



Praktikum Pemrograman Mobile

Pertemuan 2:

View Binding & Navigasi

Disusun oleh :

Azis Amirulbahar, S.P.d., M.T.I.

Program Studi Informatika

Fakultas Teknik

Universitas Jenderal Soedirman

2025



Daftar Isi

<i>Daftar Isi.....</i>	2
<i>View Binding & Navigasi.....</i>	3
A. View Binding.....	3
B. Mengenal Navigasi.....	5
C. Tahapan dalam Praktikum ke-2.....	6
1. Melakukan Konfigurasi Resources.....	6
2. Membuat <i>Reusable Layout</i>	8
3. Membuat Activity Baru.....	14
4. Modifikasi Layout pada Activity Baru (Halaman2Activity.kt).....	16
5. Modifikasi Logic pada Halaman2Activity.....	22
6. Menghubungkan activity di pertemuan-1 dan pertemuan-2.....	29
7. Melakukan Optimize Import.....	31
<i>Daftar Referensi.....</i>	33

View Binding & Navigasi

Pada praktikum pemrograman mobile yang ke-2, kita akan mulai menerapkan *coding* menggunakan bahasa Kotlin secara sederhana yaitu dengan membuat proses perpindahan (navigasi) antar tampilan dari *activity* menggunakan *intent implicit* serta *explicit*. Praktikum ini mencakup materi berikut:

- **Deklarasi variabel menggunakan Kotlin:**
<https://kotlinlang.org/docs/basic-syntax.html#variables>
- **Deklarasi function menggunakan Kotlin:** <https://kotlinlang.org/docs/functions.html>
- **Object Oriented Programming (OOP) pada Kotlin:**
<https://www.idn.id/kotlin-oop-class-constructor-property-dan-method/>
- **Constraint Layout:**
<https://developer.android.com/develop/ui/views/layout/constraint-layout>
- **View Binding:** <https://developer.android.com/topic/libraries/view-binding?hl=id>
- **Intent:** <https://developer.android.com/guide/components/intents-common?hl=id>

A. View Binding

Dikutip dari situs developer.android.com, disebutkan bahwa *ViewBinding* merupakan fitur yang memudahkan penulisan kode pada Kotlin (.kt) untuk melakukan akses terhadap layout yang telah dibuat. Pada pertemuan di teori pemrograman mobile ke-4, telah dijelaskan bahwa proses untuk mengakses *id* pada komponen XML awalnya menggunakan *function findViewById(id)*. Misal jika kita ingin mengakses tombol berikut:

```
<com.google.android.material.button.MaterialButton  
    android:id="@+id	btn_library"  
    android:layout_marginTop="20dp"  
    android:text="@string/implement_library"  
    android:gravity="center"  
    app:backgroundTint="@color/colorPrimary"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

Maka kita perlu mendeklarasikan objek komponen material button tersebut ke dalam file Kotlin kita:

```
val btnLibrary: MaterialButton = findViewById(R.id.btn_library).
```

Pada tahun 2019 melalui event Google I/O, Google memperkenalkan *ViewBinding* yang mempermudah melakukan akses komponen pada *logic* yang akan kita buat. Secara *default*, *ViewBinding* pada proyek android belum diaktifkan, sehingga kita perlu mengkonfigurasikan pada *gradle* level module yang terletak pada sub-folder: app/build.gradle.kts. **Silakan aktifkan *ViewBinding* menggunakan project pertemuan pekan lalu.**

Berikut adalah tahapan yang perlu dilakukan:

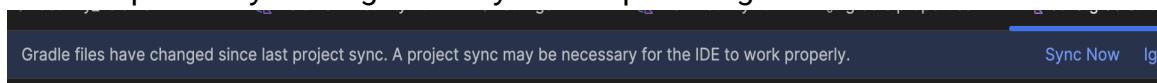
1. Buka *build.gradle.kts* dan tambahkan potongan kode berikut di dalam *scope/blok*

