**LAPORAN RESMI**

**MODUL III**

**INTENTS, MENU DAN DIALOG**

**PEMROGRAMAN BERGERAK**



**NAMA : SEVIN DIAS ANDIKA**

**N.R.P : 210441100105**

**DOSEN : Ir. ACH. DAFID, S.T., M.T.**

**ASISTEN : DAVID NASRULLOH**

**TGL PRAKTIKUM : 07 APRIL 2023**

**Disetujui : .. APRIL 2022**

**Asisten**

**DAVID NASRULLOH**

**190441100060**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## **Latar Belakang**

Sebuah aplikasi tidak mungkin hanya melibatkan satu komponen dalam pembuatannya. Untuk itu, perlu untuk mempelajari bagaimana membuat aplikasi yang mengintegrasikan beberapa komponen. Project ini akan digunakan untuk mengintegrasikan kompomenkomponen membentuk aplikasi yang fungsional. Selain itu, digunakan juga intent untuk membuat aplikasi yang berhubungan dengan halaman lain.

Intents adalah sebuah metode android untuk me-relay informasi tertentu dari satu aktivitas ke aktivitas lainnya. Secara lebih sederhana, Intents mengekspresikan kepada android untuk melakukan sesuatu. Navigation Graph (resource XML baru) adalah sumber daya yang berisi semua informasi terkait navigasi di satu lokasi terpusat, termasuk semua tempat di aplikasi, yang dikenal sebagai tujuan, dan kemungkinan jalur yang dapat dilalui pengguna melalui aplikasi. NavHostFragment (Layout XML view) adalah widget khusus yang ditambahkan ke layout, yang menampilkan berbagai tujuan dari Navigation Graph. NavController (objek Kotlin / Java) adalah objek yang melacak posisi saat ini dalam Navigation Graph, yang mengatur pertukaran konten tujuan di NavHostFragment saat kita bergerak melalui Navigation Graph.

## **Tujuan**

* Membuat sebuah Projek Aplikasi Android sederhana yang melibatkan komponenkomponen yang sudah dipelajari.
* Menggunakan menu dan Dialog.

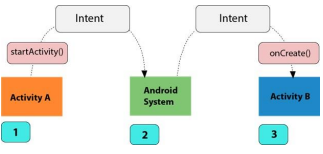
# **BAB II DASAR TEORI**

## **Project Terintegrasi**

Sebuah aplikasi tidak mungkin hanya melibatkan satu komponen dalam pembuatannya. Untuk itu, perlu untuk mempelajari bagaimana membuat aplikasi yang mengintegrasikan beberapa komponen. Project ini akan digunakan untuk mengintegrasikan kompomenkomponen membentuk aplikasi yang fungsional. Selain itu, digunakan juga intent untuk membuat aplikasi yang berhubungan dengan halaman lain.

Sebuah aplikasi tidak mungkin hanya melibatkan satu komponen dalam pembuatannya. Untuk itu, perlu untuk mempelajari bagaimana membuat aplikasi yang mengintegrasikan beberapa komponen. Project ini akan digunakan untuk mengintegrasikan kompomenkomponen membentuk aplikasi yang fungsional. Selain itu, digunakan juga intent untuk membuat aplikasi yang berhubungan dengan halaman lain.

Apa itu intents? Android menggunakan intents untuk melakukan pekerjaan tertentu di dalam aplikasinya. Begitu kita menguasai penggunaan intents, maka semua pengembangan aplikasi baru yang ada akan terbuka. Pada pertemuan ke 6 ini akan membahas tentang apa intents itu dan bagaimana dia digunakan. Intents adalah sebuah metode android untuk me-relay informasi tertentu dari satu aktivitas ke aktivitas lainnya. Secara lebih sederhana, Intents mengekspresikan kepada android untuk melakukan sesuatu.



