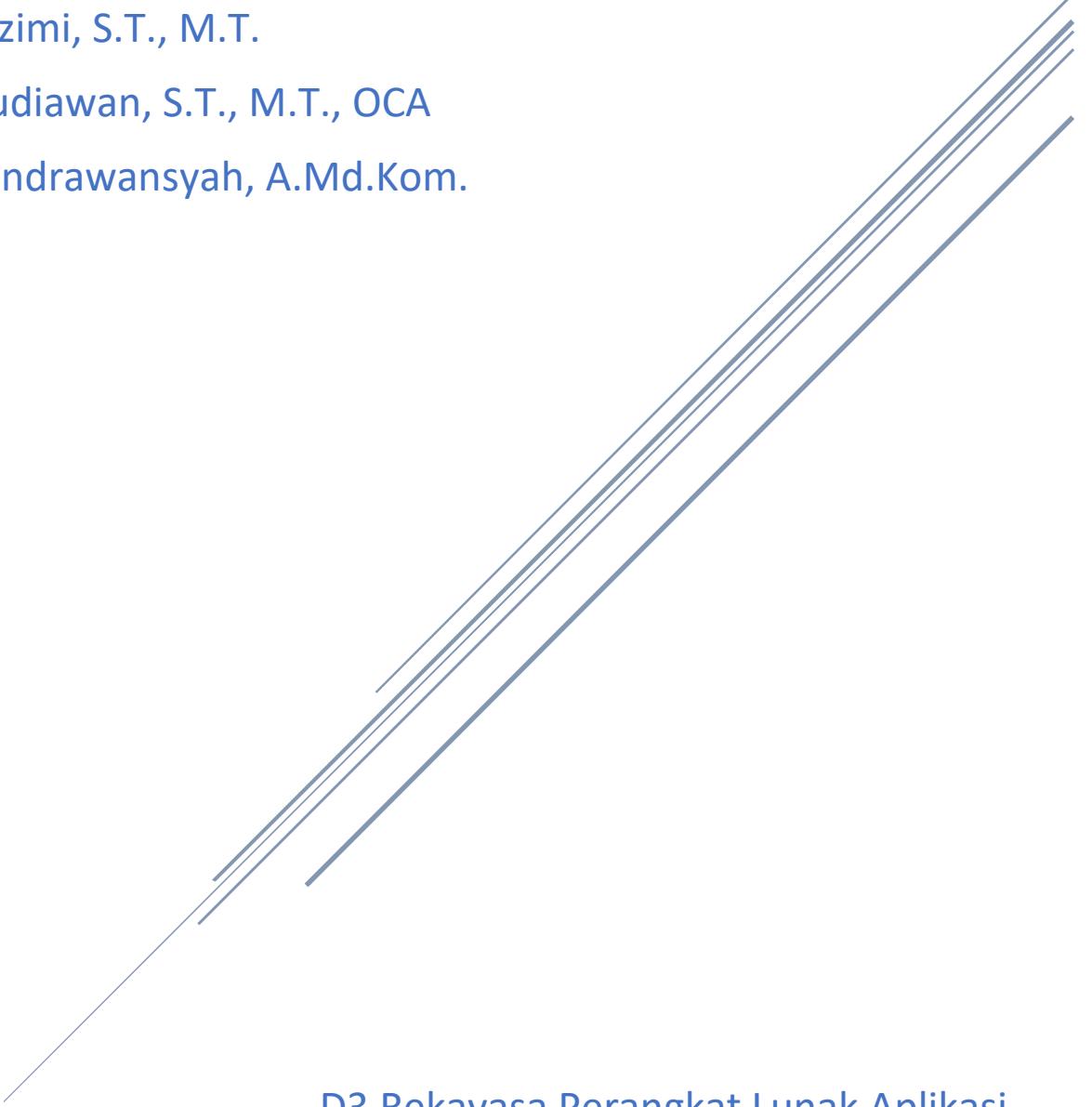


MODUL PRAKTIKUM PEMROGRAMAN UNTUK PERANGKAT BERGERAK 1

Indra Azimi, S.T., M.T.

