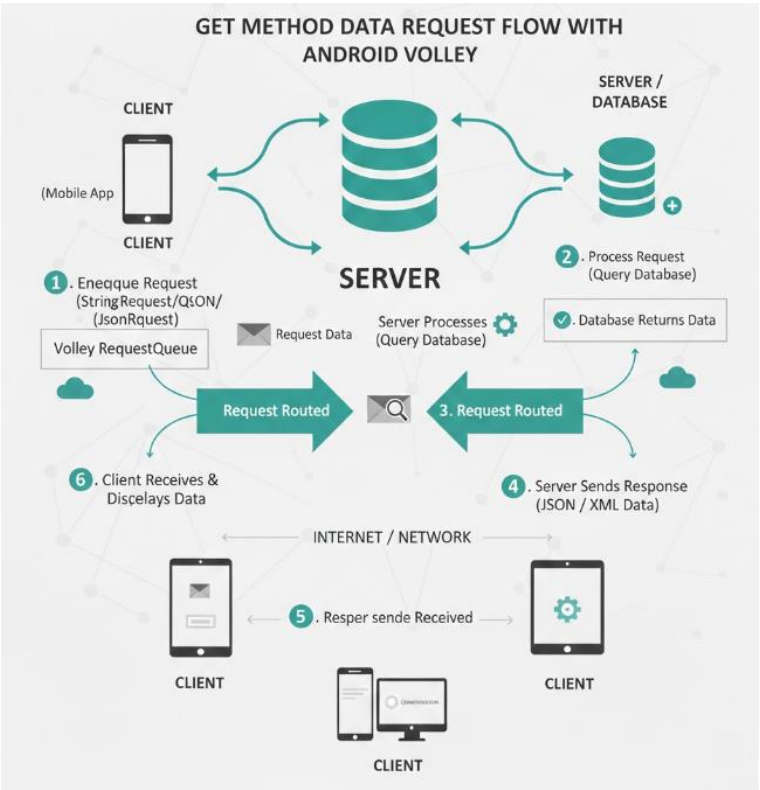
1. **Buat Antrean (RequestQueue):** Siapkan jalur pipa antrean Volley.
2. **Buat Permintaan (StringRequest):** Tentukan alamat URL tujuan dan apa yang harus dilakukan saat sukses atau gagal.
3. **Masukkan ke Antrean:** Kirim permintaan tersebut agar segera diproses.

Contoh Kode:

Java

```
// 1. URL tujuan (Misal: API Google)
String url = "https://www.google.com";

// 2. Siapkan Antrean
RequestQueue queue = Volley.newRequestQueue(this);

// 3. Buat Request
StringRequest stringRequest = new StringRequest(
    Request.Method.GET, // Metode GET
    url,                // Alamat Tujuan
```

```
    new Response.Listener<String>() { // Listener Jika SUKSES
        @Override
        public void onResponse(String response) {
            // Tampilkan 500 karakter pertama dari hasil
            tvHasil.setText("Response: " +
response.substring(0,500));
        }
    },
    new Response.ErrorListener() { // Listener Jika GAGAL
        @Override
        public void onErrorResponse(VolleyError error) {
            tvHasil.setText("Gagal koneksi! Cek internet
Anda.");
        }
    }
);

// 4. Masukkan ke antrean (Jalan!)
queue.add(stringRequest);
```



Gambar 8.2. Alur permintaan data menggunakan metode GET

8.3. Permintaan POST

Jika GET hanya "melihat", maka **POST** artinya **"Mengirim"** atau **"Menempelkan"**.

Kita menggunakan POST saat ingin mengirim data pribadi ke server untuk diproses atau disimpan. Contoh: *Login* (kirim *username* & *password*), Daftar Akun Baru, atau Posting Status Media Sosial.

Perbedaannya dengan GET terletak pada "paket" yang dibawa. Pada POST, kita menyisipkan parameter data di dalam tubuh permintaan

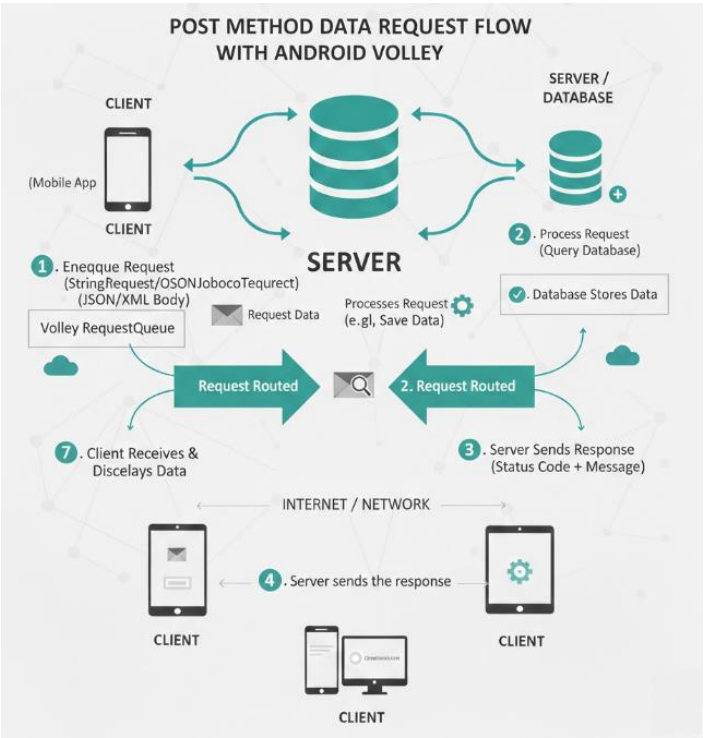
(*body*), bukan ditempel di URL. Ini membuat data lebih aman dan tidak terlihat langsung di alamat web.