Kita bisa menganggap intent sebagai sebuah pesan yang dilewatkan diantara banyak aktivitas. Misalnya, mempunyai aktivitas yang mengharuskan untuk membuka web browser dan menampilkan sebuah halaman di perangkat android. Aktivitas kita akan mengirimkan “keinginan (intent) untuk membuka halaman x di web browser” yang dikenal dengan Intent WEB-SEARCH\_ACTION, ke Android IntentResolver. IntentResolver mengurai melalui sebuah daftar Aktivitas dan memilih salah satu yang paling cocok dengan Intent kita; dalam hal ini adalah Web Browser Activity. Lalu Intent Resolver mengirimkan halaman kita ke web browser dan memulai Web Browser Activity.

Intent dipecah ke dalam dua kategori utama:

* Activity Action Intents: Intent yang digunakan untuk memanggil Activity di luar aplikasi. Hanya satu Activity yang bisa ditangani oleh Intent. Misalnya saja untuk sebuah web browser, kita harus membuka Web Browser Activity untuk menampilkan halaman.
* Broadcast Intents: Intents yang dikirimkan untuk menangani lebih dari satu Activity. Contohnya, Broadcast intent yang akan menjadi sebuah pesan yang dikirimkan oleh Android mengenai tingkat baterei saat itu. Banyak aktivitas bisa memproses Intent ini dan melakukan reaksi yang sesuai – misalnya, membatalkan sebuah Activity bila tingkat baterei berada di bawah titik tertentu.

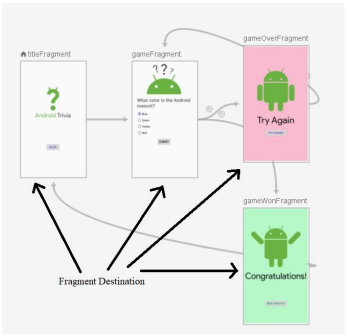
## **Menu dan Dialog**

**Komponen Navigasi.**

Navigation Graph (resource XML baru) adalah sumber daya yang berisi semua informasi terkait navigasi di satu lokasi terpusat, termasuk semua tempat di aplikasi, yang dikenal sebagai tujuan, dan kemungkinan jalur yang dapat dilalui pengguna melalui aplikasi. NavHostFragment (Layout XML view) adalah widget khusus yang ditambahkan ke layout, yang menampilkan berbagai tujuan dari Navigation Graph. NavController (objek Kotlin / Java) adalah objek yang melacak posisi saat ini dalam Navigation Graph, yang mengatur pertukaran konten tujuan di NavHostFragment saat kita bergerak melalui Navigation Graph. Saat kita menavigasi, kita akan menggunakan objek NavController, memberi tahu ke mana kita ingin pergi atau jalur apa yang ingin kita ambil dalam Navigation Graph. NavController kemudian akan menunjukkan tujuan yang sesuai di NavHostFragment.

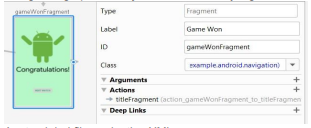
**Destinasi Navigation Graph.**

Komponen Navigasi memperkenalkan konsep destination. Destination adalah tempat apa pun yang dapat kita navigasi di aplikasi, biasanya sebuah fragment atau activity. Navigation Graph adalah jenis sumber daya baru yang mendefinisikan semua jalur yang mungkin seorang pengguna dapat ambil melalui aplikasi. Ini menunjukkan secara visual semua tujuan yang dapat dicapai dari tujuan tertentu. Android Studio menampilkan grafik di Editor Navigasinya. Contoh Navigation Graph adalah sebagai berikut.



Mengeksplorasi Navigation Editor

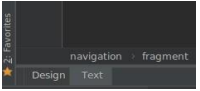
1. Buka res/navigation/mobile\_navigation.xml
2. Klik Design untuk menuju mode Design. Navigation graph menunjukkan destination yang ada.



Anatomi dari file navigation XML.

Semua perubahan yang dibuat di Editor Navigasi grafis mengubah file XML yang menyertainya, mirip dengan cara Editor Layout memodifikasi layout XML.