```
android {
```

```
    viewBinding {  
        enable = true  
    }
```

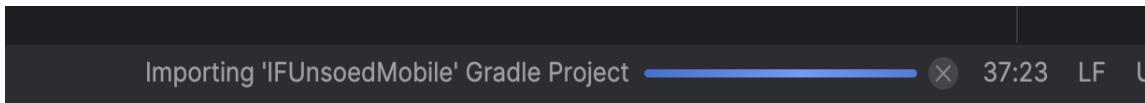
Gambar 1. Konfigurasi View Binding pada Gradle

1. Lakukan proses sync dengan klik Sync Now pada bagian atas:



Gambar 2. Menu Sync pada Gradle

Proses ini dilakukan untuk mengunduh dependensi/library yang telah dikonfigurasi sebelumnya. Proses ini mirip seperti saat praktikum di pemrograman web pada bagian melakukan penambahan *library* menggunakan *composer*. Proses sync tersebut membutuhkan waktu beberapa detik hingga menit bergantung dengan spesifikasi komputer yang digunakan serta kualitas koneksi internet.



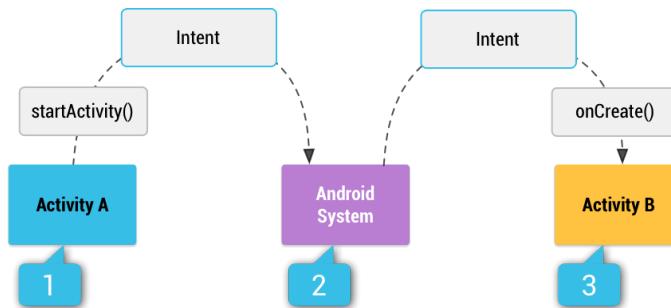
Gambar 3. Proses Sync pada Gradle

B. Mengenal Navigasi

Navigasi merupakan proses perpindahan dari satu *User Interface* (UI) ke UI lain. Navigasi juga merupakan bagian dari pengalaman pengguna/*User Experience* (UX). Selengkapnya tentang navigasi dapat dilihat pada tautan berikut: <https://developer.android.com/guide/navigation/principles>.

Proses perpindahan/navigasi dapat dilakukan dengan beberapa cara:

1. **Dari activity ke activity:** proses perpindahannya dapat dilakukan dengan *intent*. Selengkapnya tentang *intent* dapat dilihat pada tautan berikut :



Gambar 4. Navigasi dari Activity ke Activity

2. **Dari activity ke Fragment** atau sebaliknya dengan menggunakan *intent* atau *fragment transaction*. Selengkapnya tentang *fragment transaction* dapat dilihat pada tautan berikut:

<https://developer.android.com/guide/fragments/transactions>.

3. **Dari fragment ke fragment** dengan menggunakan fragment navigation, yang dapat dipelajari lebih lanjut pada tautan berikut:

<https://developer.android.com/guide/navigation>

Pada praktikum ke-2 ini kita akan mengimplementasikan navigasi antar Activity dengan menggunakan Intent. Intent merupakan objek yang digunakan untuk melakukan perpindahan antar activity dan dapat pula disertakan data untuk dikirim ke activity lain, mengelola services, maupun mengelola *broadcast receiver*.

Selengkapnya tentang intent:

<https://developer.android.com/guide/components/intents-filters?hl=id>.

C. Tahapan dalam Praktikum ke-2

Pada bagian ini, kita akan memulai proses untuk implementasi navigasi. Tahapan yang dilakukan di antaranya: mengaktifkan View Binding yang telah dilakukan sebelumnya, melakukan konfigurasi resources (gambar & teks), membuat reusable layout, membuat & memodifikasi activity baru, serta menghubungkan antara activity pertemuan pertama di pekan lalu dengan hasil praktikum di pertemuan ke-2.

1. Melakukan Konfigurasi Resources

a. Silakan masukkan teks berikut ke dalam string resources (*strings.xml*):

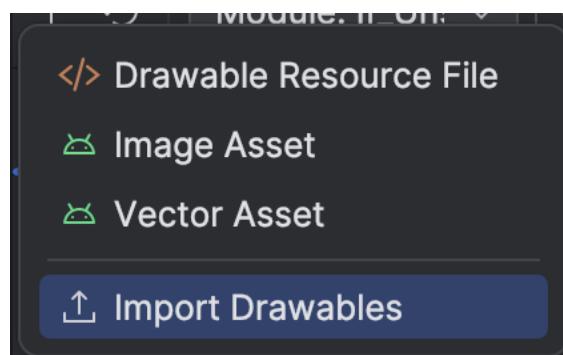
No	Name	Value
1.	alamat	Raya Mayjen Sungkono km 5, Dusun 2, Blater, Kec. Kalimanah, Kabupaten Purbalingga, Jawa Tengah 53371
2.	lokasi	https://maps.app.goo.gl/SX6c8emV88dvuWxV9
3.	telepon	(0281) 6596700
4.	email	teknik@unsoed.ac.id
5.	himpunan	Himpunan Mahasiswa Informatika (HMIF UNSOED)
6.	ig_himpunan	https://www.instagram.com/hmifunsoed
7.	profil_lulusan	Profil Lulusan
8.	desc_lulusan	Lulusan Program Studi Informatika Unsoed sebagian besar bekerja sebagai tenaga ahli di bidang perekayasaan perangkat lunak (Programer, Analis Sistem, Manajer Proyek, Database Administrator), Wirausahawan Informatika dan Dosen/Peneliti.

Tabel 1. Isian pada *strings.xml*

b. Silakan unduh resources untuk praktikum 2 dari tautan berikut:

<https://drive.google.com/drive/folders/1oYgDeWp5uPSHZ3HGI6yvxD0c9BZ2s5UH?usp=sharing>

c. Lakukan **Import Drawables** dari hasil unduhan pada poin a.



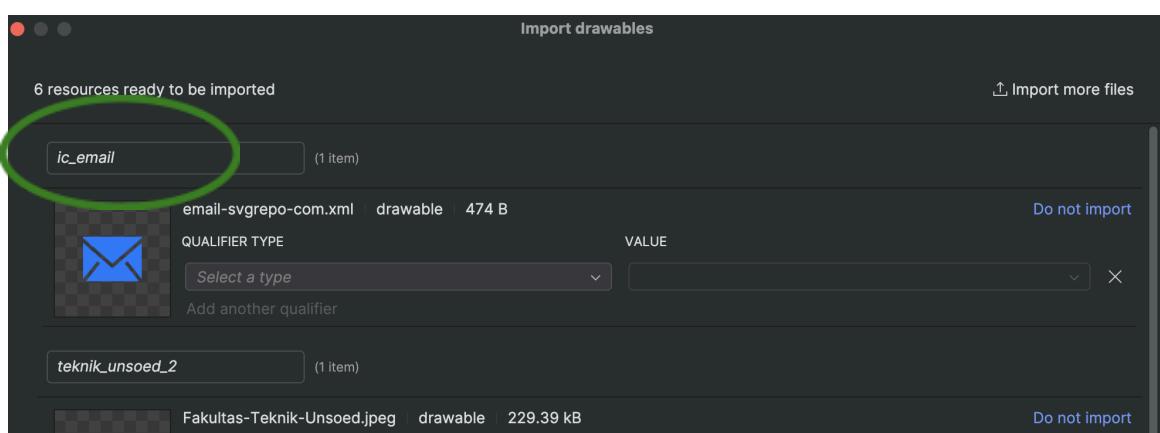
Gambar 5. Menu Import Drawables

Ganti nama – nama dari hasil unduhan dengan ketentuan sebagai berikut:

No	Semula	menjadi
1	email-svgrepo-com	ic_email
2	group-svgrepo-com	ic_himpunan
3	location-svgrepo-com	ic_location
4	phone-calling-svgrepo-com	ic_phone
5	Fakultas-Teknik-Unsoed	teknik_unsoed_2
6	unsoed	lambang_unsoed

Tabel 2. Daftar perubahan nama file gambar

Jika sudah sesuai dengan ketentuan tabel 1 di atas, silakan klik **Next □ Import**.
Tips: untuk melakukan impor sekaligus, silakan klik menu “Import more files” di sebelah kanan. Proses untuk melakukan perubahan nama dapat dilakukan di kolom yang dilingkari warna hijau pada gambar 6.



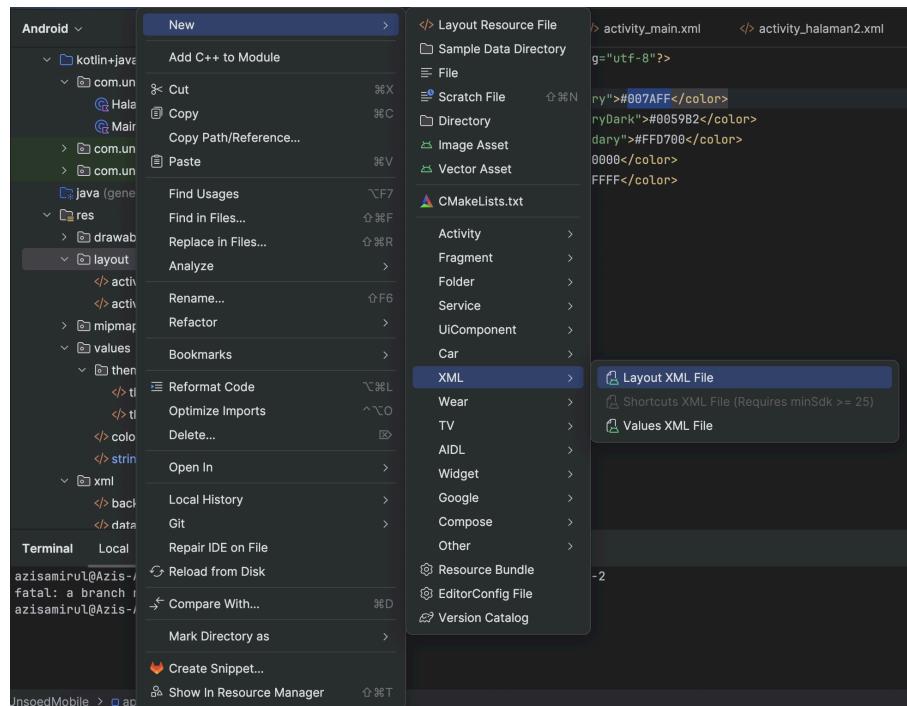
Gambar 6. Tampilan Jendela Import Drawables

2. Membuat Reusable Layout

Dalam membuat suatu layout android, terkadang suatu tampilan memiliki gaya/style yang sama. Sangat membuang – buang waktu jika kita sebagai developer harus mengulang membuat layout yang sama padahal hal tersebut dapat dilakukan sekali dan digunakan berulang kali. Selengkapnya tentang reusable layout dapat dilihat di tautan berikut:
<https://developer.android.com/develop/ui/views/layout/improving-layouts/reusing-layouts>. Berikut adalah tahapan untuk membuat *reusable layout*:

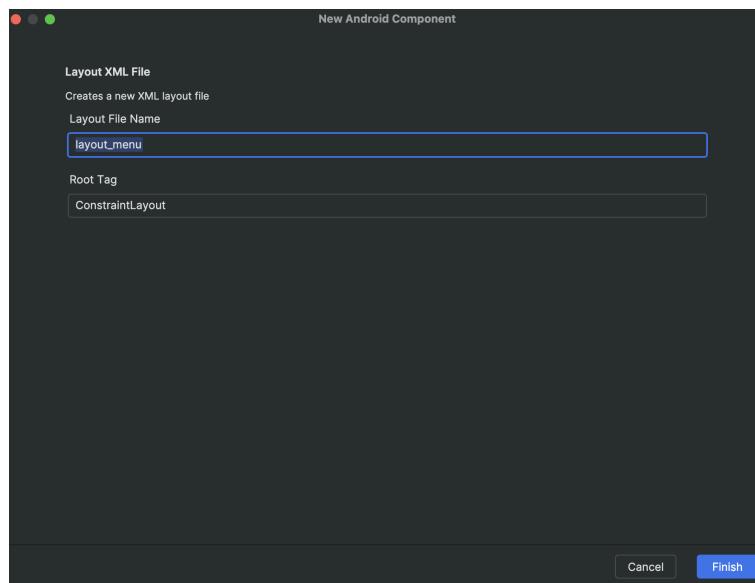
a. Pada folder app/src/main/res/layout silakan klik kanan dan pilih New

XML Layout XML File seperti pada gambar 11 berikut:



Gambar 7. Menu New XML Layout

Beberapa saat kemudian akan muncul tampilan seperti gambar 12 berikut:



Gambar 8. Tampilan Jendela New Android Component

Silakan isi sesuai dengan tabel berikut:

Label	Isian
Layout File Name	layout_menu
Root Tag	ConstraintLayout

Jika telah sesuai, silakan klik "Finish". Hasil dari pembuatan layout tersebut dapat dilihat pada lokasi sebagai berikut:

app/src/main/res/layout/layout_menu.xml & silakan buka file tersebut.

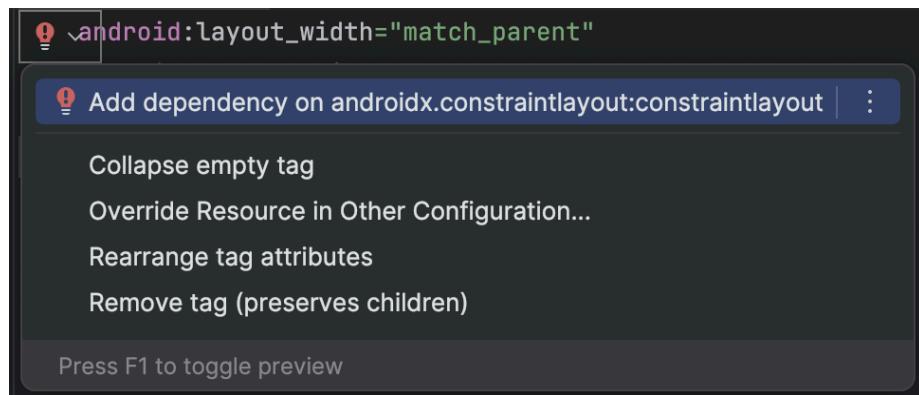
```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</ConstraintLayout>
```

Gambar 9. Tampilan kode layout_menu.xml

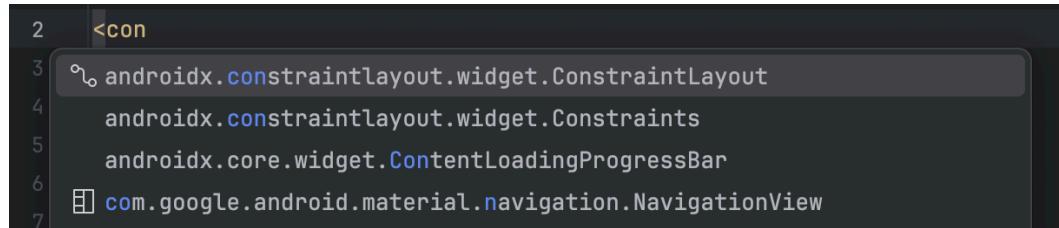
Pada gambar 9 tersebut, terlihat notifikasi error (berwarna merah), silakan arahkan kursor/mouse ke teks berwarna merah dan tunggu sampai muncul ikon

bohlam berwarna merah. Kemudian klik: “**Add dependency on androidx.constraintlayout:constraintlayout**” seperti yang terdapat pada gambar 10 berikut, dan tunggu beberapa saat dikarenakan *android studio* sedang menjalankan “*Gradle Sync in progress*”.



Gambar 10. Menu Add dependency on androidx.constraintlayout:constraintlayout

Masalah error ini terjadi dikarenakan, secara *default* (bawaan) saat melakukan pembuatan proyek android, *dependency/library ConstraintLayout* tidak disediakan secara otomatis, sama seperti fitur *ViewBinding* yang telah dipraktikkan pada poin A. di bagian awal. Setelah proses sync dilakukan, masih terdapat error, dikarenakan penulisan tag xml untuk *ConstraintLayout* tidak sesuai, silakan ketik “con” dan pilih *androidx.constraintlayout.widget.ConstraintLayout* setelah muncul layout suggestion, seperti pada gambar 11 berikut ini:



Gambar 11. Layout Suggestion setelah mengetik “Con”

b. Tambahkan CardView di atas ConstraintLayout.

CardView merupakan komponen untuk menampilkan informasi yang berbentuk seperti kartu yang memiliki sudut di tepi.