Di Volley, kita perlu melakukan *Override* pada metode getParams().

Contoh Kode:

Java

```
String url = "https://api.contoh.com/login";

StringRequest postRequest = new StringRequest(
    Request.Method.POST,
    url,
    new Response.Listener<String>() { ... }, // Listener Sukses
    new Response.ErrorListener() { ... }     // Listener Gagal
) {
    // AREA INI UNTUK MENYISIPKAN DATA
    @Override
    protected Map<String, String> getParams() {
        // Map adalah struktur Kunci-Nilai
        Map<String, String> params = new HashMap<>();
        params.put("username", "budi123");
        params.put("password", "rahasia");
        return params;
    }
};

queue.add(postRequest);
```



Gambar 8.3. Alur permintaan data menggunakan metode POST

8.4. Menangani JSON

Saat server menjawab permintaan kita, mereka jarang menjawab dengan kalimat bahasa manusia seperti "Halo Budi, cuaca hari ini cerah."

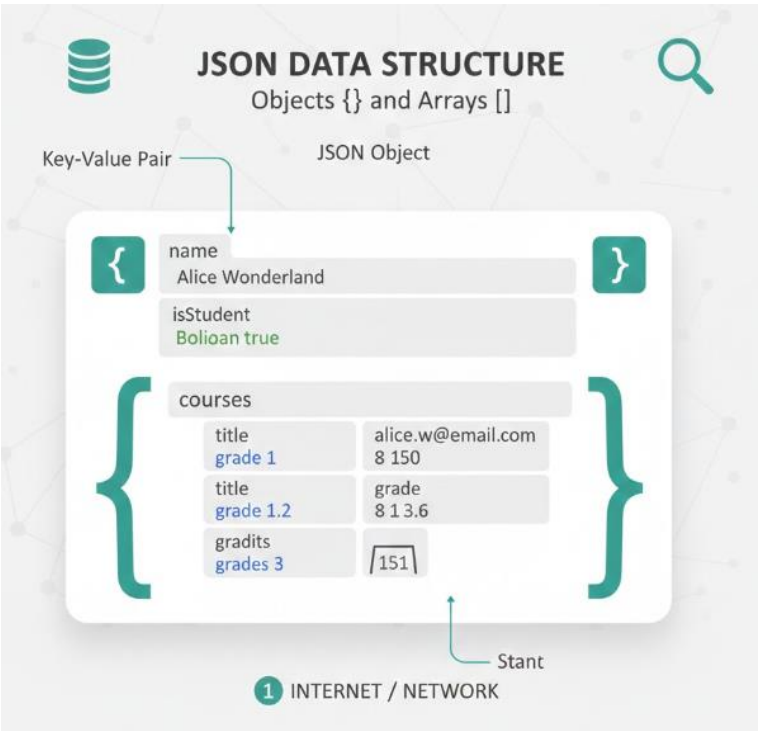
Mereka menjawab menggunakan bahasa baku komputer yang disebut **JSON (JavaScript Object Notation)**.

JSON adalah format teks yang ringan, ringkas, dan mudah dibaca oleh programmer. Formatnya sangat mirip dengan struktur data di Java.

Contoh Data JSON:

Json

```
{
  "kota": "Yogyakarta",
  "suhu": 30,
  "cuaca": ["Cerah", "Berawan"]
}
```



Gambar 8.4. Struktur data JSON yang terdiri dari Objek {} dan Array []

Tugas kita di Android adalah melakukan **Parsing** (Mengurai/Menerjemahkan) teks JSON tersebut menjadi variabel Java agar bisa ditampilkan di TextView.

Komponen Utama Parsing:

1. **JSONObject:** Merepresentasikan data dalam kurung kurawal { }.
2. **JSONArray:** Merepresentasikan data daftar dalam kurung siku [ ].