Klik tab Text:



Akan muncul file xml berikut:

<navigation 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/navigation"

app:startDestination="@id/titleFragment">

<!-- ...tags for fragments and activities here -->

</navigation> catatan:

<navigation> adalah root node dari setiap navigation graph.

<navigation> terdiri satu atau lebih destination, direpresentasikan dengan

elemen <activity> atau <fragment> .

app:startDestination adalah atribut yang menentukan destination yang

dijalankan secara default ketika pengguna membuka app pertama kali.

Untuk sebuah fragment destination:

<fragment

android:id="@+id/titleFragment" android:name=

"com.example.android.navigation.TitleFragment"

android:label="Intro"

tools:layout="@layout/fragment\_title">

<action

android:id="@+id/action\_titleFragment\_to\_gameFragment"

app:destination="@id/gameFragment" />

</fragment>

**Catatan:**

android:id mendefinisikan ID untuk fragmen yang dapat digunakan untuk

referensi destination di tempat lain dalam XML ini dan kode kita.

android: name mendeklarasikan nama kelas yang memenuhi syarat dari fragment

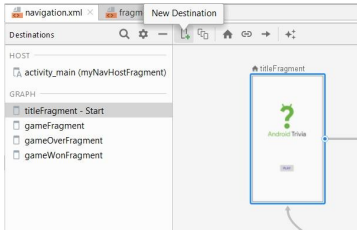
untuk dipakai saat menavigasi ke destinasi tersebut.

tools:layout menentukan layout apa yang harus ditampilkan dalam editor grafis.

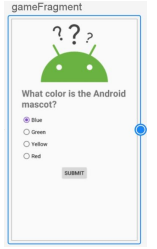
Menambahkan Destination ke Navigation Graph.

Aplikasi sampel dimulai dengan beberapa destination dalam grafik. Pada langkah ini, kita akan menambahkan destination baru! Kita harus menambahkan destination ke Navigation Graph sebelum dapat menavigasi ke sana.

1. Buka res/navigation/navigation.xml, dan klik tab Design.
2. Klik ikon New Destination, dan pilih "gameFragment"



Kita akan peroleh sebagai berikut.



<fragment

android:id="@+id/gameFragment"

android:name="com.example.android.navigation.GameFragment"

android:label="Game"

tools:layout="@layout/fragment\_game">

</fragment>

Menggunakan Navigation Graph untuk menavigasi.

Saat ini kita memiliki navigation graph yang luar biasa ini, tetapi kita sebenarnya tidak menggunakannya untuk menavigasi.

Activities dan Navigation

Komponen Navigasi mengikuti panduan yang dijabarkan dalam Prinsip Navigasi. Prinsip Navigasi merekomendasikan kita menggunakan aktivitas sebagai titik masuk untuk aplikasi kita. Aktifitas juga akan berisi navigasi global, seperti bottom nav. Sebagai perbandingan, fragment akan menjadi layout spesifik tujuan yang sebenarnya. Agar semua ini berfungsi, kita perlu memodifikasi layout aktivitas kita untuk memuat widget khusus yang disebut NavHostFragment. NavHostFragment menukar destination fragment yang berbeda masuk dan keluar saat kita menavigasi navigation graph.



Layout sederhana yang mendukung navigasi mirip dengan gambar di atas terlihat seperti ini:

<LinearLayout

.../>

<androidx.appcompat.widget.Toolbar

.../>

<fragment

android:id="@+id/myNavHostFragment" android:name=

"androidx.navigation.fragment.NavHostFragment"

android:layout\_width="match\_parent" android:layout\_height="match\_parent"

app:defaultNavHost="true"

app:navGraph="@navigation/navigation" />

<com.google.android.material.bottomnavigation.BottomNavigationView

.../>

</LinearLayout>

Catatan:

* Ini adalah layout for an activity, berisi navigation global, termasuk sebuah bottom nav dan a toolbar
* android:name="androidx.navigation.fragment.NavHostFragment" and app:defaultN avHost="true" menghubungkan back button sistem ke NavHostFragment

app:navGraph="@navigation/navigation" mengaitkan NavHostFragment dengan navigation graph. Navigation graph ini menentukan semua destination pengguna dapat menavigasi ke, di fragment NavHostFragment ini.