Reza Budiawan, S.T., M.T., OCA

Dwiko Indrawansyah, A.Md.Kom.



D3 Rekayasa Perangkat Lunak Aplikasi
Telkom University

Daftar Isi

Modul 07: Scrollable List.....	1
1. Overview	1
2. Getting Started.....	2
3. Task	2
3.1. Menyiapkan data di ViewModel	2
3.2. Membuat tampilan list.....	5
3.3. Menangani keadaan jika list kosong	8
3.4. Membuat item dapat diklik.....	10
3.5. Membuat tombol FAB.....	12
4. Summary	14
5. Self-Reflection.....	14
6. Challenge	15

Modul 07: Scrollable List

1. Overview

Pada modul ini dan beberapa modul ke depan, kita akan membuat sebuah aplikasi untuk menulis catatan sederhana di smartphone. Pengguna dapat membuat catatan baru yang disimpan di database SQLite, mengubah catatan yang ada, atau bahkan menghapusnya.. Membuat aplikasi memang butuh waktu, dan karena kita menggunakan pendekatan **incremental development** dalam pengembangan aplikasi, di modul ini kita hanya akan membuat aplikasinya sampai menampilkan list catatan yang dapat di-scroll secara vertikal (atas/bawah) dulu.

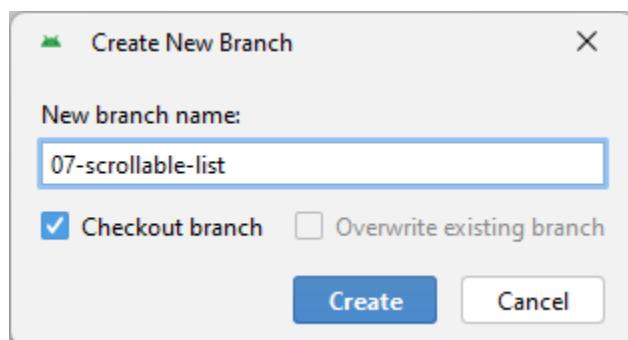
Aplikasi yang akan kita buat di modul ini dapat dilihat kode lengkapnya di repository Github berikut: <https://github.com/indraazimi/mobpro1-compose> pada branch 07-scrollable-list. Gambar di bawah adalah tampilan dari aplikasi yang akan kita buat.



2. Getting Started

Seperti biasa, kita akan melanjutkan praktikum ini menggunakan project praktikum. Silahkan buka project-nya di Android Studio. Lihat pojok kanan bawah dan pastikan kita sedang berada di branch **master**. Jika tidak, maka check-out dulu branch tersebut.

Pastikan ada file MainScreen.kt di dalam package screen. Selanjutnya coba jalankan project master tersebut. Pastikan aplikasi yang muncul adalah layar kosong yang punya Top App Bar di atasnya ya.. Jika aplikasi masih berjalan dengan baik, maka klik menu Git > New Branch..., kemudian masukkan nama branch "07-scrollable-list", lalu klik tombol Create.



Berikutnya, kita akan menambahkan beberapa komponen pada project Android Studio ini. Kerjakan task-task pada modul ini di branch yang baru saja kita buat, langkah demi langkah.. Semoga berhasil!

Dilarang keras untuk copy – paste kode dari modul/sumber lain!

**Ngoding pelan-pelan akan membuat kamu lebih jago di masa depan.
Lakukan commit setiap selesai 1 sub-task. Selamat ngoding!**

3. Task

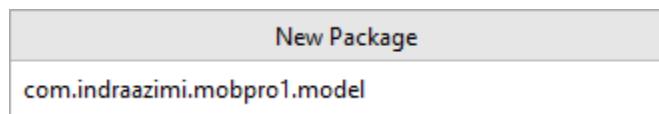
3.1. Menyiapkan data di ViewModel

Pada aplikasi ini, kita akan menampilkan daftar catatan ke dalam tampilan list. Untuk itu, diperlukan datanya terlebih dahulu. Data ini biasanya didapat dari database lokal seperti SQLite, hasil request ke server melalui API, dan lainnya. Akan tetapi karena kita belum belajar mengambil data dari database ataupun dari internet, kali ini kita akan menggunakan data dummy yang kita bentuk dalam list dulu (seperti ArrayList di bahasa pemrograman Java).