Contoh Kode Parsing:

Java

```
// Anggaplah variabel 'response' berisi teks JSON di atas
try {
    // 1. Ubah String mentah menjadi JSONObject
    JSONObject jsonObject = new JSONObject(response);

    // 2. Ambil data sesuai kuncinya
    String namaKota = jsonObject.getString("kota");
    int suhuUdara = jsonObject.getInt("suhu");

    // 3. Tampilkan ke layar
    tvKota.setText(namaKota);
    tvSuhu.setText(String.valueOf(suhuUdara) + " Celcius");

} catch (JSONException e) {
    e.printStackTrace();
    Toast.makeText(this, "Gagal membaca format JSON",
Toast.LENGTH_SHORT).show();
}
```

Ingat, proses *parsing* wajib dibungkus dalam blok try-catch karena selalu ada risiko format data dari server rusak atau tidak sesuai, yang

bisa menyebabkan aplikasi *crash*.

Latihan dan Tugas

Bekerja dengan jaringan seringkali membutuhkan kesabaran ekstra karena faktor sinyal internet yang tidak stabil. Jangan menyerah jika terjadi *error*, bacalah pesan kesalahannya di Logcat.

Latihan: Cek Cuaca Sederhana

Kita akan menggunakan API publik gratisan untuk latihan.

1. Daftar akun gratis di layanan API Cuaca (seperti OpenWeatherMap) untuk mendapatkan API Key.
2. Buat aplikasi dengan satu tombol "Cek Cuaca".
3. Gunakan **GET Request** ke URL API tersebut.
4. Ambil data suhu (main.temp) dan deskripsi cuaca (weather.description) dari JSON yang didapat.
5. Tampilkan di layar HP Anda.

Tugas: Formulir Pendaftaran Online

Gunakan layanan API Dummy seperti **ReqRes.in** (https://reqres.in/).

1. Buat formulir pendaftaran yang berisi Nama dan Pekerjaan (Job).
2. Saat tombol "Daftar" ditekan, kirim data tersebut menggunakan **POST Request** ke endpoint https://reqres.in/api/users.
3. Jika sukses, server ReqRes akan membalas dengan JSON yang berisi id baru dan waktu pembuatan (createdAt).
4. Tampilkan pesan Toast: "Sukses mendaftar dengan ID: [Nomor ID]".

Referensi

1. Android Developers. (2025). *Transmit network data using Volley*. Google Developers. https://developer.android.com/training/volley
2. Bansal, S. (2023). *Android Networking: The Complete Guide to Volley and Retrofit*. New York: TechPress.
3. Haryanto, T. (2022). *Pemrograman Android Modern dengan Java dan Kotlin* (Bab Koneksi Internet). Jakarta: Informatika.
4. Mozilla Developer Network. (2024). *Working with JSON*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON
5. Vogel, L. (2021). *Android development with Volley - Tutorial*. Vogella. https://www.vogella.com/tutorials/AndroidGoogleMaps/article.html

Bab 9
Fragment dan Navigasi

Bayangkan Anda memiliki sebuah bingkai foto yang sangat indah di dinding ruang tamu. Apakah setiap kali Anda bosan dengan fotonya, Anda akan membuang bingkainya ke tempat sampah lalu membeli bingkai baru beserta fotonya?

Tentu tidak. Itu pemborosan yang luar biasa.

Yang Anda lakukan adalah **tetap memasang bingkainya**, lalu cukup **mengganti lembaran foto** di dalamnya.

Di bab-bab sebelumnya, setiap kali kita ingin pindah halaman, kita membuat Activity baru menggunakan perintah startActivity. Ini ibarat membuang bingkai lama dan memasang bingkai baru. Cara ini memakan banyak memori dan perpindahan layar terkadang terasa "berat".

Di Android modern, kita diperkenalkan dengan **Fragment**.

Dalam analogi ini, **Activity** adalah "Bingkai"-nya, dan **Fragment** adalah "Lembaran Foto"-nya. Dengan Fragment, aplikasi kita bisa memiliki satu Activity saja, tetapi isinya (kontennya) bisa berubah-ubah dengan sangat cepat, ringan, dan mulus. Inilah rahasia mengapa aplikasi seperti Instagram atau Gmail terasa sangat responsif saat Anda berpindah tab.

9.1. Apa Itu Fragment?

Secara teknis, **Fragment** adalah bagian modular dari antarmuka pengguna (UI) yang hidup di dalam sebuah Activity. Fragment tidak bisa hidup sendiri; ia bersifat "parasit" yang harus menempel pada inangnya (Activity).