NavController

Akhirnya, ketika pengguna melakukan sesuatu seperti mengklik tombol, kita perlu memicu perintah navigasi. Kelas khusus yang disebut NavController adalah apa yang memicu swap fragmen di NavHostFragment.

// Command untuk navigasi ke gameFragment\_to\_gameWonFragment findNavController()

.navigate(R.id.action\_gameFragment\_to\_gameWonFragment)

Perhatikan bahwa kita memberikan destination ID atau action untuk menavigasi. Ini adalah ID yang ditentukan dalam navigation graph XML. Ini adalah contoh memberikan destination ID. NavController sangat powerfull karena ketika kita memanggil metode seperti navigate() atau popBackStack (), NavController menerjemahkan perintah-perintah ini ke dalam operasi framework yang sesuai berdasarkan jenis destination yang kita navigasikan ke atau dari. Misalnya, ketika kita memanggil navigate() dengan sebuah activity destination, NavController memanggil startActivity () atas nama kita. Ada beberapa cara untuk mendapatkan objek NavController yang terkait dengan NavHostFragment. Di Kotlin, disarankan untuk menggunakan salah satu fungsi ekstensi berikut, tergantung pada apakah akan memanggil perintah navigasi dari dalam sebuah fragment, activity atau view:

* Fragment.findNavController()
* View.findNavController()
* Activity.findNavController(viewId: Int)

NavController dikaitkan dengan NavHostFragment. Jadi, metode apa pun yang digunakan, kita harus yakin bahwa fragment, view, atau view ID adalah NavHostFragment itu sendiri, atau memiliki NavHostFragment sebagai induk. Kalau tidak, kita akan mendapatkan IllegalStateException.

Navigasi ke sebuah Destination dengan NavController

Kita akan bernavigasi menggunakan NavController dengan menghubungkan tombol Navigate Ke Destination untuk menavigasi ke destination action\_game Fragment\_to\_gameWon Fragment (yang merupakan tujuan adalah GameWonFragment):

1. Buka TittleFragment.kt
2. Sambungkan navigate\_destination\_button dalam onViewCreated()

TittleFragment.kt

playButton.setOnClickListener { view : View -> view.findNavController().

navigate(R.id.action\_titleFragment\_to\_gameFragment) }

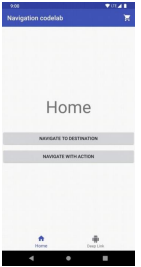
2. Jalankan app dan Navigate To Destination button. Button menavigasi ke game\_fragment destination.

Kita juga dapat menggunakan metode Navigation. Create Navigate OnClickListener (@IdRes destId: int, bundle: Bundle). Metode ini akan membangun OnClickListener untuk menavigasi ke tujuan yang diberikan, dengan sekumpulan argumen untuk diteruskan ke tujuan. Kodenya sebagai berikut.

val button = view.findViewById(R.id.navigate\_destination\_button) button?.setOnClickListener( Navigation.createNavigateOnClickListener(R.id.game\_fragment, null) )