```
<androidx.cardview.widget.CardView android:background="#DDEBF7"
    android:layout_width="match_parent"
    app:layout_constraintTop_toTopOf="parent"
    android:padding="10dp"
    android:layout_margin="4dp"
    android:layout_height="wrap_content"
    app:cardCornerRadius="8dp"
    android:foreground="?android:attr/selectableItemBackground"
    android:clickable="true"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
```

Gambar 12. Baris kode komponen CardView pada layout_menu.xml

Catatan: *ConstraintLayout* yang sebelumnya telah dibuat, akan berada di dalam scope *CardView*.

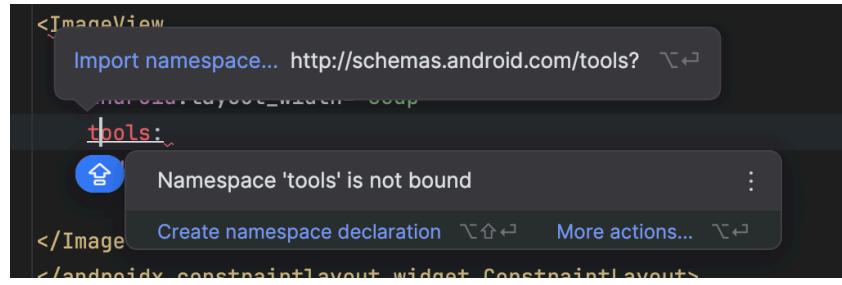
c. Di dalam **ViewGroup ConstraintLayout** tambahkan komponen berikut:

```
<ImageView
    android:id="@+id/img_icon"
    android:layout_width="40dp"
    android:layout_height="40dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:src="@drawable/ic_phone" />
```

Gambar 13. Baris kode komponen Imageview pada layout_menu.xml

Jika terdapat pemberitahuan pada penulisan “tools” silakan arahkan kursor/mouse ke teks tersebut hingga muncul suggestion “Import namespace..”. Penjelasan tentang namespace dapat dilihat pada tautan berikut:

https://docs.oracle.com/cd/E13190_01/liquiddata/docs85/xquery/namespaces.html



Gambar 14. Proses Import Namespace property "tools"

Namespace *tools* pada Android XML merupakan *namespace* khusus untuk *Android Studio/Android build tools*, yang digunakan untuk melakukan pratinjau/preview pada saat melakukan perancangan. Output tersebut **hanya dapat dilihat pada preview** design pada sisi kanan. Sehingga jika aplikasi dijalankan maka tidak akan muncul.

d. Di bawah ImageView (img_icon) tambahkan TextView:

```
<TextView  
    android:id="@+id/tv_layout"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_marginLeft="4dp"  
    android:textColor="@color/black"  
    android:textSize="14sp"  
    android:textStyle="bold"  
    app:layout_constraintBottom_toBottomOf="@id/img_icon"  
    app:layout_constraintLeft_toRightOf="@id/img_icon"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="@id/img_icon"  
    tools:text="@string/telepon" />
```

Gambar 15. Baris kode komponen TextView pada layout_menu.xml

Jika semua kode telah diimplementasikan, maka keseluruhan kode yang tertulis adalah sebagai berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    android:background="#DEEBF7"
    android:clickable="true"
    android:foreground="?android:attr/selectableItemBackground"
    android:padding="10dp"
    app:cardCornerRadius="8dp"
    app:layout_constraintTop_toTopOf="parent">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="100dp">

        <ImageView
            android:id="@+id/img_icon"
            android:layout_width="40dp"
            android:layout_height="40dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:src="@drawable/ic_phone" />

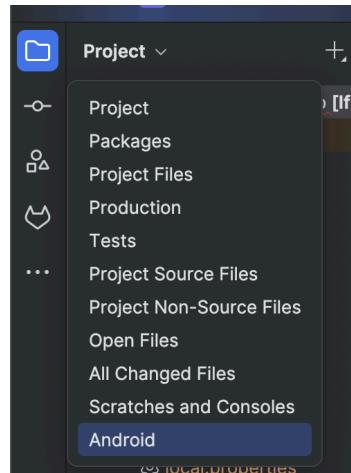
        <TextView
            android:id="@+id/tv_layout"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:textColor="@color/black"
            android:textSize="14sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="@+id/img_icon"
            app:layout_constraintLeft_toRightOf="@+id/img_icon"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="@+id/img_icon"
            tools:text="@string/telepon" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.cardview.widget.CardView>
```