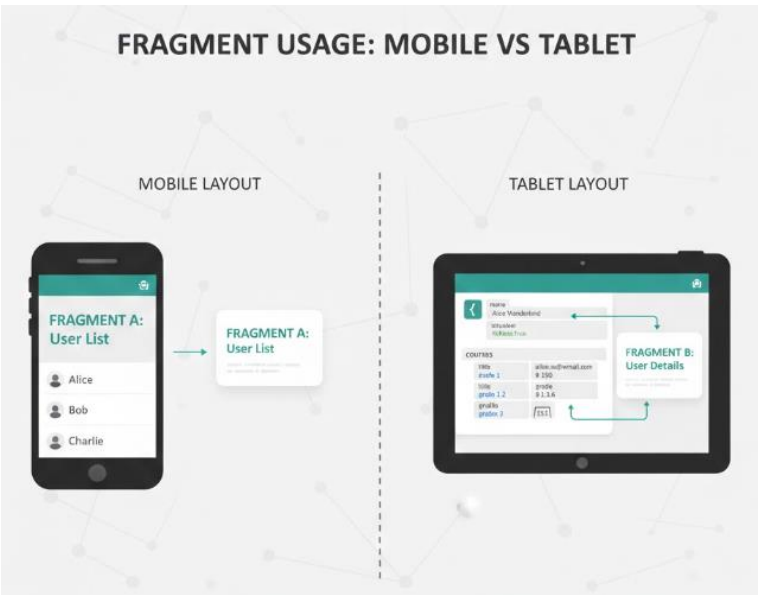
Namun, Fragment punya kelebihan besar: **Reusability** (Bisa dipakai ulang).

Bayangkan Anda membuat sebuah Fragment bernama "Daftar Berita".

- Di **Smartphone** (layar kecil), Fragment ini memenuhi satu layar penuh.

- Di **Tablet** (layar besar), Fragment ini bisa ditaruh di sisi kiri, sementara sisi kanannya menampilkan Fragment lain yaitu "Isi Berita".

Kode Fragment-nya sama persis, tapi cara penempatannya bisa berbeda tergantung wadahnya. Inilah yang membuat Fragment sangat fleksibel untuk desain responsif.



Gambar 9.1. Ilustrasi penggunaan Fragment yang berbeda pada HP dan Tablet

9.2. Siklus Hidup Fragment

Karena Fragment hidup "menumpang" pada Activity, siklus hidupnya sedikit lebih rumit. Ia harus sinkron dengan inangnya.

Jika Activity punya 6 fase utama (seperti yang kita bahas di Bab 4), Fragment punya beberapa fase tambahan yang spesifik:

1. **onAttach():** Momen saat Fragment "ditempelkan" ke Activity. Ibarat tamu yang baru mengetuk pintu dan dipersilakan masuk.
2. **onCreateView():** Momen paling penting. Di sinilah kita "mengembangkan" (*inflate*) layout XML menjadi tampilan nyata.
3. **onViewCreated():** Tampilan sudah jadi. Di sinilah tempat terbaik untuk mengatur logika tombol atau mengisi teks (fungsinya mirip onCreate di Activity).
4. **onDestroyView():** Tampilan Fragment dihancurkan (misal saat diganti Fragment lain), tapi objek Fragment-nya mungkin masih ada di memori.
5. **onDetach():** Fragment benar-benar lepas dari Activity. Tamu pamit pulang.

Poin Kunci: Fragment tidak akan pernah mencapai fase onResume (aktif) jika Activity inangnya belum onResume. Anak tidak bisa bangun pagi jika orang tuanya masih tidur.

9.3. Menambahkan Fragment ke Activity

Ada dua cara untuk memasukkan "foto" ke dalam "bingkai": dipaku mati (Statis) atau dijepit agar bisa diganti-ganti (Dinamis).

A. Cara Statis (Lewat XML)

Cara ini mudah, tapi kaku. Fragment ditanam langsung di layout Activity dan tidak bisa diganti selama aplikasi berjalan.

XML

```xml
<LinearLayout ... >
    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragment_container"
        android:name="com.contoh.app.HomeFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

B. Cara Dinamis (Lewat Java)

Inilah cara yang paling sering digunakan. Kita menggunakan **FragmentManager** untuk melakukan transaksi bongkar-pasang Fragment.

Bayangkan kita punya sebuah wadah kosong (FrameLayout) di layout Activity. Lalu lewat kode Java, kita perintahkan:

XML

```xml
// 1. Panggil Manajer
FragmentManager fragmentManager = getSupportFragmentManager();

// 2. Mulai Transaksi (Seperti transaksi bank)
FragmentTransaction transaction =
fragmentManager.beginTransaction();

// 3. Lakukan Aksi (Replace: Ganti isi wadah dengan
FragmentBaru)
transaction.replace(R.id.wadah_fragment, new HomeFragment());