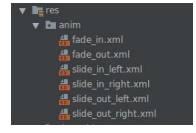
Mengubah transisi Navigation

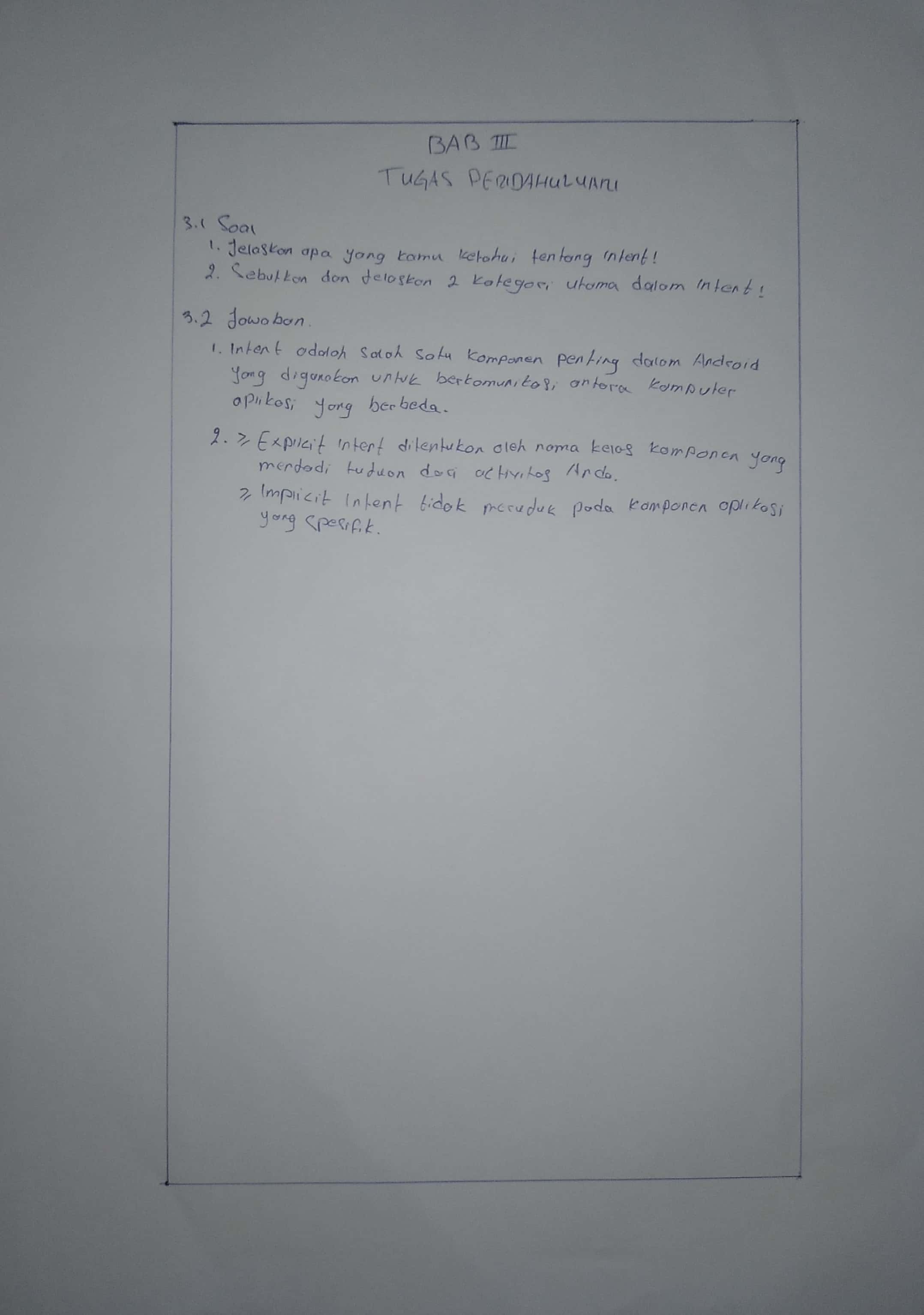
Setiap panggilan navigate() memiliki transisi standar yang tidak terlalu menarik yang terkait dengannya, seperti terlihat di bawah:



Transisi default, serta atribut lain yang terkait dengan panggilan, dapat diganti dengan menyertakan satu set NavOptions. NavOptions menggunakan pola Builder yang memungkinkan kita mengganti dan menetapkan hanya opsi yang dibutuhkan. Ada juga ktx DSL untuk NavOptions, yang akan digunakan.