Walaupun data kita dummy, data tersebut haruslah mirip dengan data sebenarnya yang akan diinput oleh pengguna aplikasi. **Jika kita analisa**, sebuah catatan biasanya terdiri dari atribut berikut:

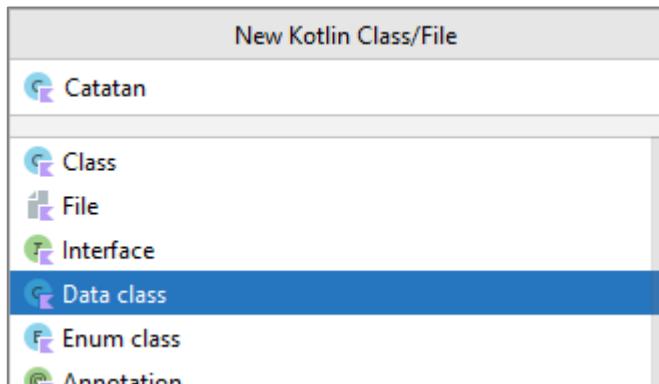
- a. Kapan dicatatnya?
- b. Apa judul catatannya?
- c. Apa isi dari catatan tersebut?

Oleh karena sebuah catatan terdiri dari beberapa atribut, maka kita perlu membuat kelas baru yang akan menjadi model dari data kita. Dan agar project tersusun rapi, masukkan model ini ke dalam package tersendiri. Pada Project Explorer, klik kanan nama package (`org.d3ifxxxx.mobpro1`) kemudian pilih New > Package... Pada dialog yang muncul, tambahkan kata model seperti pada gambar.



Note: gambar di atas hanya untuk ilustrasi. Pastikan nama package baru kalian mengikuti format: `org.d3ifxxxx.mobpro1.model`

Selanjutnya, klik kanan package model di Project Explorer, kemudian pilih New > Kotlin Class/File.. Pilih Data Class, lalu masukkan Catatan, dan tekan Enter.. Jangan lupa tambahkan file ke Git.



Selanjutnya isi kelas Catatan.kt agar memiliki 3 atribut sesuai analisa, ditambah ID catatan.

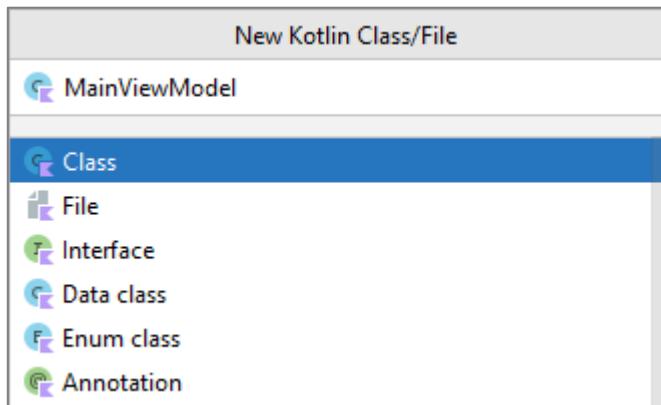
```
data class Catatan(  
    val id: Long,  
    val judul: String,  
    val catatan: String,  
    val tanggal: String  
)
```

Berikutnya, gunakan kelas tersebut untuk membuat list Catatan. Oleh karena catatan kita nantinya akan diambil dari database, maka kita tidak membuat list-nya di MainActivity.kt seperti modul 03 (aplikasi Galeri Hewan). Akan tetapi, kita akan membuat list-nya di kelas khusus yaitu ViewModel. Untuk itu, tambahkan dulu dependency di build.gradle (app)

```
dependencies {  
    ...  
    implementation("androidx.compose.material3:material3")  
    implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.7.0")  
  
    testImplementation("junit:junit:4.13.2")  
    ...  
}
```

Jika sudah, lakukan Gradle sync. Klik Sync Now di kanan atas Code Editor.