Gambar 16. Tampilan seluruh kode pada layout_menu.xml

3. Membuat Activity Baru

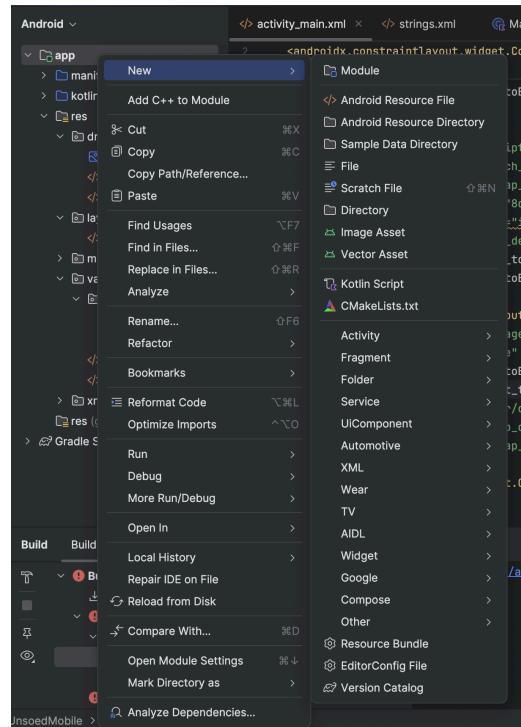
Berikut adalah tahapan dalam membuat activity baru:

1. Jika tampilan dalam struktur proyek android menggunakan mode “**project**” atau yang lain, silakan ubah terlebih dahulu menjadi “**android**”.



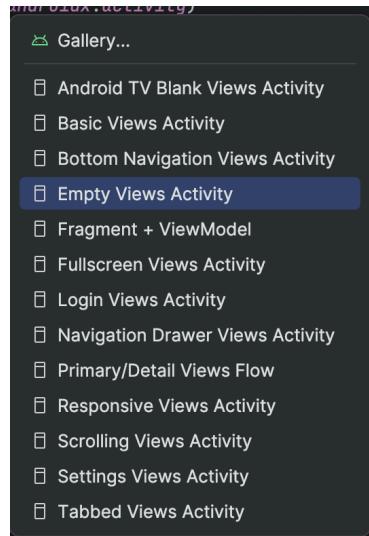
Gambar 17. Model tampilan pada struktur proyek android

a. Klik kanan pada ikon modul app □ New □ Activity



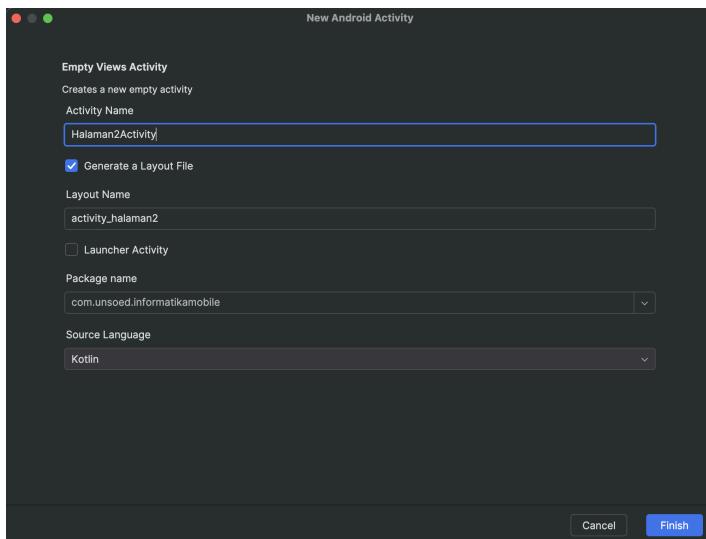
Gambar 18. Tampilan sub-menu “New” pada project android

b. Pilih *Empty View Activity* :



Gambar 19. Pilihan sub-menu New Activity

2. Setelah muncul jendela “New Android Activity”, silakan isi Activity Name dengan nama: “**Halaman2Activity**”.



Gambar 20. Tampilan jendela New Android Activity

Sesuaikan isian seperti pada gambar 8 tersebut, dan klik “Finish”. Android Studio akan melakukan generate class **Halaman2Activity.kt** yang digunakan untuk mengolah logika aplikasi pada sub-folder:
`app/src/main/java/com/unsoed/informatikamobile/Halaman2Activity.kt`,

serta layout pada sub-folder:

app/src/main/res/layout/activity_halaman2.xml

4. Modifikasi Layout pada Activity Baru (Halaman2Activity.kt)

Setelah membuat Activity baru pada tahapan 3 di atas, maka langkah selanjutnya adalah melakukan modifikasi layout dengan menggunakan resources yang telah ditambahkan sebelumnya, dengan hasil akhir seperti pada gambar berikut ini:



Gambar 21. Hasil Akhir ActivityHalaman2

Silakan ikuti langkah – Langkah berikut:

- Sesuaikan RootView pada activity_halaman2.xml menggunakan ConstraintLayout seperti gambar 21 berikut:**

```
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".Halaman2Activity">
```

Gambar 22. RootView dengan ConstraintLayout pada activity_halaman2.xml

- b. **Tambahkan ImageView di dalam ConstraintLayout** yang telah ditulis pada tahapan a di atas, untuk menampilkan gambar teknik_unsoed_2.

```
<ImageView  
    android:id="@+id/iv_gedung_teknik"  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:adjustViewBounds="true"  
    android:scaleType="fitXY"  
    android:src="@drawable/teknik_unsoed_2"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Gambar 23. ImageView dengan id: iv_gedung_teknik

- c. **Tambahkan ImageView untuk menampilkan gambar lambang_unsoed.**

Posisi ImageView “iv_logo_unsoed” akan terletak di atas ImageView “iv_gedung_teknik”.

```
<ImageView  
    android:id="@+id/iv_logo_unsoed"  
    android:layout_width="120dp"  
    android:layout_height="120dp"  
    android:adjustViewBounds="true"  
    android:scaleType="fitXY"  
    android:src="@drawable/lambang_unsoed"  
    app:layout_constraintBottom_toBottomOf="@+id/iv_gedung_teknik"  
    app:layout_constraintLeft_toLeftOf="@+id/iv_gedung_teknik"  
    app:layout_constraintRight_toRightOf="@+id/iv_gedung_teknik"  
    app:layout_constraintTop_toTopOf="parent" />
```

Gambar 24. ImageView dengan id:iv_logo_unsoed

Berikut adalah penjelasan dari implementasi komponen di dalam ViewGroup

ConstraintLayout:

Atribut	Nilai	Penjelasan
android:id="@+id/iv_logo_unsoed"	iv_logo_unsoed	Memberikan ID unik pada ImageView, digunakan untuk mengakses dari kode (binding/Kotlin).
android:layout_width = "120dp"	120dp	Menentukan lebar tampilan sebesar 120 density pixel.
android:layout_height = "120dp"	120dp	Menentukan tinggi tampilan sebesar 120 density pixel.
android:adjustViewBounds="true"	true	Membuat ImageView menyesuaikan ukuran gambar agar proporsinya tetap

Atribut	Nilai	Penjelasan
		terjaga ketika scaleType memungkinkan.
android:scaleType="fitXY"	fitXY	Mengatur gambar agar mengisi penuh ukuran ImageView (bisa menyebabkan gambar terdistorsi).
android:src="@drawable/lambang_unsoed"	@drawable/lambang_unsoed	Menentukan gambar yang ditampilkan dari folder res/drawable.
app:layout_constraintBottom_toBottomOf="@id/iv_gedung_teknik"	@id/iv_gedung_teknik	Mengikat batas bawah ImageView ke batas bawah komponen iv_gedung_teknik.
app:layout_constraintLeft_toLeftOf="@id/iv_gedung_teknik"	@id/iv_gedung_teknik	Mengikat batas kiri ImageView ke batas kiri iv_gedung_teknik.
app:layout_constraintRight_toRightOf="@id/iv_gedung_teknik"	@id/iv_gedung_teknik	Mengikat batas kanan ImageView ke batas kanan iv_gedung_teknik.
app:layout_constraintTop_toTopOf="parent"	parent	Mengikat batas atas ImageView ke batas atas parent (ConstraintLayout).

d. Tambahkan TextView untuk menampilkan teks: "Profil Lulusan"