Untuk transisi animasi, kita dapat menentukan resource animation XML di folder resource anim dan kemudian menggunakan animation tersebut untuk transisi. Beberapa contoh termasuk dalam kode app:





**BAB IV  
IMPLEMENTASI**

## **Tugas Praktikum**

1. Buat aplikasi yang menerapkan Intent dan Room Database

## **Source Code**

1. Activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout 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="match\_parent"

tools:context=".MainActivity">

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/recycler\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

<com.google.android.material.floatingactionbutton.FloatingActionButton

android:id="@+id/fab"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentEnd="true"

android:layout\_alignParentBottom="true"

android:layout\_margin="16dp"

android:src="@drawable/baseline\_add\_24"/>

</RelativeLayout>

1. MainActivity.kt

package com.example.praktikummodul34

import android.annotation.SuppressLint

import android.content.DialogInterface

import android.content.Intent

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import androidx.appcompat.app.AlertDialog

import androidx.recyclerview.widget.DividerItemDecoration

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import androidx.recyclerview.widget.RecyclerView.VERTICAL

import com.example.praktikummodul34.adapter.UserAdapter

import com.example.praktikummodul34.data.AppDatabase

import com.example.praktikummodul34.data.entity.User

import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity() {

private lateinit var recyclerView: RecyclerView

private lateinit var fab:FloatingActionButton

private var list = mutableListOf<User>()

private lateinit var adapter: UserAdapter

private lateinit var database: AppDatabase

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

recyclerView = findViewById(R.id.recycler\_view)

fab = findViewById(R.id.fab)

database = AppDatabase.getInstance(applicationContext)

adapter = UserAdapter(list)

adapter.setDialog(object : UserAdapter.Dialog{

override fun onClick(position: Int) {

// membuat dialog view

val dialog = AlertDialog.Builder(this@MainActivity)

dialog.setTitle(list[position].fullName)

dialog.setItems(R.array.items\_option, DialogInterface.OnClickListener{ dialog, which ->

if (which==0){

// coding ubah

val intent = Intent(this@MainActivity, EditorActivity::class.java)

intent.putExtra("id", list[position].uid)

startActivity(intent)

} else if (which==1){

// coding hapus

database.userDao().delete(list[position])

getData()

} else {

// coding batal

dialog.dismiss()

}

})

// untuk menampilkan dialog

val dialogView = dialog.create()

dialogView.show()

}

})

recyclerView.adapter = adapter

recyclerView.layoutManager = LinearLayoutManager(applicationContext, VERTICAL, false)

recyclerView.addItemDecoration(DividerItemDecoration(applicationContext, VERTICAL))

fab.setOnClickListener {

startActivity(Intent(this, EditorActivity::class.java))

}

}

override fun onResume() {

super.onResume()

getData()

}

@SuppressLint("NotifyDataSetChanged")

fun getData(){

list.clear()

list.addAll(database.userDao().getAll())

adapter.notifyDataSetChanged()

}

}

1. Activity\_editor.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout 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="match\_parent"

android:orientation="vertical"

android:padding="14dp"

tools:context=".EditorActivity">

<com.google.android.material.textfield.TextInputLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:boxBackgroundMode="filled"

android:layout\_marginBottom="14dp">

<EditText

android:id="@+id/full\_name"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="textPersonName"

android:hint="Nama Lengkap"/>

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:boxBackgroundMode="filled"

android:layout\_marginBottom="14dp">

<EditText

android:id="@+id/email"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="textEmailAddress"

android:hint="Alamat Email"/>

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:boxBackgroundMode="filled"

android:layout\_marginBottom="14dp">

<EditText

android:id="@+id/phone"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="phone"

android:hint="Nomor Telephon"/>

</com.google.android.material.textfield.TextInputLayout>

<Button

android:id="@+id/btn\_save"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Simpan"/>