// 4. Sahkan Transaksi (Commit)
transaction.commit();
```

Dengan 4 baris kode di atas, layar pengguna berubah tanpa harus berpindah Activity.

9.4. Navigasi Antar Fragment

Bagaimana cara berpindah dari Fragment A ke Fragment B?

Meskipun saat ini Google menyarankan penggunaan *Jetpack Navigation Component* (yang menggunakan grafik visual), untuk pemula yang menggunakan Java, memahami **FragmentManager** secara manual adalah fondasi yang wajib dikuasai.

Konsepnya sederhana: **Ganti (Replace)** atau **Tambah (Add)**.

- **Replace:** Membuang Fragment lama, memasukkan Fragment baru. (Hemat memori, tapi Fragment lama mati/reset).
- **Add:** Menumpuk Fragment baru di atas Fragment lama. (Fragment lama masih hidup di bawahnya, tapi memori bertambah).

Back Stack (Tumpukan Kembali)

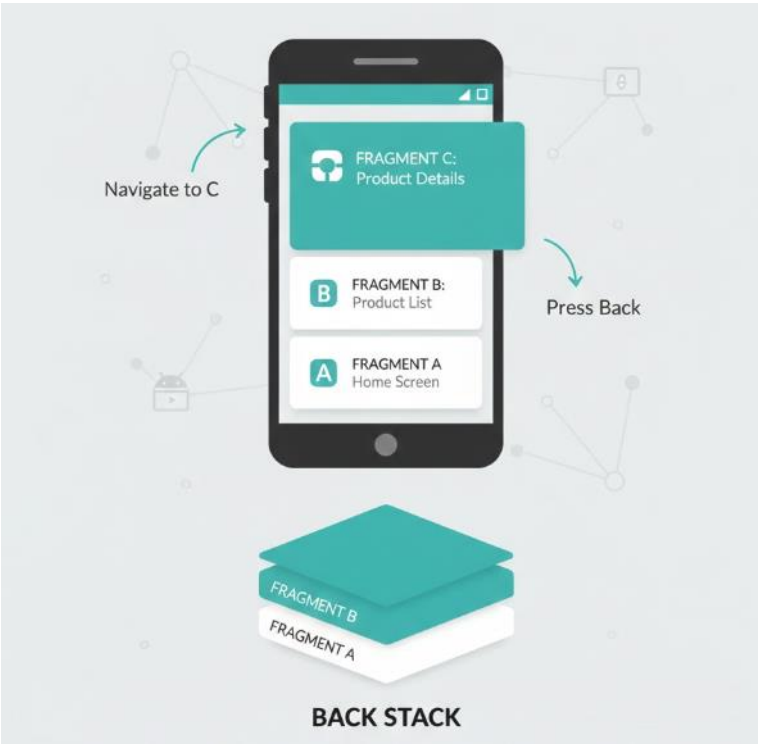
Masalah utama Fragment adalah tombol *Back*. Secara *default*, jika Anda mengganti Fragment lalu menekan tombol *Back* di HP, aplikasi akan langsung keluar (menutup Activity), bukan kembali ke Fragment sebelumnya.

Agar tombol *Back* berfungsi seperti *browser* (kembali ke halaman sebelumnya), kita harus mencatat transaksi tersebut ke dalam "Buku Riwayat" (*Back Stack*).

Java
`transaction.replace(R.id.wadah, new ProfilFragment());`

```
// Mantra ajaib agar tombol Back berfungsi normal
transaction.addToBackStack(null);