```

<TextView
    style="@style/TextAppearance.Material3.TitleLarge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:background="@color/colorSecondary"
    android:padding="4dp"
    android:text="Profil Lulusan"
    android:textColor="@color/colorPrimary"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/iv_logo_unsoed" />

```

Gambar 25. TextView untuk menampilkan teks "Profil Lulusan"

- e. Tambahkan ScrollView, yang digunakan untuk **scrolling** pada tampilan di bawah gambar.

```
<ScrollView  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:padding="6dp"  
    app:layout_constraintBottom_toTopOf="@+id/btn_back"  
    app:layout_constraintTop_toBottomOf="@+id/iv_gedung_teknik">
```

Gambar 26. Kode untuk menambahkan ScrollView

- f. Tambahkan ConstraintLayout di dalam ScrollView, seperti pada gambar 27 yang terdapat dalam kotak berwarna putih berikut:

```
<ScrollView  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:padding="6dp"  
    app:layout_constraintBottom_toTopOf="@+id/btn_back"  
    app:layout_constraintTop_toBottomOf="@+id/iv_gedung_teknik">  
  
    <androidx.constraintlayout.widget.ConstraintLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content">
```

Gambar 27. Penambahan ConstraintLayout di dalam ScrollView

Pada tahapan g (menambahkan CardView) sampai dengan l (menambahkan tag *include* dengan id: *layout_ig*) dilakukan *di dalam scope ConstraintLayout yang terdapat di dalam ScrollView*.

g. Tambahkan CardView yang berisi TextView

seperti pada gambar 28 berikut (TextView ini digunakan untuk menampilkan deskripsi lulusan):

```
<com.google.android.material.card.MaterialCardView  
    android:id="@+id/card_profil"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="6dp"  
    app:cardBackgroundColor="@color/material_dynamic_primary90"  
    app:cardCornerRadius="8dp"  
    app:layout_constraintTop_toTopOf="parent">  
  
    <TextView  
        android:id="@+id/tv_lulusan"  
        style="@style/TextAppearance.AppCompat.Title"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:justificationMode="inter_word"  
        android:padding="20dp"  
        android:text="Lulusan Program Studi Informatika Unsoed sebagian besar..."  
        android:textSize="14sp"  
        android:textStyle="italic" />  
    </com.google.android.material.card.MaterialCardView>
```

Gambar 28. Penambahan CardView dan TextView di dalamnya

h. Tambahkan LinearLayout dengan orientasi vertical di bawah CardView:

```
<LinearLayout  
    android:id="@+id/linear_layout_profil"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="8dp"  
    android:orientation="vertical"  
    app:layout_constraintTop_toBottomOf="@+id/card_profil">
```

Gambar 29. Penambahan LinearLayout di bawah CardView

- i. **Di dalam LinearLayout, tambahkan tag include**, include digunakan untuk memanggil layout lain. Layout lain yang dimaksud dalam tahapan ini adalah *layout_menu.xml* yang sebelumnya telah dibuat. Include layout yang pertama memiliki id: *layout_phone*.

```
<include  
    android:id="@+id/layout_phone"  
    layout="@layout/layout_menu"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="4dp" />
```

Gambar 30. Potongan kode include dengan id *layout_phone*

- j. Tambahkan tag **include** dengan id: *layout_email* yang terletak di bawah *layout_phone*.

```
<include  
    android:id="@+id/layout_email"  
    layout="@layout/layout_menu"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="4dp" />
```

Gambar 31.Potongan kode include dengan id *layout_email*

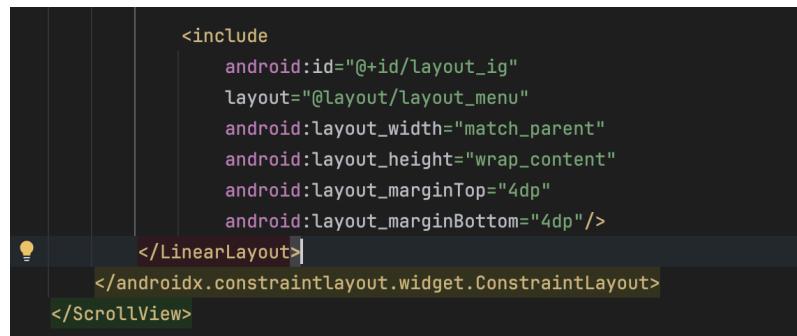
- k. Tambahkan tag **include** dengan id: *layout_location* di bawah *layout_email*.

```
<include  
    android:id="@+id/layout_location"  
    layout="@layout/layout_menu"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="4dp" />
```

Gambar 32.Potongan kode include dengan id *layout_location*

- l. Tambahkan kembali tag **include** dengan id: *layout_ig* di bawah *layout_location*. Layout_ig merupakan bagian terakhir sebelum closing tag </LinearLayout>,

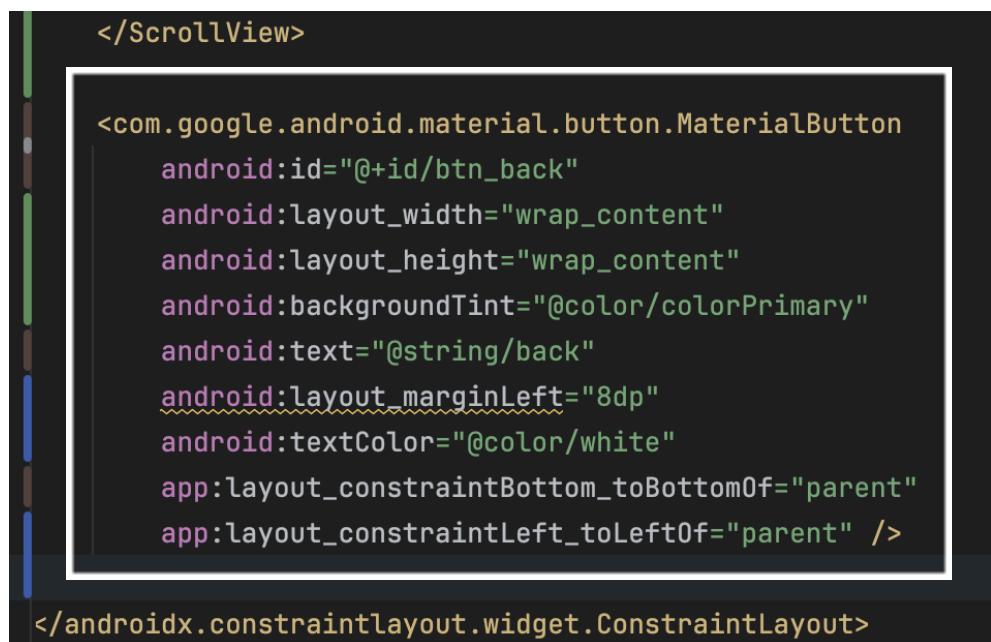
```
</android.constraintlayout.widget.ConstraintLayout>, dan  
</ScrollView>.
```



```
        <include  
            android:id="@+id/layout_ig"  
            layout="@layout/layout_menu"  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:layout_marginTop="4dp"  
            android:layout_marginBottom="4dp"/>  
    </LinearLayout>  
</androidx.constraintlayout.widget.ConstraintLayout>  
</ScrollView>
```

Gambar 33. Potongan kode include dengan id layout_menu

- m. Tambahkan komponen **MaterialButton** di bawah closing tag ScrollView.