Selanjutnya, klik kanan package screen di Project Explorer, kemudian pilih New > Kotlin Class/File.. Pilih Class, lalu masukkan MainViewModel, dan tekan Enter.. Jangan lupa tambahkan file ke Git.



Lakukan extends ViewModel dan siapkan datanya menggunakan loop agar data dummy-nya dinamis.

```
class MainViewModel : ViewModel() {  
  
    val data = getDataDummy()  
  
    private fun getDataDummy(): List<Catatan> {  
        val data = mutableListOf<Catatan>()  
        for (i in 29 downTo 20) {  
  
        }  
        return data  
    }
}
```

Di dalam perulangan bikin catatan yang agak panjang ya agar seperti aslinya.

```
for (i in 29 downTo 20 ) {
    data.add(
        Catatan(
            i.toLong(),
            "Kuliah Mobpro $i Feb",
            "Yey, hari ini belajar membuat aplikasi Android counter dan berhasil. Hehe.. Mudah2an modul selanjutnya juga lancar. Aamiin.",
            "2024-02-$i 12:34:56"
        )
    )
}
```

Sekarang jalankan aplikasi.. Tampilan masih kosong, dan itu wajar. Sebab kita baru menyiapkan data dummy. Belum sampai menampilkannya.. Jika tidak terjadi error, lakukan commit sesuai judul task.

3.2. Membuat tampilan list

Sebuah tampilan list akan terdiri dari item-item sejumlah data yang ada di dalamnya. Jadi sekarang yang perlu kita lakukan pertama adalah membuat list item. **Coba kita analisa..** Sesuai screenshot di Overview, untuk setiap item, data akan disusun secara vertikal. Sehingga jelas kita harus memakai container berupa komponen Column. Di dalamnya, kita akan menggunakan tiga komponen Text untuk menampilkan tiga atribut dari setiap catatan.

Berikut kode tampilan sesuai analisa di atas.. Letakkan kode ini di bawah Composable ScreenContent.

```
@Composable
fun ListItem(catatan: Catatan) {
    Column {
        Text(text = catatan.judul)
        Text(text = catatan.catatan)
        Text(text = catatan.tanggal)
    }
}
```

Selanjutnya panggil data yang akan ditampilkan dengan ListItem. Data ini ada di ViewModel, jadi panggil dulu ViewModel-nya seperti berikut.

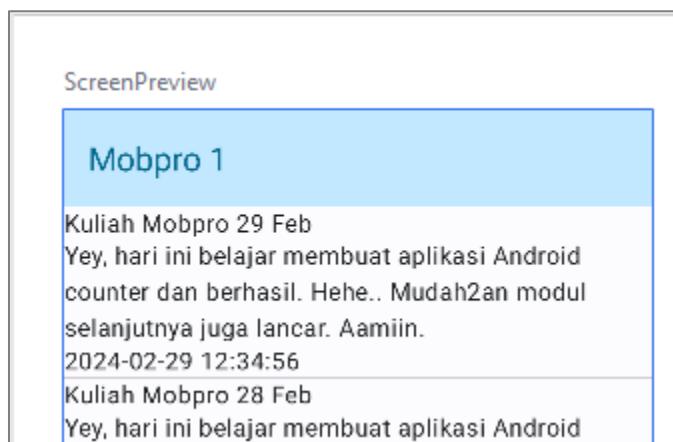
```
@Composable
fun ScreenContent(modifier: Modifier) {
    val viewModel: MainViewModel = viewModel()
    val data = viewModel.data
```