</LinearLayout>

1. EditorActivity.kt

package com.example.praktikummodul34

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.widget.Button

import android.widget.EditText

import android.widget.Toast

import androidx.room.RoomDatabase

import com.example.praktikummodul34.data.AppDatabase

import com.example.praktikummodul34.data.entity.User

class EditorActivity : AppCompatActivity() {

private lateinit var fullName: EditText

private lateinit var email: EditText

private lateinit var phone: EditText

private lateinit var btnSave: Button

private lateinit var database: AppDatabase

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_editor)

fullName = findViewById(R.id.full\_name)

email = findViewById(R.id.email)

phone = findViewById(R.id.phone)

btnSave = findViewById(R.id.btn\_save)

database = AppDatabase.getInstance(applicationContext)

val intent = intent.extras

if (intent!=null){

val id = intent.getInt("id", 0)

val user = database.userDao().get(id)

fullName.setText(user.fullName)

email.setText(user.email)

phone.setText(user.phone)

}

btnSave.setOnClickListener {

if (fullName.text.isNotEmpty() && email.text.isNotEmpty() && phone.text.isNotEmpty()) {

if (intent!==null){

// coding edit data

database.userDao().update(

User(

intent.getInt("id", 0),

fullName.text.toString(),

email.text.toString(),

phone.text.toString()

)

)

}else{

// coding tambah data

database.userDao().insertAll(

User(

null,

fullName.text.toString(),

email.text.toString(),

phone.text.toString()

)

)

}

finish()

} else {

Toast.makeText(

applicationContext,

"Silahkan Isi Semua Data Dengan valid",

Toast.LENGTH\_SHORT

).show()

}

}

}

}

1. Rowuser.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:padding="14dp">

<TextView

android:id="@+id/full\_name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textStyle="bold" />

<TextView

android:id="@+id/email"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<TextView

android:id="@+id/phone"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</LinearLayout>

1. UserAdapter.kt

package com.example.praktikummodul34.adapter

import android.view.LayoutInflater

import android.view.TextureView

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

import com.example.praktikummodul34.data.entity.User

import com.example.praktikummodul34.R;

class UserAdapter(var list: List<User>) : RecyclerView.Adapter<UserAdapter.ViewHolder>() {

private lateinit var dialog: Dialog

fun setDialog(dialog: Dialog){

this.dialog = dialog

}

interface Dialog{

fun onClick(position: Int)

}

inner class ViewHolder(view: View) : RecyclerView.ViewHolder(view){

var fullname: TextView

var email: TextView

var phone: TextView

init {

fullname = view.findViewById(R.id.full\_name)

email = view.findViewById(R.id.email)

phone = view.findViewById(R.id.phone)

view.setOnClickListener{

dialog.onClick(layoutPosition)

}

}

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {

val view = LayoutInflater.from(parent.context).inflate(R.layout.row\_user, parent, false)

return ViewHolder(view)

}

override fun getItemCount(): Int {

return list.size

}

override fun onBindViewHolder(holder: ViewHolder, position: Int) {

holder.fullname.text = list[position].fullName

holder.email.text = list[position].email

holder.phone.text = list[position].phone

}

}

1. AppDatabase.kt

package com.example.praktikummodul34.data

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

import com.example.praktikummodul34.data.dao.UserDao

import com.example.praktikummodul34.data.entity.User

@Database(entities = [User::class], version = 1)

abstract class AppDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

companion object{

private var instant: AppDatabase? = null

fun getInstance(context: Context): AppDatabase{

if(instant==null){

instant = Room.databaseBuilder(context, AppDatabase::class.java, "app-database")

.fallbackToDestructiveMigration()

.allowMainThreadQueries()

.build()

}

return instant!!

