Aplikasi dengan Navigasi Dinamis

Berikut tutorial cara membuat navigasi antarlayar tanpa Grafik Navigasi untuk aplikasi sederhana dengan Android Studio dengan bahasa Kotlin menggunakan Jetpack Compose.

1. Membuat Proyek Android Studio Baru dengan Jetpack Compose

1. **Buka Android Studio**.
2. Klik **"Start a new Android Studio project"**.
3. Pilih template **"Empty Compose Activity"**, lalu klik **"Next"**.
4. Isi detail proyek:
    - **Name**: `NavigasiSederhana`
    - **Package name**: `com.example.navigasisederhana`
    - **Save location**: Pilih lokasi yang diinginkan.
    - **Language**: **Kotlin**
    - **Minimum SDK**: Pilih sesuai kebutuhan.
5. Klik **"Finish"** dan tunggu hingga proyek selesai dibuat.

2. Membuat Layar (Screens) dengan Jetpack Compose

Kita akan membuat dua layar sederhana: **Main Screen** dan **Second Screen**.

a. Membuat File Kotlin untuk Layar

2. **Buat Package Baru (Opsional)**:
    - Klik kanan pada direktori `com.example.navigasisederhana`.
    - Pilih **New** > **Package**.
    - Beri nama `screens`.
3. **Buat File Kotlin untuk Main Screen**:
    - Klik kanan pada package `screens`.
    - Pilih **New** > **Kotlin File/Class**.
    - Beri nama `MainScreen`.
    - Beri pilihan **File**

```
// MainScreen.kt
package com.example.navigasisederhana.screens

import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
```

```kotlin
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

@Composable
fun MainScreen(onNavigate: () -> Unit) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Ini adalah Main Screen",
            fontSize = 24.sp,
            modifier = Modifier.padding(bottom = 20.dp)
        )
        Button(onClick = onNavigate) {
            Text(text = "Second Screen")
        }
    }
}

@Preview(showBackground = true)
@Composable
fun MainScreenPreview() {
        MainScreen({})
}
```

4. **Buat File Kotlin untuk Second Screen**:
    - Klik kanan pada package `screens`.
    - Pilih **New** > **Kotlin File/Class**.
    - Beri nama `SecondScreen`.
    - Beri pilihan **File**

```kotlin
// SecondScreen.kt
package com.example.navigasisederhana.screens

import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
```

```
@Composable
fun SecondScreen(onNavigateBack: () -> Unit) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Ini adalah Second Screen",
            fontSize = 24.sp,
            modifier = Modifier.padding(bottom = 20.dp)
        )
        Button(onClick = onNavigateBack) {
            Text(text = "Kembali ke Main Screen")
        }
    }
}

@Preview(showBackground = true)
@Composable
fun SecondScreenPreview() {
    SecondScreen({})
}
```

b. Menambahkan Layar ke `MainActivity`

Buka `MainActivity.kt` dan modifikasi agar dapat menampilkan layar sesuai navigasi yang diinginkan.

```
// MainActivity.kt
package com.example.navigasisederhana

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.navigasisederhana.screens.MainScreen
import com.example.navigasisederhana.screens.SecondScreen
import com.example.navigasisederhana.ui.theme.NavigasiSederhanaTheme

class MainActivity : ComponentActivity() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            NavigasiSederhanaTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) {
innerPadding ->
                    MyApp()
                }
            }
        }
    }
}

@Composable
fun MyApp() {
    // Mengelola state navigasi
    var currentScreen by remember {
mutableStateOf<Screen>(Screen.Main) }

    when (currentScreen) {
        Screen.Main -> MainScreen (
            onNavigate = { currentScreen = Screen.Second }
        )
        Screen.Second -> SecondScreen (
            onNavigateBack = { currentScreen = Screen.Main }
        )
    }
}

enum class Screen {
    Main, Second
}
```

Penjelasan Kode:

- **State Management**:
  - Kita menggunakan:
    ```
    var currentScreen by remember { mutableStateOf<Screen>(Screen.Main) }
    ```
    Ini untuk menyimpan state layar saat ini.
  - Ketika state **currentScreen** berubah, Jetpack Compose akan merender ulang UI sesuai dengan layar yang ditentukan.
- **Navigasi**:
  - **MainScreen** memiliki callback **onNavigate** yang mengubah **currentScreen** menjadi **Screen.*Second***.
  - **SecondScreen** memiliki callback **onNavigateBack** yang mengubah **currentScreen** kembali menjadi **Screen.*Main***.