Setelah itu baru gunakan Composable ListItem di dalam Composable ScreenContent. Di sini kita menggunakan komponen LazyColumn, karena tidak semua data akan ditampilkan dalam satu waktu.. Kita juga memakai komponen Divider agar batas tiap item terlihat jelas.

```
@Composable
fun ScreenContent(modifier: Modifier) {
    val viewModel: MainViewModel = viewModel()
    val data = viewModel.data

    Column(
        LazyColumn(
            modifier = modifier.fillMaxSize().padding(16.dp)) {
            modifier = modifier.fillMaxSize()
        ) {
            items(data) {
                ListItem(catatan = it)
                Divider()
            }
        }
    )
}
```

Sekarang tampilan preview dari list sudah terlihat.. Tampilannya belum bagus, karena memang belum dipoles. Ketika membuat aplikasi, fokuslah pada fungsionalitas dahulu.



Jika aplikasi kita jalankan, maka akan muncul list catatan yang dapat di-scroll.. Fungsi sudah OK, maka saatnya kita memoles tampilan sesuai aturan umum, diantaranya:

- Harus ada jarak antara suatu komponen dengan pinggir layar atau komponen lain di sekelilingnya.
- Tampilan aplikasi harus tetap konsisten, bagaimanapun data yang diinput pengguna.
- Data yang penting harus diberi penekanan lebih (karena dia penting).

Aturan #1: Harus ada jarak antara suatu komponen dengan pinggir layar atau komponen lain di sekelilingnya.. Untuk memenuhi aturan ini, berikan padding kepada container dari list item, yaitu Column. Berikan pula jarak antar komponen seperti kode berikut.

```
@Composable
fun ListItem(catatan: Catatan) {
    Column{
        Column(
            modifier = Modifier.fillMaxWidth().padding(16.dp),
            verticalArrangement = Arrangement.spacedBy(8.dp)
        ) {
            Text(text = catatan.judul)
            Text(text = catatan.catatan)
            Text(text = catatan.tanggal)
        }
    }
}
```

Aturan #2: Tampilan aplikasi harus tetap konsisten, bagaimanapun data yang diinput pengguna.. Pengguna aplikasi kita mungkin akan menuliskan judul dan isi catatan yang panjangnya bervariasi. Untuk memenuhi aturan ini, berikan jumlah baris maksimal yang akan tampil pada komponen Text, sehingga jika judul dan isi catatan ternyata panjang, list item tidak melar terlalu besar. Berikan pula indikator ellipsis jika teks yang muncul di list item terpotong.

```
@Composable
fun ListItem(catatan: Catatan) {
    Column(
        modifier = Modifier.fillMaxWidth().padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        Text(
            text = catatan.judul,
            maxLines = 1,
            overflow = TextOverflow.Ellipsis
        )
        Text(
            text = catatan.catatan,
            maxLines = 2,
            overflow = TextOverflow.Ellipsis
        )
        Text(text = catatan.tanggal)
    }
}
```

Bagaimana tampilan aplikasi kita sekarang? Lebih baik bukan? 😊

Aturan #3: Data yang penting harus diberi penekanan lebih (karena dia penting).. Dari tiga atribut catatan, atribut manakah yang paling penting? Tentunya judul, karena dia pendek dan dapat memberi tahu pengguna secara cepat isi catatan di dalamnya. Maka, untuk memenuhi aturan ini, buatlah agar komponen Text yang berisi atribut judul menjadi tebal.

```
@Composable
fun ListItem(catatan: Catatan) {
    Column(
        modifier = Modifier.fillMaxWidth().padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        Text(
            text = catatan.judul,
            maxLines = 1,
            overflow = TextOverflow.Ellipsis,
            fontWeight = FontWeight.Bold
        )
        Text(
            text = catatan.catatan,
            maxLines = 2,
            overflow = TextOverflow.Ellipsis
        )
        Text(text = catatan.tanggal)
    }
}
```

Sebenarnya masih banyak aturan lain yang harus diikuti ketika membuat tampilan aplikasi. Namun, karena mata kuliah kita fokusnya di pemrograman, silahkan explore sendiri ya!

Sekarang jalankan aplikasi.. Jika sudah OK, lakukan commit sesuai judul task.