```
</ScrollView>  
  
<com.google.android.material.button.MaterialButton  
    android:id="@+id/btn_back"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:backgroundTint="@color/colorPrimary"  
    android:text="@string/back"  
    android:layout_marginLeft="8dp"  
    android:textColor="@color/white"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>
```

Gambar 34. Potongan kode MaterialButton

Pastikan kembali layout telah sesuai dengan spesifikasi akhir seperti pada [gambar 21](#).

5. Modifikasi Logic pada Halaman2Activity

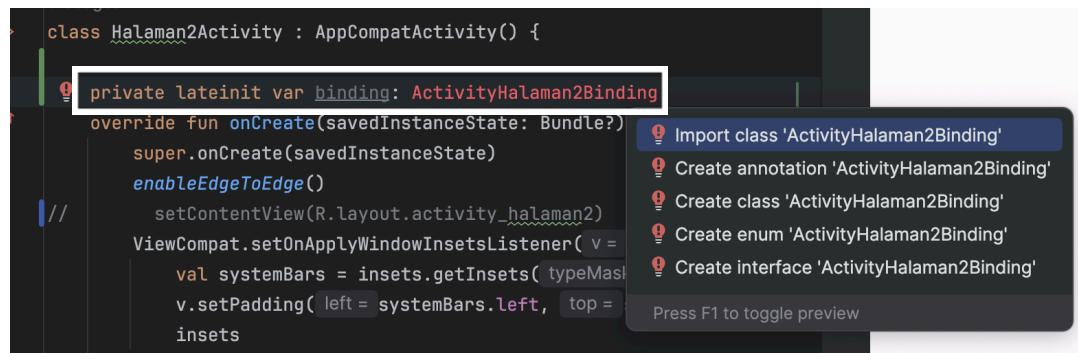
Pada tahapan modifikasi logic ini, kita akan mengimplementasikan *ViewBinding* serta Intent menggunakan Kotlin. Silakan buka file Halaman2Activity.kt.

- a. Menambahkan variabel *binding*

Silakan tulis kode yang terdapat pada persegi berwarna putih pada gambar 35 berikut, dan jika terjadi error yang terdapat pada teks *ActivityHalaman2Binding*, tunggu beberapa saat hingga *suggestion muncul*. Hal ini dikarenakan *ActivityHalaman2Binding* terdapat pada package :

com.unsoed.informatikamobile.databinding.

yang otomatis ter-generate ketika kita mengaktifkan *ViewBinding*.



```
class Halaman2Activity : AppCompatActivity() {  
    private lateinit var binding: ActivityHalaman2Binding  
    override fun onCreate(savedInstanceState: Bundle?)  
        super.onCreate(savedInstanceState)  
        enableEdgeToEdge()  
        setContentView(R.layout.activity_halaman2)  
        ViewCompat.setOnApplyWindowInsetsListener(v =  
            val systemBars = insets.getInsets(  
                v.setPadding(left = systemBars.left, top =  
                    insets  
            )  
        )  
    }  
}
```

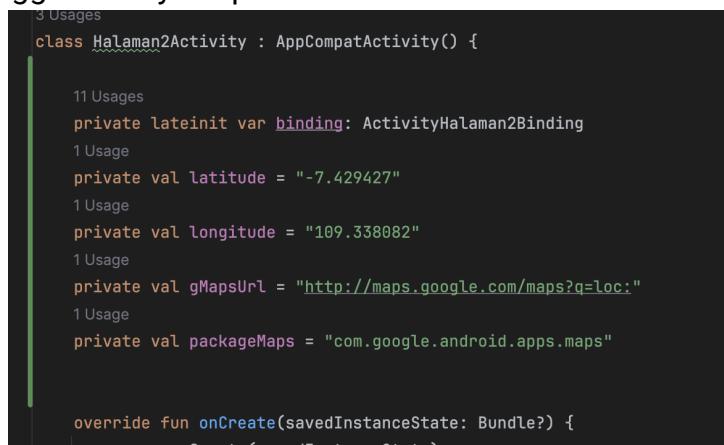
Gambar 35.Deklarasi variabel binding

b. Menambahkan variabel pada Halaman2Activity

Di bawah variabel binding, tambahkan variabel yang bersifat *immutable* (val) dengan nama variabel sesuai dengan teks berwarna ungu dan nilainya sesuai dengan teks berwarna hijau sebagai berikut:

```
Latitude: "-7.429427"  
Longitude: "109.338082"  
gMapsUrl: "http://maps.google.com/maps?q=loc:"  
packageMaps: "com.google.android.apps.maps"
```

sehingga hasilnya seperti di bawah ini:



```
3 Usages  
class Halaman2Activity : AppCompatActivity() {  
  
    11 Usages  
    private lateinit var binding: ActivityHalaman2Binding  
    1 Usage  
    private val latitude = "-7.429427"  
    1 Usage  
    private val longitude = "109.338082"  
    1 Usage  
    private val gMapsUrl = "http://maps.google.com/maps?q=loc:"  
    1 Usage  
    private val packageMaps = "com.google.android.apps.maps"  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
    }  
}
```

Gambar 36. Deklarasi variabel pada kelas activity: Halaman2Activity.kt

c. Konfigurasi ViewBinding pada method `onCreate`

Oncreate merupakan lifecycle method milik Activity yang Dipanggil pertama kali saat Activity dibuat. Silakan hapus kode:

```
enableEdgeToEdge()
```

serta:

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {  
    v, insets ->  
        val systemBars =  
            insets.getInsets(WindowInsetsCompat.Type.systemBars())  
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,  
            systemBars.bottom)  
        insets  
}
```

Kemudian tambahkan potongan kode berikut:

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
  
    binding = ActivityHalaman2Binding.inflate(layoutInflater)
```

Kemudian ganti value di dalam pemanggilan `setContentView` sebagai berikut:

```
setContentView(binding.root)
```

Sehingga hasil akhirnya adalah sebagai berikut:

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
  
    binding = ActivityHalaman2Binding.inflate(layoutInflater)  
    setContentView(binding.root)  
}
```

Penjelasan:

- *ActivityHalaman2Binding* adalah kelas otomatis yang dibuat berdasarkan layout *activity_halaman2.xml*.
- *Inflate(layoutInflater)* digunakan untuk membaca layout XML dan mengubahnya menjadi objek View.
- *setContentView*: Menentukan isi tampilan utama Activity yaitu menggunakan **root view** dari layout yang sudah di-inflate melalui binding. *binding.root* adalah **root layout** dari *activity_halaman2.xml* (*ConstraintLayout*).

d. Menambahkan function *initLayout*

Pada tahapan ini, kita akan menambahkan function *initLayout*, yang digunakan untuk melakukan proses pemberian nilai pada *ImageView* serta *TextView* yang sebelumnya telah dibuat pada *layout_menu.xml*. Pada tahapan sebelumnya kita pernah melakukan pemberian nilai pada xml misalnya: `android:text=""` atau `android:src=""` yang disebut dengan deklaratif. Selain melakukan pemberian nilai/value yang dilakukan langsung pada xml, kita juga dapat melakukannya di file Kotlin (.kt) yang disebut dengan istilah “programmatically”. Silakan buat *function* dengan nama `initLayout` di bawah function *onCreate* dengan potongan kode sebagai berikut:

```

private fun initLayout() {
    binding.layoutLocation.let {
        it.imgIcon.setImageResource(R.drawable.ic_location)
        it.tvLayout.setText(R.string.alamat)
    }

    binding.layoutEmail.let {
        it.imgIcon.setImageResource(R.drawable.ic_email)
        it.tvLayout.setText(R.string.email)
    }

    binding.layoutIg.let {
        it.imgIcon.setImageResource(R.drawable.ic_himpunan)
        it.tvLayout.setText(R.string.ig_himpunan)
    }

    binding.layoutPhone.let {
        it.imgIcon.setImageResource(R.drawable.ic_phone)
        it.tvLayout.setText(R.string.telepon)
    }
}

```

`let` adalah **scope function**, yang menerapkan lambda expression. Selengkapnya tentang lambda expression:

<https://www.petanikode.com/java-lambda/>.