3. Mengelola Navigasi dengan State

Pendekatan ini menggunakan **state** untuk mengelola navigasi antar layar. Ini adalah metode yang sederhana dan efektif untuk aplikasi dengan navigasi yang tidak terlalu kompleks.

a. State untuk Menentukan Layar yang Ditampilkan

Kita menggunakan `mutableStateOf` untuk menyimpan status layar saat ini dan `remember` untuk mempertahankan state selama lifecycle composable.

```
var currentScreen by remember { mutableStateOf<Screen>(Screen.Main) }
```

b. Menggunakan `when` untuk Menampilkan Layar yang Sesuai

Dengan `when`, kita dapat menentukan composable mana yang akan ditampilkan berdasarkan nilai `currentScreen`.

```
    when (currentScreen) {
        Screen.Main -> MainScreen (
            onNavigate = { currentScreen = Screen.Second }
        )
        Screen.Second -> SecondScreen (
            onNavigateBack = { currentScreen = Screen.Main }
        )
    }
```

c. Menambahkan Layar Tambahan (Opsional)

Jika ingin menambahkan lebih banyak layar, cukup tambahkan kondisi baru dalam `when` dan buat composable yang sesuai.

```
    when (currentScreen) {
        Screen.Main -> MainScreen (
            onNavigate = { currentScreen = Screen.Second }
        )
        Screen.Second -> SecondScreen (
            onNavigateBack = { currentScreen = Screen.Main }
        )
        Screen.Third -> ThirdScreen (
            onNavigateBack = { currentScreen = Screen.Second }
        )
    }
```

Dalam kode di atas `ThirdScreen` tidak pernah dipanggil. Kita bisa memanggilnya lewat `SecondScreen`. Kita perlu tambahkan sebuah button lagi di `SecondScreen` untuk memanggil `ThirdScreen`.

```
@Composable
fun SecondScreen(
    onNavigate: () -> Unit,
    onNavigateBack: () -> Unit
    ) {
    Column(
        modifier = Modifier
            .fillMaxSize()
```

```
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Ini adalah Second Screen",
            fontSize = 24.sp,
            modifier = Modifier.padding(bottom = 20.dp)
        )
        Button(onClick = onNavigate) {
            Text(text = "Third Screen")
        }
        Button(onClick = onNavigateBack) {
            Text(text = "Kembali ke Main Screen")
        }
    }
}
```

Selanjutnya dalam `MainActivity` di struktur when menjadi seperti berikut.

```
when (currentScreen) {
    Screen.Main -> MainScreen (
        onNavigate = { currentScreen = Screen.Second }
    )
    Screen.Second -> SecondScreen (
        onNavigate = { currentScreen = Screen.Third },
        onNavigateBack = { currentScreen = Screen.Main }
    )
    Screen.Third -> ThirdScreen (
        onNavigateBack = { currentScreen = Screen.Second }
    )
}
```

4. Mengirim Data antar Layar

Seringkali, kita perlu mengirim data dari satu layar ke layar lainnya. Berikut adalah cara melakukannya dengan menggunakan parameter pada composable.

a. Memodifikasi MainScreen untuk Mengirim Data

Misalkan, kita ingin mengirim pesan dari `MainScreen` ke `SecondScreen`.

1. **Update `MainScreen.kt`**:

   Tambahkan `TextField` untuk memasukkan pesan.

```
@Composable
fun MainScreen(onNavigate: (String) -> Unit) {
    var message by remember { mutableStateOf("") }

    Column(
```

```
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Ini adalah Main Screen",
            fontSize = 24.sp,
            modifier = Modifier.padding(bottom = 20.dp)
        )
        TextField(
            value = message,
            onValueChange = { message = it },
            label = { Text("Masukkan Pesan") },
            modifier = Modifier
                .fillMaxWidth()
                .padding(bottom = 20.dp)
        )
        Button(onClick = { onNavigate(message) }) {
            Text(text = "Pergi ke Second Screen")
        }
    }
}
```

Jangan lupa tambahkan import berikut.

```
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.material3.TextField
```

2. **Update** MainActivity.kt:

Di fungsi `MyApp()` sesuaikan callback `onNavigate` untuk menerima pesan.

```
@Composable
fun MyApp() {
    // Mengelola state navigasi
    var currentScreen by remember {
mutableStateOf<Screen>(Screen.Main) }
    var messageToSend by remember { mutableStateOf("")}

    when (currentScreen) {
        Screen.Main -> MainScreen (
            onNavigate = { message ->
                messageToSend = message
                currentScreen = Screen.Second
            }
        )
```

```
        Screen.Second -> SecondScreen (
            message = messageToSend,
            onNavigate = { currentScreen = Screen.Third },
            onNavigateBack = { currentScreen = Screen.Main }
        )
        Screen.Third -> ThirdScreen (
            onNavigateBack = { currentScreen = Screen.Second }
        )
    }
}
```

b. Memodifikasi **SecondScreen** untuk Menerima Data

1. **Update `SecondScreen.kt`**:

