Navigasi Antar Layar dengan Compose

Navigasi Antar Layar dengan Compose adalah cara untuk berpindah dari satu tampilan (screen) ke tampilan lainnya dalam aplikasi Android yang dibangun menggunakan Jetpack Compose. Mirip seperti kita berpindah halaman di sebuah buku, navigasi ini memungkinkan pengguna untuk berinteraksi dengan berbagai bagian aplikasi secara intuitif.

Kenapa Perlu Navigasi Antar Layar?

- **Struktur Aplikasi yang Jelas:** Membantu mengatur aplikasi menjadi beberapa layar yang fokus pada tugas tertentu, sehingga lebih mudah dikelola dan dipahami.
- **Pengalaman Pengguna yang Baik:** Memungkinkan pengguna untuk berpindah antar fitur dengan mulus, meningkatkan pengalaman pengguna secara keseluruhan.
- **Pengelolaan State yang Lebih Baik:** Setiap layar dapat memiliki state-nya sendiri, sehingga memudahkan dalam mengelola data dan menghindari konflik.

Komponen Utama:

- **NavHost:** Komponen utama yang menampung semua destinasi (screen) dalam aplikasi.
- **NavController:** Mengontrol navigasi antar destinasi.
- **NavGraph:** Representasi grafis dari struktur navigasi aplikasi, menunjukkan hubungan antar destinasi.

Cara Kerja:

1. **Definisikan Destinasi:** Setiap layar dalam aplikasi dianggap sebagai destinasi. Masing-masing destinasi didefinisikan dengan composable function.
2. **Buat NavGraph:** Buat sebuah NavGraph yang menghubungkan semua destinasi. NavGraph ini akan menentukan bagaimana pengguna dapat berpindah dari satu destinasi ke destinasi lainnya.
3. **Gunakan NavHost:** Tempatkan NavHost di dalam composable root aplikasi Anda dan hubungkan dengan NavGraph.
4. **Navigasi:** Gunakan NavController untuk menavigasi antar destinasi. Anda bisa menggunakan fungsi seperti navigate() untuk berpindah ke destinasi tertentu.

Contoh Sederhana:

1. Tambahkan Dependensi

Pastikan Anda telah menambahkan dependensi `navigation-compose` di file `build.gradle`:

```
implementation ("androidx.navigation:navigation-compose:2.8.4")
```

2. Buat Composable untuk Setiap Layar

Setiap layar diwakili oleh sebuah fungsi `@Composable`. Contohnya:

```
@Composable
fun HomeScreen(onNavigateToDetail: () -> Unit) {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "Home Screen",
            style = MaterialTheme.typography.headlineLarge
        )
        Button(onClick = onNavigateToDetail) {
            Text("Go to Detail")
        }
    }
}

@Composable
fun DetailScreen(onBack: () -> Unit) {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "Detail Screen",
            style = MaterialTheme.typography.headlineLarge)
        Button(onClick = onBack) {
            Text("Back to Home")
        }
    }
}
```

3. Gunakan NavController untuk Mengatur Navigasi

Buat **`NavHost`** untuk mendefinisikan rute layar:

```
@Composable
fun MainNavigation() {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = "home"
    ) {
        composable("home") {
            HomeScreen(onNavigateToDetail = {
                navController.navigate("detail")
            })
        }
        composable("detail") {
```

```
            DetailScreen(onBack = {
                navController.popBackStack()
            })
        }
    }
}
```

4. Panggil Navigasi di Fungsi `setContent`

Terakhir, panggil navigasi utama di dalam **`setContent`** di **`MainActivity`**:

```
setContent {
    NavigasiTheme {
        Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
            MainNavigation()
        }
    }
}
```

Penjelasan Inti

1. **NavController**: Objek yang bertugas mengatur navigasi antar layar.
2. **NavHost**: Komponen yang mendefinisikan semua rute aplikasi.
3. **composable()**: Fungsi untuk mendeklarasikan layar berdasarkan rute.

Contoh 2

Berikut adalah contoh lebih kompleks dengan beberapa fitur tambahan seperti **mengoper data antar layar**, **menampilkan daftar data**, dan **navigasi dengan parameter**.