Artinya, ia digunakan untuk mengeksekusi sebuah blok kode dalam lingkup (scope) dari sebuah objek. Fungsi utama `let`:

1. Membuat scope khusus (block)
2. *Null-safety* (menghindari *NullPointerException*)

Selengkapnya tentang scope function dapat dilihat di tautan berikut:

<https://dev.to/alfianandinugraha/let-run-apply-also-di-kotlin-2inb>

Berikut adalah penjelasan dari function `initLayout`:

Bagian Kode	Komponen	Fungsi yang Dipanggil	Penjelasan
<code>binding.layoutLocation.let { ... }</code>	layoutLocation (custom layout di XML)	<code>it.imgIcon.setImageResource(R.drawable.ic_location)</code> <code>it.tvLayout.setText(R.string.alamat)</code>	- Mengatur ikon menjadi gambar lokasi (ic_location). - Mengatur teks menjadi string alamat dari strings.xml.
<code>binding.layoutEmail.let { ... }</code>	layoutEmail	<code>it.imgIcon.setImageResource(R.drawable.ic_email)</code> <code>it.tvLayout.setText(R.string.email)</code>	- Ikon email (ic_email). - Teks diisi dengan email dari strings.xml.

binding.layoutIg .let { ... }	layoutIg	it.imgIcon.setImageResource(R.drawable.ic_himpunan) it.tvLayout.setText(R.string.ig_himpunan)	- Ikon himpunan/Instagram (ic_himpunan) .- Teks diisi dengan URL Instagram dari strings.xml.
binding.layoutPhone .let { ... }	layoutPhone	it.imgIcon.setImageResource(R.drawable.ic_phone) it.tvLayout.setText(R.string.telepon)	- Ikon telepon (ic_phone).- Teks diisi dengan nomor telepon dari strings.xml.

Tabel 3. Penjelasan Function InitView

e. Menambahkan function initListener

Di bawah function initView() silakan tambahkan function initListener:

```
private fun initListener() {
    binding.layoutLocation.root.setOnClickListener {
        val gMapsIntentUri = "$gMapsUrl$latitude,$longitude".toUri()
        val mapIntent = Intent( action = Intent.ACTION_VIEW, gMapsIntentUri)
        startActivity( intent = mapIntent.setPackage(packageMaps))
    }

    binding.layoutIg.root.setOnClickListener {
        val intent = Intent( action = Intent.ACTION_VIEW)
        intent.data = getString( resId = R.string.ig_himpunan).toUri()
        startActivity(intent)
    }

    binding.layoutEmail.root.setOnClickListener {
        val intent = Intent( action = Intent.ACTION_SENDTO).apply {
            data = "mailto:${getString( resId = R.string.email)}".toUri()
        }
        startActivity(intent)
    }

    binding.layoutPhone.root.setOnClickListener {
        val intent = Intent( action = Intent.ACTION_DIAL).apply {
            data = "tel:${getString( resId = R.string.telepon)}".toUri()
        }
        startActivity(intent)
    }

    binding.btnExit.setOnClickListener {
        finish()
    }
}
```

Gambar 37. Potongan kode function InitListener

Pada implementasi baris kode ini, kita telah mempraktikan tentang intent implisit, yaitu intent yang berinteraksi dengan lingkungan luar aplikasi, di antaranya : membuka peta, membuka link, mengirim email, membuka fitur telepon.

Selain itu kita juga telah mempraktikan salah satu fitur dari Kotlin yaitu untuk menghubungkan variabel dengan tanda \$ yaitu pada saat melakukan penulisan kode: "\$gMapsUrl\$latitude,\$longitude" .Pola penulisan tersebut dinamakan **string template**. Silakan buka tautan berikut untuk mendalami fitur lain yang berkaitan dengan String di bahasa Kotlin:

<https://kotlinlang.org/docs/strings.html>. Jika dijabarkan adalah sebagai berikut:

- gMapsUrl, latitude, dan longitude adalah variabel.
- Dengan tanda \$, Kotlin akan mengganti bagian itu dengan **nilai variabel**.
- Hasil akhirnya berupa **string lengkap** yang menyatukan URL Google Maps dengan *latitude&longitude*.

Berikut adalah penjelasan baris kode dari function initListener:

Baris Kode	Penjelasan
<code>private fun initListener() {</code>	Mendefinisikan fungsi privat bernama initListener, berisi semua inisialisasi listener (aksi ketika user menekan elemen UI yang telah kita buat).
<code>binding.layoutLocation.root.setOnClickListener { ... }</code>	Memberikan listener klik pada komponen layout lokasi.
<code>val gMapsIntentUri = "\$gMapsUrl\$latitude,\$longitude".toUri()</code>	Membuat URI lokasi dari variabel gMapsUrl, latitude, dan longitude, lalu mengubahnya menjadi Uri.
<code>val mapIntent = Intent(Intent.ACTION_VIEW, gMapsIntentUri)</code>	Membuat Intent dengan action ACTION_VIEW untuk membuka lokasi menggunakan Google Maps.
<code>startActivity(mapIntent.setPackage(packageMaps))</code>	Menjalankan intent dengan aplikasi Google Maps (setPackage memastikan hanya Google Maps yang dipakai).
<code>binding.layoutIg.root.setOnClickListener { ... }</code>	Listener klik untuk layout Instagram.
<code>val intent = Intent(Intent.ACTION_VIEW)</code>	Membuat intent untuk membuka link/URL.
<code>intent.data = getString(R.string.ig_himpunan).toUri()</code>	Mengisi data intent dengan URL Instagram dari resource string.
<code>startActivity(intent)</code>	Menjalankan intent untuk membuka Instagram di browser/aplikasi Instagram.
<code>binding.layoutEmail.root.setOnClickListener { ... }</code>	Listener klik untuk layout email.
<code>val intent = Intent(Intent.ACTION_SENDTO).apply { ... }</code>	Membuat intent untuk mengirim email dengan action SENDTO.
<code>data = "mailto:\${getString(R.string.email)}".toUri()</code>	Menentukan penerima email menggunakan format mailto: dari resource string.

Baris Kode	Penjelasan
<code>startActivity(intent)</code>	Menjalankan intent untuk membuka aplikasi email.
<code>binding.layoutPhone.root.setOnClickListener { ... }</code>	Listener klik untuk layout telepon.
<code>val intent = Intent(Intent.ACTION_DIAL).apply { ... }</code>	Membuat intent untuk membuka dialer (bukan langsung panggil).
<code>data = "tel:\${getString(R.string.telepon)}".toUri()</code>	Mengisi nomor telepon dengan format tel: dari resource string.
<code>startActivity(intent)</code>	Menjalankan intent untuk membuka aplikasi telepon dengan nomor terisi.
<code>binding.btnExit.setOnClickListener { finish() }</code>	Listener klik tombol back, menjalankan finish() untuk menutup activity saat ini.
<code>}</code>	Penutup fungsi initListener.