   Tambahkan parameter `message` untuk menampilkan pesan yang diterima.

```
@Composable
fun SecondScreen(
    message: String,
    onNavigate: () -> Unit,
    onNavigateBack: () -> Unit
) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Ini adalah Second Screen",
            fontSize = 24.sp,
            modifier = Modifier.padding(bottom = 20.dp)
        )
        Text(
            text = "Pesan: $message",
            fontSize = 18.sp,
            modifier = Modifier.padding(bottom = 20.dp)
        )
        Button(onClick = onNavigate) {
            Text(text = "Third Screen")
        }
        Button(onClick = onNavigateBack) {
            Text(text = "Kembali ke Main Screen")
        }
    }
}
```

```
@Preview(showBackground = true)
@Composable
fun SecondScreenPreview() {
    SecondScreen(
        message = "",
        onNavigate={},
        onNavigateBack={})
}
```

Penjelasan Kode:

- **MainScreen**:
  - Menggunakan `TextField` untuk memasukkan pesan.
  - Callback `onNavigate` sekarang menerima parameter `String` untuk pesan.
- **MainActivity**:
  - Menyimpan pesan yang dimasukkan di `MainScreen` ke dalam `messageToSend`.
  - Mengoper `messageToSend` ke `SecondScreen`.
- **SecondScreen**:
  - Menerima `message` sebagai parameter dan menampilkannya.

---

5. Menjalankan dan Menguji Aplikasi

Sekarang, mari kita jalankan aplikasi dan uji navigasi antar layar.

Langkah-Langkah:

1. **Klik tombol "Run"** (ikon segitiga hijau) di Android Studio atau tekan **`Shift + F10`**.
2. **Pilih perangkat emulator atau fisik** untuk menjalankan aplikasi.
3. Setelah aplikasi diluncurkan, kamu akan melihat **Main Screen** dengan `TextField` dan tombol **"Second Screen"**.
4. **Masukkan pesan** di `TextField`.
5. **Klik tombol "Second Screen"**, dan kamu akan dinavigasi ke **Second Screen** yang menampilkan pesan yang kamu masukkan.
6. **Klik tombol "Kembali ke Main Screen"**, dan kamu akan kembali ke **Main Screen**.

---

6. Penanganan Back Stack

Dalam pendekatan ini, navigasi diatur secara manual, sehingga kita juga perlu mengelola back stack jika diperlukan. Namun, untuk aplikasi sederhana, kita bisa mengandalkan **Android's default back stack**.

a. Menggunakan Tombol Back di Perangkat

- **Dari Second Screen**:

    - Menekan tombol **Back** pada perangkat akan mengembalikan pengguna ke **Main Screen**.
- **Dari Main Screen**:
    - Menekan tombol **Back** akan keluar dari aplikasi.

b. Mengelola Back Stack secara Manual (Opsional)

Jika kamu ingin memiliki kontrol lebih atas back stack, kamu bisa menggunakan **stack data structure** untuk menyimpan riwayat layar yang telah dikunjungi.

Namun, untuk kesederhanaan, kita tidak akan memasukkan implementasi ini dalam tutorial ini. Pendekatan default sudah memadai untuk navigasi sederhana.

---

7. Kesimpulan

Dalam tutorial ini, kamu telah belajar bagaimana cara **membuat navigasi antar layar tanpa menggunakan Grafik Navigasi (Navigation Graph)** dalam aplikasi Android sederhana menggunakan **Jetpack Compose** dan **Kotlin**. Berikut adalah poin-poin utama yang telah dibahas:

1. **Membuat Proyek Baru**: Menyiapkan proyek Android Studio dengan Jetpack Compose.
2. **Membuat Layar (Screens)**: Membuat dua composable functions, yaitu `MainScreen` dan `SecondScreen`.
3. **Mengelola Navigasi dengan State**: Menggunakan `mutableStateOf` untuk mengelola state navigasi dan menampilkan composable yang sesuai.
4. **Mengirim Data antar Layar**: Mengirim data dari `MainScreen` ke `SecondScreen` melalui parameter composable.
5. **Menjalankan dan Menguji Aplikasi**: Menguji navigasi antar layar dan pengiriman data.
6. **Penanganan Back Stack**: Memahami perilaku default back stack Android saat navigasi.