Aplikasi dengan Daftar dan Detail

1. **Struktur Navigasi:**
    - **Screen A**: Menampilkan daftar item.
    - **Screen B**: Menampilkan detail item yang dipilih.

Langkah-Langkah Implementasi

1. Tambahkan Dependensi

Tambahkan `navigation-compose` di `build.gradle` seperti sebelumnya:

```
implementation ("androidx.navigation:navigation-compose:2.8.4")
```

2. Buat Model Data

Definisikan model data yang akan ditampilkan:

```
data class Item(val id: Int, val name: String, val description: String)
```

3. Buat Layar untuk Daftar (Screen A)

Tampilkan daftar item dengan navigasi ke layar detail. UI ini digunakan untuk menampilkan daftar item.

1. Dalam paket yang ada, buat file baru bernama `ListScreen.kt`.
2. Dalam file `ListScreen.kt` tulis kode seperti berikut.

```
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

@Composable
fun ListScreen(items: List<Item>, onItemClick: (Item) -> Unit) {
    LazyColumn {
        items(items) { item ->
            ListItem(item, onClick = { onItemClick(item) })
        }
    }
}

@Composable
fun ListItem(item: Item, onClick: () -> Unit) {
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .clickable { onClick() }
            .padding(16.dp)
    ) {
        Text(
            text = item.name,
            style = MaterialTheme.typography.bodyMedium)
    }
}
```

4. Buat Layar untuk Detail (Screen B)

UI ini digunakan untuk menampilkan detail berdasarkan parameter ID dari navigasi:

1. Dalam paket yang ada, buat file baru bernama `DetailScreen.kt`.
2. Dalam file `DetailScreen.kt` tulis kode seperti berikut.

```
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

@Composable
fun DetailScreen(items: List<Item>, itemId: Int, onBack: () -> Unit) {
    val selectedItem = items.find { it.id == itemId }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "${selectedItem?.name}",
            style = MaterialTheme.typography.bodyMedium
        )
        Text(
            text = "${selectedItem?.description}",
            style = MaterialTheme.typography.bodyMedium
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = onBack) {
            Text("Back")
        }
    }
}
```

5. Atur Navigasi di NavHost

Definisikan rute dan parameter navigasi.

1. Dalam paket yang ada, buat file baru bernama `NavigationApp.kt`.

2. Dalam file `NavigationApp.kt` tulis kode seperti berikut.

```
import androidx.compose.runtime.Composable
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument

@Composable
fun NavigationApp() {
    val navController = rememberNavController()
    val items = List(10) {
        Item(it, "Item $it", "Description for Item $it")
    }

    NavHost(
        navController = navController,
        startDestination = "list"
    ) {
        composable("list") {
            ListScreen(
                items = items,
                onItemClick = {
                    item -> navController.navigate("detail/${item.id}")
                }
            )
        }
        composable(
            route = "detail/{itemId}",
            arguments = listOf(navArgument("itemId") {
                type = NavType.IntType
            })
        ) { backStackEntry ->
            val itemId = backStackEntry.arguments?.getInt("itemId") ?: 0
            DetailScreen(
                items = items,
                itemId = itemId,
                onBack = {navController.popBackStack()}
            )
        }
    }
}
```

6. Set Content di **MainActivity**

Panggil navigasi di `setContent`:

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

```
        enableEdgeToEdge()
        setContent {
            Navigasi2Theme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    NavigationApp()
                }
            }
        }
    }
}
```

Penjelasan Tambahan

1. **Menggunakan `LazyColumn` untuk Daftar:** Komponen ini digunakan untuk menampilkan daftar yang bisa di-scroll secara efisien.
2. **Mengoper Data Antar Layar:** Data dikirim melalui rute parameter seperti `detail/{itemId}`.
3. **Argument Validasi:** `navArgument` memastikan tipe data parameter sesuai, misalnya `NavType.IntType`.
4. **Dynamic Content:** Detail layar mengambil data berdasarkan `itemId` yang dikirimkan.

Hasilnya:

- **Screen A**: Menampilkan daftar item. Klik item akan membuka layar detail.
- **Screen B**: Menampilkan detail item berdasarkan ID.