transaction.commit();
```

Dengan satu baris addToBackStack(null), Android akan mengingat bahwa sebelum ini pengguna ada di HomeFragment. Jadi saat tombol *Back* ditekan, Android akan membatalkan transaksi terakhir dan mengembalikan HomeFragment ke layar.



Gambar 9.2. Ilustrasi tumpukan Back Stack saat navigasi antar Fragment

Latihan dan Tugas

Bekerja dengan Fragment membutuhkan ketelitian dalam mengelola *Context*. Ingat, di dalam Fragment, kata kunci this merujuk ke

Fragment itu sendiri, bukan Activity. Gunakan getActivity() atau requireContext() jika Anda membutuhkan konteks aplikasi.

Latihan: Tombol Ganti Wajah

1. Buat satu Activity (MainActivity) yang memiliki dua tombol di bagian atas: "Merah" dan "Biru".
2. Siapkan wadah kosong (FrameLayout) di bawah tombol tersebut.
3. Buat dua Fragment sederhana: RedFragment (background merah) dan BlueFragment (background biru).
4. Buat logika: Jika tombol "Merah" ditekan, lakukan replace pada wadah dengan RedFragment. Begitu juga sebaliknya untuk tombol "Biru".

Tugas: Aplikasi Mini Navigasi

Buatlah kerangka aplikasi yang memiliki **Bottom Navigation** (Navigasi Bawah) manual.

1. Buat layout dengan 3 tombol di bawah: **Home**, **Profil**, **Info**.
2. Buat 3 Fragment sesuai nama tombol tersebut. Berikan teks di tengah layar masing-masing Fragment (misal: "Ini Halaman Home").
3. Saat aplikasi pertama dibuka, tampilkan **HomeFragment**.
4. Saat tombol ditekan, ganti tampilan Fragment sesuai tombolnya.
5. **Tantangan:** Pastikan tombol *Back* tidak langsung menutup aplikasi jika pengguna sudah berpindah-pindah menu, melainkan kembali ke menu sebelumnya.

Referensi

1. Android Developers. (2025). *Fragments*. Google Developers. https://developer.android.com/guide/fragments
2. Big Nerd Ranch. (2023). *Android Programming: The Big Nerd Ranch Guide* (Chapter on Fragments). Big Nerd Ranch Guides.
3. Hortasm, M. (2022). *Mahir Coding Android dengan Android Studio: Studi Kasus Fragment*. Bandung: Informatika.
4. Meier, R. (2021). *Professional Android 4 Application Development* (Fragment Management). Wrox.
5. Smyth, N. (2024). *Android Studio Jellyfish Essentials - Java Edition*. Payload Media.

Bab 10
Proyek Akhir Aplikasi Pengelola Catatan Sederhana

Selamat! Jika Anda membaca halaman ini, artinya Anda telah bertahan melewati badai *NullPointerException*, pusingnya mengatur *Layout*, hingga rumitnya logika *Adapter*.

Bab 10 ini bukanlah bab teori baru. Ini adalah "Ujian Akhir" atau "Boss Level" dalam perjalanan belajar Anda.

Ibarat seorang koki yang sudah belajar cara memotong bawang (Bab Layout), cara menyalakan kompor (Bab Activity), dan cara meracik bumbu (Bab Java), sekarang saatnya Anda diminta untuk: **"Masaklah satu hidangan lengkap yang lezat."**

Kita akan membangun sebuah aplikasi **Pengelola Catatan (Notes App)**. Aplikasi ini akan menggabungkan kemampuan:

1. **RecyclerView** (Menampilkan daftar catatan).
2. **SQLite** (Menyimpan catatan agar tidak hilang).
3. **Intent** (Pindah dari daftar ke detail edit).
4. **Activity Lifecycle** (Memastikan data ter-update saat kembali ke halaman utama).

Tarik napas panjang, siapkan kopi, dan mari kita mulai merakit karya *masterpiece* Anda.

10.1. Perencanaan dan Desain Aplikasi

Seorang *programmer* pemula biasanya langsung mengetik kode. Seorang *programmer* profesional mengambil kertas dan pena terlebih dahulu.

Sebelum membuka Android Studio, kita harus mendefinisikan apa yang akan kita buat agar tidak tersesat di tengah jalan. Kita sebut ini fase

Perancangan.

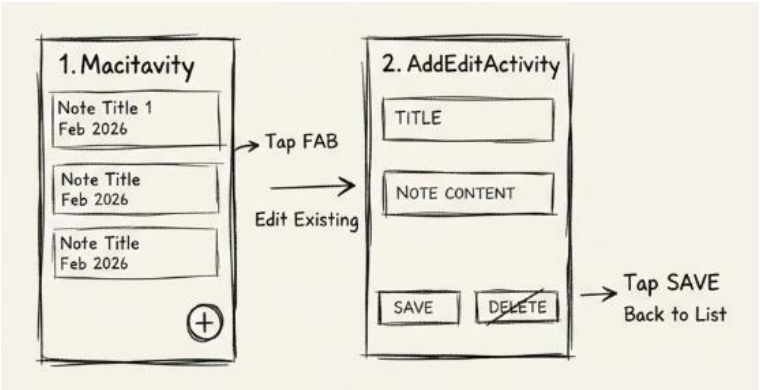
Fitur Utama (MVP - Minimum Viable Product):

1. Menampilkan daftar semua catatan (Judul & Tanggal).
2. Menambah catatan baru.
3. Mengubah (Edit) catatan yang sudah ada.
4. Menghapus catatan.

Sketsa Alur (Wireframe):

Kita butuh dua layar utama:

1. **MainActivity:** Halaman depan berisi RecyclerView dan satu tombol melayang (*Floating Action Button*) untuk menambah data.
2. **AddEditActivity:** Halaman formulir yang berisi EditText (Judul & Isi) dan tombol Simpan.



Gambar 10.1. Sketsa kasar (Wireframe) alur aplikasi Catatan

Struktur Database (SQLite):

Kita butuh satu tabel bernama notes.

- id (Integer, Primary Key, Auto Increment): ID unik untuk setiap catatan.
- title (Text): Judul catatan.
- description (Text): Isi catatan.
- date (Text): Tanggal pembuatan/perubahan.

10.2. Implementasi CRUD

Inilah jantung dari proyek ini. Kita akan menerapkan operasi **CRUD** (*Create*, *Read*, *Update*, *Delete*) secara utuh.

Tahap 1: Fondasi Database (DatabaseHelper)

Ingat Bab 7? Kita buat kelas NoteHelper. Kali ini, pastikan Anda memiliki metode untuk mengambil **satu** data spesifik (untuk fitur Edit) dan mengambil **semua** data (untuk List).

Java