3.3. Menangani keadaan jika list kosong

Ketika aplikasi catatan kita dipakai pertama kali oleh pengguna, tentunya belum ada catatan apapun di dalamnya. Misalkan ada pengguna yang baru meng-install aplikasi kita, kemudian melihat aplikasi ini kosong tanpa ada isi apapun, maka itu dapat memberi kesan yang kurang baik. Sebagai developer Android yang baik, kita harus menangani hal ini..

Terdapat [beberapa cara](#) untuk menangani hal ini. Dan di modul sekarang, kita akan menggunakan cara yang paling sederhana, yaitu membuat "empty state". Yaitu sebuah pesan yang muncul untuk memberi tahu pengguna bahwa list-nya kosong.

Sebagai langkah awal, buatlah percabangan agar tampilan ketika data kosong dan data ada berbeda.

```
@Composable
fun ScreenContent(modifier: Modifier) {
    val viewModel: MainViewModel = viewModel()
    val data = viewModel.data

    if (data.isEmpty()) {

    }
    else {
        LazyColumn(
            modifier = modifier.fillMaxSize()
        ) {
            items(data) {
                ListItem(catatan = it)
                Divider()
            }
        }
    }
}
```

Berikutnya, buatlah pesan yang akan ditampilkan di strings.xml.

```
<resources>
    <string name="app_name">Mobpro 1</string>
    <string name="list_kosong">Belum ada data</string>
</resources>
```

Jika sudah, buat komponen Column sebesar layar, berikan jarak ke pinggir dan buat komponen Text agar berada di tengah-tengah layar seperti contoh berikut ini.

```
if (data.isEmpty()) {
    Column(
        modifier = modifier.fillMaxSize().padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = stringResource(id = R.string.list_kosong))
    }
}
```

Sekarang jalankan aplikasi kita. Apakah empty state nya muncul?

..

Jawabannya tentu saja tidak. Haha.. Data kita harus kosong, baru empty state nya muncul. Untuk itu, ganti isi variable data menjadi empty list seperti berikut.

```
@Composable
fun ScreenContent(modifier: Modifier) {
    val viewModel: MainViewModel = viewModel()
    val data = viewModel.data
    val data = emptyList<Catatan>()
```

Sekarang jalankan aplikasi lagi. Apakah empty state nya muncul?

Jika empty state sudah muncul, maka kembalikan lagi isi variable data ke keadaan awal (berisi data dari ViewModel). Setelah itu baru lakukan commit sesuai judul dari task.

3.4. Membuat item dapat diklik

Di task ini, kita akan membuat agar item yang ada pada list dapat diklik dan menampilkan pesan ke pengguna menggunakan Toast. Sebagai langkah awal, tambahkan parameter baru pada Composable ListItem yang akan menangani event saat list item diklik oleh pengguna.

```
@Composable
fun ListItem(catatan: Catatan) {
fun ListItem(catatan: Catatan, onClick: () -> Unit) {
    Column(
```

Composable ListItem sekarang memiliki dua parameter, yaitu data yang digunakan dan event yang ditangani. Sadarkah kalian sebenarnya kita sedang melakukan apa..?

Ini namanya state hoisting dan kita sudah melakukannya sejak modul 03.. Sekarang gunakan onClick tadi pada modifier Column agar list item dapat diklik seperti berikut.

```
@Composable
fun ListItem(catatan: Catatan, onClick: () -> Unit) {
    Column(
        modifier = Modifier.fillMaxWidth()
            .clickable { onClick() }
            .padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    )
```

Selanjutnya, tambahkan pesan yang akan dimunculkan ke pengguna.

```
<resources>
    <string name="app_name">Mobpro 1</string>
    <string name="list_kosong">Belum ada data</string>
    <string name="x_diklik">%1$s diklik!</string>
</resources>
```