Contoh 3

Berikut adalah tutorial **navigasi antar layar** dengan **Jetpack Compose**, **ViewModel**, dan **pengelolaan data asinkron**. Tutorial ini lebih lengkap dan terstruktur untuk membantu Anda memahami cara menggabungkan navigasi dan pengelolaan data dengan benar.

1. Persiapan Proyek

1. Buat proyek baru.
2. Tambahkan dependensi berikut di `build.gradle (module :app)`:

```
implementation("androidx.navigation:navigation-compose:2.8.4")
implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.8.7")
```

Klik **Sync Now**

2. Struktur Aplikasi

Aplikasi ini memiliki dua layar utama:

1. **ListScreen**: Menampilkan daftar item.

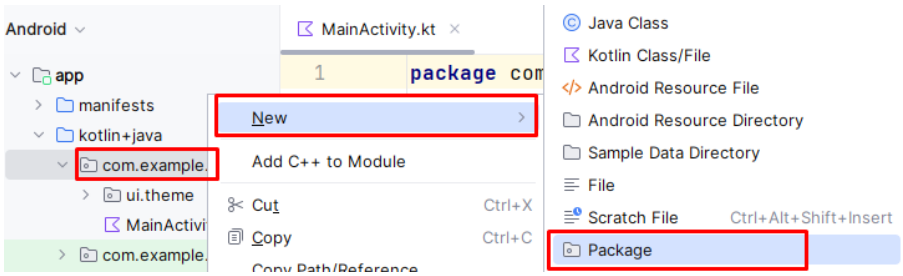
2. **DetailScreen**: Menampilkan detail item yang dipilih.

Data dikelola menggunakan **ViewModel** untuk memastikan UI responsif dan terpisah dari logika bisnis.

3. Model Data

Definisikan model data untuk aplikasi. Model data berisi: id, name, dan description.

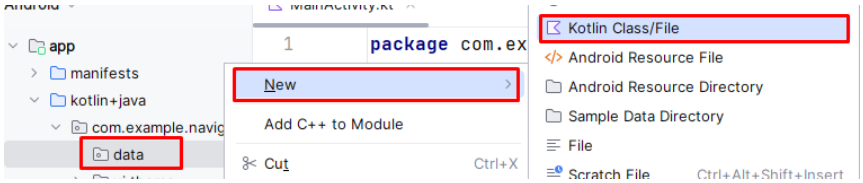
1. Dalam paket yang ada, buat paket baru dengan nama `data`. Klik kanan pada paket yang ada, pilih menu **New > Package**.



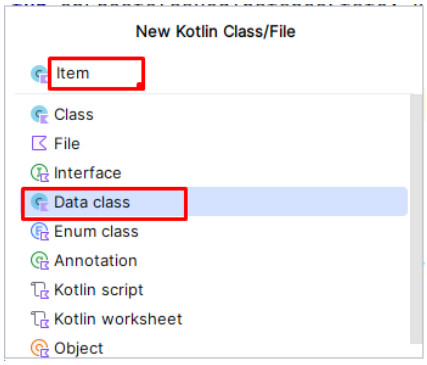
Kemudian tuliskan/tambahkan `data`, klik <enter>

2. Dalam paket `data` ini, buat kelas data baru dengan nama `Item`.

   Klik kanan pada paket data, pilih menu **New > Kotlin Class/File**



3. Dalam jendela New Kotlin Class/File, pilih opsi **Data class**, dan tulis `Item` pada isian yang tersedia, klik <enter>



4. Pada kelas `Item`, tulis kode seperti berikut.

```
data class Item(val id: Int, val name: String, val description: String)
```

4. ViewModel untuk Data

Buat ViewModel untuk mengelola data.

1. Dengan cara sama seperti sebelumnya, buat paket bernama `model`.
2. Dalam paket `model`, buat kelas baru dengan nama `ItemViewModel`.
3. Pada kelas `ItemViewModel`, tulis kode seperti berikut.

```
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.navigasi3.data.Item
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

class ItemViewModel : ViewModel() {
    var items: List<Item> = emptyList()
        private set

    private val _loading = MutableStateFlow(false)
    val loading: StateFlow<Boolean> = _loading

    init {
        fetchItems()
    }

    // Mencari item berdasar itemId
    fun getItem(itemId: Int): Item? {
        return items.find { it.id == itemId }
    }

    fun fetchItems() {
        viewModelScope.launch {
            _loading.value = true
            delay(2000) // Simulasi pengambilan data dari API
            items = List(10) {
                Item(it, "Item $it", "Description for Item $it")
            }
            _loading.value = false
        }
    }
}
```

5. UI untuk ListScreen

UI ini digunakan untuk menampilkan daftar item. Jika data masih dimuat, tampilkan indikator loading.