```
// Contoh Query Penting di NoteHelper
public Cursor queryAll() {
    // Mengambil semua data diurutkan dari yang terbaru (id DESC)
    return database.query(DATABASE_TABLE, null, null, null, null, null, "id DESC");
}

public long insert(ContentValues values) {
    return database.insert(DATABASE_TABLE, null, values);
}
// ... lengkapi dengan update() dan delete()
```

Tahap 2: Menampilkan Daftar (Read)

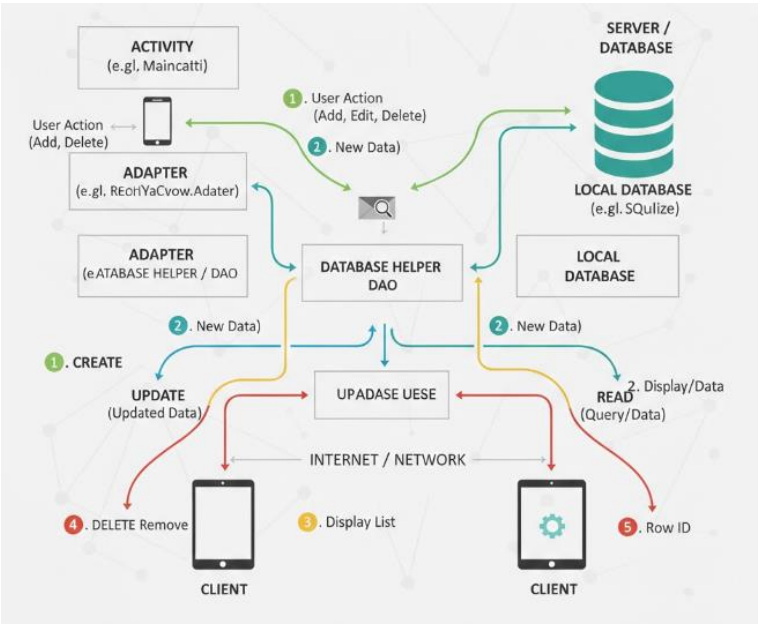
Di MainActivity, kita pasang RecyclerView.

Tantangannya adalah: Data dari SQLite berbentuk Cursor, sedangkan RecyclerView butuh ArrayList.

Solusi: Anda perlu membuat metode *MappingHelper* sederhana untuk mengubah Cursor menjadi ArrayList<Note>. Dan jangan lupa memanggil ulang data saat Activity kembali aktif.

Java

```
// Di MainActivity (onResume)
@Override
protected void onResume() {
    super.onResume();
    // Kenapa di onResume? Agar saat kita kembali dari halaman Edit,
    // daftar catatan langsung diperbarui otomatis.
    loadNotesFromDatabase();
}
// ... lengkapi dengan update() dan delete()
```



Gambar 10.2. Diagram alur data CRUD antara Activity, Adapter, dan Database

Tahap 3: Tambah & Edit (Create & Update)

Bagian ini sering membingungkan pemula: *"Apakah saya harus membuat Activity terpisah untuk Tambah dan Edit?"*

Jawabannya: **Tidak perlu.** Hematlah tenaga Anda. Gunakan satu AddEditActivity saja.

Logikanya:

- Jika Activity dibuka **tanpa** membawa data Intent -> Berarti Mode **Tambah Baru**.
- Jika Activity dibuka **membawa** data Intent (misal ada ID catatan) -> Berarti Mode **Edit**.

Java

```
// Di AddEditActivity
if (getIntent().hasExtra("EXTRA_NOTE")) {
    action = "Edit";
    note = getIntent().getParcelableExtra("EXTRA_NOTE");
    edtTitle.setText(note.getTitle()); // Isi kolom otomatis
} else {
    action = "Tambah";
}

btnSave.setOnClickListener(v -> {
    if (action.equals("Tambah")) {
        noteHelper.insert(values);
    } else {
        noteHelper.update(note.getId(), values);
    }
    finish(); // Tutup Activity
});
```

10.3. Fitur Tambahan: Pencarian & Navigasi

Aplikasi yang baik adalah aplikasi yang memudahkan pengguna. Mari tambahkan fitur pemanis agar aplikasi ini terasa profesional.

Pencarian (Search)

Bayangkan Anda punya 100 catatan. Tidak mungkin pengguna *scroll* satu per satu mencari catatan resep masakan. Kita tambahkan SearchView di Toolbar.

Triknya: Jangan melakukan *query* ke database setiap kali mengetik huruf (itu berat). Cukup saring (filter) ArrayList yang sudah ada di memori Adapter.

1. Tambahkan menu search di res/menu/main_menu.xml.
2. Di Adapter, buat metode filterList(ArrayList<Note>

filteredNotes).

3. Di MainActivity, pasang *listener* pada SearchView.

Navigasi Klik

Saat item di RecyclerView diklik, aplikasi harus membuka halaman Edit dan membawa data catatan tersebut.

Pastikan di NoteAdapter, pada bagian onBindViewHolder, Anda memasang *OnClickListener*.

Java