Tabel 4. Penjelasan Baris Kode pada Function InitListener

f. Panggil function initView dan InitListener di method onCreate

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityHalaman2Binding.inflate(layoutInflater)
    setContentView(binding.root)

    initLayout()
    initListener()
}
```

Gambar 38. Potongan kode untuk memanggil function initLayout() dan initListener() pada method onCreate

Berikut adalah penjelasan dari potongan kode pada gambar 39 di atas:

Baris Kode	Penjelasan
<code>override fun onCreate(savedInstanceState: Bundle?) {</code>	Mendefinisikan fungsi onCreate yang dioverride dari superclass AppCompatActivity. Fungsi ini dipanggil pertama kali ketika Activity dibuat. Parameter savedInstanceState menyimpan state activity sebelumnya (jika ada).
<code>super.onCreate(savedInstanceState)</code>	Memanggil implementasi onCreate dari superclass (AppCompatActivity) agar inisialisasi standar Android tetap dijalankan.
<code>binding = ActivityHalaman2Binding.inflate(layoutInflator)</code>	Membuat objek View Binding untuk activity_halaman2.xml. inflate(layoutInflater)

	akan menghubungkan file XML layout dengan objek binding.
<code>setContentView(binding.root)</code>	Menetapkan tampilan utama activity menggunakan root view dari objek binding. Artinya UI yang ditampilkan ke layar berasal dari layout <code>activity_halaman2.xml</code> .
<code>initLayout()</code>	Memanggil fungsi <code>initLayout()</code> → yang telah kita buat sebelumnya
<code>initListener()</code>	Memanggil fungsi <code>initListener()</code> yang telah kita buat sebelumnya
<code>}</code>	Penutup fungsi <code>onCreate</code> .

Tabel 5. Penjelasan potongan kode pada blok `onCreate` yang terdapat pemanggilan fungsi `initLayout` dan `initListener`

Pada method tersebut juga sebenarnya kita telah mengimplementasikan konsep OOP pada Kotlin, berikut adalah penjelasan singkat:

KONSEP OOP	PENERAPAN DALAM KODE	Penjelasan
Class	<code>Halaman2Activity : AppCompatActivity()</code>	Activity adalah sebuah class turunan dari <code>AppCompatActivity</code> . Setiap Activity di Android dibuat sebagai class.
Inheritance (Pewarisan)	<code>: AppCompatActivity()</code>	Halaman2Activity mewarisi semua fungsi dan properti dari <code>AppCompatActivity</code> , termasuk <code>onCreate()</code> .
Polymorphism (Polimorfisme)	<code>override fun onCreate(savedInstanceState: Bundle?)</code>	Fungsi <code>onCreate</code> yang sama dari superclass dioverride agar memiliki perilaku berbeda di subclass <code>Halaman2Activity</code> .

Tabel 6. Penjelasan Singkat Konsep OOP pada Activity

6. Menghubungkan **activity** di pertemuan-1 dan pertemuan-2

Proses selanjutnya adalah menghubungkan antara activity pada pertemuan 1 lalu ke activitiy pada pertemuan ke-2 ini, sehingga akan dihasilkan proses navigasi antar activity. Berikut adalah tahapan yang dilakukan:

a . Buka file

`app/src/main/java/com/unsoed/informatikamobile/MainActivity.kt`

b. deklarasi ViewBinding pada MainActivity.kt

Proses ini seperti yang telah dilakukan sebelumnya pada Halaman2Activity.



```
1 Usage
class MainActivity : AppCompatActivity() {
    3 Usages
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

    }
}
```

Gambar 39. Potongan kode inisialisasi ViewBinding pada method onCreate

c. Buatlah function initNavigation

Function *initNavigation* digunakan untuk melakukan navigasi dari ActivityMain ke Halaman2Activity menggunakan *intent explicit*.



```
1 Usage
private fun initNavigation(){
    binding.btnToPage2.setOnClickListener{
        startActivity(Intent(packageContext = this, cls = Halaman2Activity::class.java))
    }
}
```

Gambar 40. Potongan baris kode initNavigation

Baris Kode	Penjelasan
<code>binding.btnToPage2.setOnClickListener{ ... }</code>	Memberikan listener klik pada tombol btnToPage2 (yang sudah dihubungkan lewat View Binding). Aksi di dalam blok (...) akan dijalankan saat tombol ditekan.
<code>startActivity(...)</code>	Memanggil method untuk memulai Activity baru .
<code>Intent(this, Halaman2Activity::class.java)</code>	Membuat objek Intent yang berisi informasi: - this → konteks saat ini (Activity yang sedang berjalan). - Halaman2Activity::class.java → Activity tujuan yang akan dibuka (Halaman2Activity).

Tabel 7. Penjelasan baris kode pada function initNavigation

Berikut adalah tambahan penjelasan potongan baris mengenai implementasi *intent explicit* pada function initNavigation:

Bagian	Penjelasan
<code>Intent(this, Halaman2Activity::class.java)</code>	Ini adalah Explicit Intent . Tujuannya jelas, yaitu membuka activity Halaman2Activity.
<code>this</code>	Menunjukkan konteks dari activity saat ini (misalnya MainActivity).
<code>Halaman2Activity::class.java</code>	Kelas activity tujuan yang sudah terdaftar di AndroidManifest.xml.
<code>startActivity(...)</code>	Menjalankan intent tersebut sehingga Android menampilkan Halaman2Activity.

Tabel 8. Penjelasan Implementasi Intent Explicit pada function InitNavigation

7. Melakukan Optimize Import

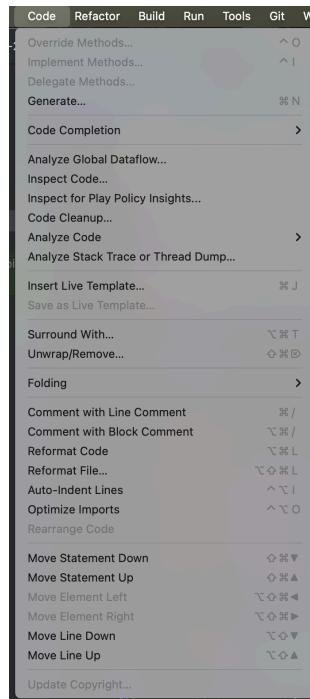
Jika terdapat baris – baris kode yang berwarna abu – abu seperti gambar 41, silakan klik “Optimize Import”. “Optimize import” digunakan untuk melakukan proses optimasi import class dari package yang tidak terpakai. Package yang tidak terpakai tersebut sering terjadi ketika kita telah menuliskan kode dan juga mengimport package tertentu, namun pada akhirnya kode tersebut dihapus karena penyesuaian logic yang telah kita buat. Menghapus kode tidak otomatis menghapus import package. IDE android studio akan mengimpor package sesuai kelas yang digunakan.

Melakukan proses *optimize import* sangat berguna untuk menjaga kerapihan kode, memudahkan pembacaan, dan mencegah kebingungan saat pengembangan. Istilah kode import yang tidak terpakai (yang berwarna abu – abu) disebut dengan “*unused import*”. Gambar 41 adalah contoh dari “*unused import*”.

```
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.unsoed.informatikamobile.databinding.ActivityHalaman2Binding
import androidx.core.net.toUri
```

Gambar 41. Unused Import pada class

Untuk menghilangkan baris – baris kode unused import tersebut dapat dilakukan dengan mengakses menu *Code □ Optimize Import*.



Gambar 42. Menu Optimize Import

Proses penulisan kode maupun konfigurasi telah selesai, silakan coba untuk melakukan build & run menggunakan emulator atau perangkat android.

Daftar Referensi

Android Developers. (n.d.). *Build a responsive UI with ConstraintLayout*. Google.
<https://developer.android.com/develop/ui/views/layout/constraint-layout>

Android Developers. (n.d.). *Intents umum / Components guide*. Google.
<https://developer.android.com/guide/components/intents-common?hl=id>

Android Developers. (n.d.). *View binding / App architecture*. Google.
<https://developer.android.com/topic/libraries/view-binding?hl=id>

ID Networkers. (n.d.). *Kotlin OOP (Class, Constructor, Property dan Method)*. IDN.ID.
<https://www.idn.id/kotlin-oop-class-constructor-property-dan-method/>

Kotlin Programming Language. (n.d.). *Basic syntax*. JetBrains.
<https://kotlinlang.org/docs/basic-syntax.html#variables>

Kotlin Programming Language. (n.d.). *Functions*. JetBrains.
<https://kotlinlang.org/docs/functions.html>

Oracle. (n.d.). *Understanding XML Namespaces*. Oracle.
https://docs.oracle.com/cd/E13190_01/liquiddata/docs85/xquery/namespaces.html