3. Dalam paket yang ada, buat file baru bernama `ListScreen.kt`.
4. Dalam file `ListScreen.kt` tulis kode seperti berikut.

```
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.navigasi3.data.Item
import com.example.navigasi3.model.ItemViewModel

@Composable
fun ListScreen(
    viewModel: ItemViewModel,
    onItemClick: (Item) -> Unit
) {
    val items = viewModel.items
    val isLoading by viewModel.loading.collectAsState()

    Box(modifier = Modifier.padding(16.dp).fillMaxSize()) {
        if (isLoading) {
            CircularProgressIndicator(
                modifier = Modifier.align(Alignment.Center)
            )
        } else {
            LazyColumn(modifier = Modifier.fillMaxSize()) {
                items(items) { item ->
                    ListItem(item, onClick = { onItemClick(item) })
                }
            }
        }
    }
}

@Composable
fun ListItem(item: Item, onClick: () -> Unit) {
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .clickable { onClick() }
            .padding(16.dp)
```

```
    ) {
        Text(
            text = item.name,
            style = MaterialTheme.typography.displaySmall
        )
    }
}
```

6. UI untuk DetailScreen

UI ini digunakan untuk menampilkan detail berdasarkan ID item yang dipilih.

3. Dalam paket yang ada, buat file baru bernama `DetailScreen.kt`.
4. Dalam file `DetailScreen.kt` tulis kode seperti berikut.

```
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.navigasi3.model.ItemViewModel

@Composable
fun DetailScreen(
    viewModel: ItemViewModel,
    itemId: Int, onBack: () -> Unit
) {
    val item = viewModel.getItem(itemId)

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "${item?.name}",
            style = MaterialTheme.typography.displayMedium
        )
        Spacer(modifier = Modifier.height(8.dp))
        Text(
            text = "${item?.description}",
            style = MaterialTheme.typography.displaySmall
```

```
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = onBack) {
            Text("Back")
        }
    }
}
```

7. Navigasi Antar Layar

Gunakan `NavHost` untuk mengelola navigasi.

3. Dalam paket yang ada, buat file baru bernama `NavigationApp.kt`.
4. Dalam file `NavigationApp.kt` tulis kode seperti berikut.

```
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument
import com.example.navigasi3.model.ItemViewModel

@Composable
fun NavigationApp(modifier: Modifier = Modifier) {
    val navController: NavHostController = rememberNavController()
    val viewModel: ItemViewModel = viewModel()

    NavHost(
        navController = navController,
        startDestination = "list"
    ) {
        composable("list") {
            ListScreen(
                viewModel = viewModel,
                onItemClick = { item ->
                    navController.navigate("detail/${item.id}")
                }
            )
        }
        composable(
            route = "detail/{itemId}",
            arguments = listOf(navArgument("itemId") {
                type = NavType.IntType
            })
        ) { backStackEntry ->
            val itemId = backStackEntry.arguments?.getInt("itemId") ?: 0
            DetailScreen(
                viewModel = viewModel,
```

```
                    itemId = itemId,
                    onBack = { navController.popBackStack() })
            }
        }
}
```

8. Integrasi dengan MainActivity

Panggil `NavigationApp` di `MainActivity`:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Navigasi3Theme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    NavigationApp(
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}
```

Fitur yang Ditambahkan

1. **Navigasi Antar Layar**:
   - `ListScreen` ke `DetailScreen` menggunakan parameter `itemId`.
2. **Pengelolaan Data dengan ViewModel**:
   - Data dikelola di `ItemViewModel` dan disediakan secara reaktif.
3. **Loading State**:
   - Tampilkan indikator loading saat data diambil.

Hasil

1. **ListScreen**:
   - Menampilkan daftar item.
   - Klik item untuk navigasi ke layar detail.
2. **DetailScreen**:
   - Menampilkan detail berdasarkan ID item.