```
holder.itemView.setOnClickListener(v -> {
    Intent intent = new Intent(context, AddEditActivity.class);
    // Bawa objek catatan ini ke sebelah
    intent.putExtra("EXTRA_NOTE", listNotes.get(position));
    context.startActivity(intent);
});
```

10.4. Peluncuran dan Debugging Akhir

Kode selesai bukan berarti tugas selesai. Sekarang saatnya menjadi **Quality Assurance (QA)**. Uji aplikasi Anda sekejam mungkin.

Uji Coba Brutal (Monkey Test)

Jangan tes aplikasi dengan lembut. Cobalah skenario berikut:

- Klik tombol "Simpan" saat kolom masih kosong (Aplikasi harus menolak dengan pesan error, bukan *crash*).
- Putar layar HP saat mengisi data (Data tidak boleh hilang -> Ingat onSaveInstanceState).
- Hapus semua catatan hingga kosong (Tampilan harus tetap rapi, mungkin muncul teks "Tidak ada data").

Build APK

Jika semua sudah aman, Anda bisa mengubah kode mentah ini menjadi file .apk yang bisa dibagikan ke teman atau dosen Anda.

Caranya: Klik menu **Build > Build Bundle(s) / APK(s) > Build APK**.

Latihan dan Tugas

Ini adalah langkah terakhir sebelum Anda sah menguasai dasar Android.

Latihan: Selesaikan Puzzle

Saya telah memberikan potongan logika di atas. Tugas Anda adalah merangkainya menjadi aplikasi utuh.

- Pastikan fitur Tambah berjalan.
- Pastikan fitur Edit berjalan.
- Pastikan fitur Hapus berjalan (saran: gunakan dialog konfirmasi "Apakah Anda yakin?" agar lebih aman).

Tugas: Reminder dengan AlarmManager

Aplikasi catatan akan lebih berguna jika bisa mengingatkan pengguna.

1. Tambahkan fitur "Ingatkan Saya" (pilih jam dan menit) di halaman AddEditActivity.
2. Gunakan AlarmManager dan BroadcastReceiver (Bab ini belum dibahas detail, anggap ini tantangan riset mandiri!).
3. Saat waktu yang ditentukan tiba, tampilkan **Notifikasi** sederhana yang berisi judul catatan tersebut.

Gambar 10.3. Contoh hasil notifikasi pengingat pada aplikasi Android

Penutup dari Penulis

Selamat! Anda telah menyelesaikan buku ini. Dari baris pertama "Hello World" yang sederhana, kini Anda mampu membuat aplikasi manajemen data yang kompleks.

Ingatlah, teknologi Android berkembang sangat cepat. Apa yang tertulis di buku ini adalah fondasi yang kokoh. Jangan berhenti belajar. Jelajahi *Kotlin*, *Jetpack Compose*, atau *Flutter* di langkah berikutnya. Dunia pengembangan *mobile* ada di genggaman Anda.

Teruslah berkarya, wahai Calon Developer Hebat!

Referensi

1. Android Developers. (2025). *Save data in a local database using Room* (Sebagai referensi lanjutan modern). Google Developers.
2. Haryanto, T. (2024). *Membangun Aplikasi Android Profesional: Studi Kasus Lengkap*. Jakarta: Informatika.
3. Pratama, I. P. A. E. (2023). *Sistem Informasi dan*

Implementasinya (Studi Kasus Sistem Informasi Notes). Bandung: Informatika.

4. Rowe, J. (2024). *Android App Development for the Perplexed*. Apress.
5. Smyth, N. (2025). *Android Studio Koala Essentials*. Payload Media.

Buku Langkah Mudah Membangun Aplikasi Android Profesional dengan Java dirancang sebagai panduan praktis dan sistematis bagi pemula hingga tingkat menengah dalam mengembangkan aplikasi Android. Buku ini mengajak pembaca memahami proses pembuatan aplikasi secara bertahap, mulai dari persiapan lingkungan kerja, pengenalan Android Studio, hingga pengembangan fitur aplikasi yang lebih kompleks. Dengan pendekatan yang aplikatif, buku ini membahas berbagai konsep penting seperti pembuatan antarmuka (layout), pengelolaan siklus hidup activity, penggunaan intent untuk navigasi antar layar, hingga pengelolaan data menggunakan SharedPreferences dan SQLite.

Penerbit buku yang memajukan literasi dan kreativitas dengan menyediakan platform terjangkau bagi penulis berbakat dari berbagai latar belakang

Office Yogyakarta : 087777899993
Marketing 1 : 088221740145
Marketing 2 : 085961447209
Marketing 3 : 0882005806664
Instagram : @ypad_penerbit
Website : https://ypad.store
Email : teampenerbit@ypad.store

ISBN 978-634-295-150-7 (PDF)