}

}

}

1. UserDao.kt

package com.example.praktikummodul34.data.dao

import androidx.room.\*

import com.example.praktikummodul34.data.entity.User

@Dao

interface UserDao {

//untukmengambilsemuadata//

@Query("SELECT \* FROM user")

fun getAll(): List<User>

//mengambildataberdasarkanuserID

@Query("SELECT \* FROM user WHERE uid IN (:userIds)")

fun loadAllByIds(userIds: IntArray): List<User>

@Insert

fun insertAll(vararg users: User)

@Delete

fun delete(user: User)

@Query("SELECT \* FROM user WHERE uid = :uid")

fun get(uid: Int) : User

@Update

fun update(user: User)

}

1. String.xml

<resources>

<string name="app\_name">praktikummodul34</string>

<string-array name="items\_option">

<item>Ubah</item>

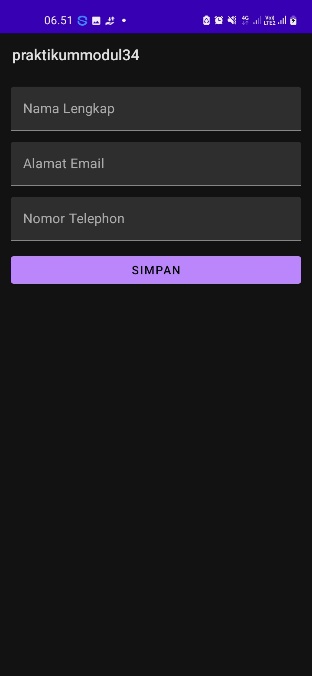
<item>Hapus</item>

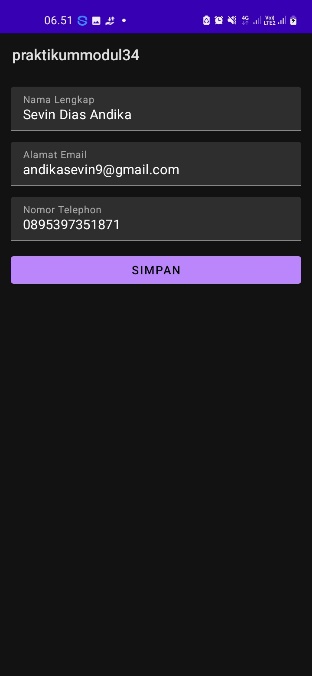
<item>Batal</item>

</string-array>

</resources>

## **Hasil**





# **BAB V PENUTUP**

## **Analisa**

Dari hasil praktikum, praktikan menganalisa bahwa Intents adalah sebuah metode android untuk me-relay informasi tertentu dari satu aktivitas ke aktivitas lainnya. Secara lebih sederhana, Intents mengekspresikan kepada android untuk melakukan sesuatu. Navigation Graph (resource XML baru) adalah sumber daya yang berisi semua informasi terkait navigasi di satu lokasi terpusat, termasuk semua tempat di aplikasi, yang dikenal sebagai tujuan, dan kemungkinan jalur yang dapat dilalui pengguna melalui aplikasi. NavHostFragment (Layout XML view) adalah widget khusus yang ditambahkan ke layout, yang menampilkan berbagai tujuan dari Navigation Graph. NavController (objek Kotlin / Java) adalah objek yang melacak posisi saat ini dalam Navigation Graph, yang mengatur pertukaran konten tujuan di NavHostFragment saat kita bergerak melalui Navigation Graph.

Destination adalah tempat apa pun yang dapat kita navigasi di aplikasi, biasanya sebuah fragment atau activity. Navigation Graph adalah jenis sumber daya baru yang mendefinisikan semua jalur yang mungkin seorang pengguna dapat ambil melalui aplikasi. Ini menunjukkan secara visual semua tujuan yang dapat dicapai dari tujuan tertentu.

## **Kesimpulan**

1. Intents adalah sebuah metode android untuk me-relay informasi tertentu dari satu aktivitas ke aktivitas lainnya. Secara lebih sederhana, Intents mengekspresikan kepada android untuk melakukan sesuatu.
2. Activity Action Intents: Intent yang digunakan untuk memanggil Activity di luar aplikasi. Hanya satu Activity yang bisa ditangani oleh Intent. Misalnya saja untuk sebuah web browser, kita harus membuka Web Browser Activity untuk menampilkan halaman.
3. Broadcast Intents: Intents yang dikirimkan untuk menangani lebih dari satu Activity.