Terakhir, tambahkan context dan pesan Toast berikut di ScreenContent.

```
@Composable
fun ScreenContent(modifier: Modifier) {
    val viewModel: MainViewModel = viewModel()
    val data = viewModel.data
    val context = LocalContext.current

    if (data.isEmpty()) {
        ...
    } else {
        LazyColumn(...) {
            items(data) {
                ListItem(catatan = it) {
                    val pesan = context.getString(R.string.x_diklik, it.judul)
                    Toast.makeText(context, pesan, Toast.LENGTH_SHORT).show()
                }
                Divider()
            }
        }
    }
}
```

Sekarang jalankan aplikasi. Klik salah satu item, maka akan muncul efek ripple berwarna abu pada item tersebut dan muncul pesan Toast. Bagaimana? Keren bukan? 😊

Dengan menggunakan Jetpack Compose, membuat list yang dapat di-scroll sekarang menjadi lebih sederhana. Kita tidak perlu membuat adapter lagi seperti zaman XML.

Jika semua sudah OK, silahkan lakukan commit sesuai judul task ini.

Consistent incremental development leads to exponential growth..

Richard Riche

3.5. Membuat tombol FAB

Di task ini, kita akan membuat Floating Action Button atau biasa disingkat FAB. Tombol ini berada di kanan bawah layar dan biasanya digunakan untuk aksi utama aplikasi. Di aplikasi kita, FAB nantinya akan digunakan untuk menulis catatan baru.. Tambahkan komponen FAB di Scaffold seperti berikut..

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MainScreen() {
    Scaffold(
        topBar = {
            TopAppBar(
                title = ...
                colors = ...
            )
        },
        floatingActionButton = {
            FloatingActionButton(
                onClick = {}
            )
        }
    ) { padding ->
        ScreenContent(Modifier.padding(padding))
    }
}
```

FAB sendiri hanya berisi ikon, sehingga wajib kita beri deskripsi untuk membantu orang yang punya keterbatasan penglihatan. Tulis deskripsi ini di strings.xml

```
<resources>
    ...
    <string name="x_diklik">%1$s diklik!</string>
    <string name="tambah_catatan">Tambah catatan</string>
</resources>
```

Di modul ini FAB belum bisa menambah data baru.. Jadi siapkan pesan agar pengguna tidak bingung.

```
<resources>
    ...
    <string name="x_diklik">%1$s diklik!</string>
    <string name="tambah_catatan">Tambah catatan</string>
    <string name="tambah_error">Belum bisa tambah catatan.</string>
</resources>
```

Selanjutnya tambahkan ikon Add ke dalam FAB seperti ini.

```
FloatingActionButton(  
    onClick = {}  
) {  
    Icon(  
        imageVector = Icons.Filled.Add,  
        contentDescription = stringResource(R.string.tambah_catatan),  
        tint = MaterialTheme.colorScheme.primary  
    )  
}
```

Buat variable context agar nanti kita dapat memakai Toast.

```
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun MainScreen() {  
    val context = LocalContext.current  
  
    Scaffold(  
        ...  
    ) {  
        ...  
    }  
}
```

Panggil Toast dari onClick FAB dan tampilkan pesannya.

```
FloatingActionButton(  
    onClick = {  
        Toast.makeText(context, R.string.belum_bisa, Toast.LENGTH_SHORT).show()  
    }  
) {  
    Icon(...)  
}
```

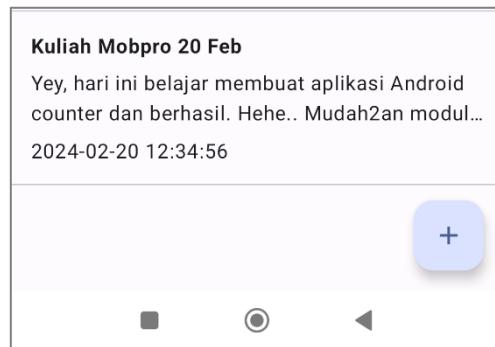
Sekarang jalankan aplikasi. Klik FAB, apakah pesannya muncul..?

Jika ya, coba scroll hingga data terakhir.. Apakah kamu sadar ada masalah?

Ternyata data terakhir tertutup oleh FAB sehingga tidak dapat dibaca. Untuk mengatasi masalah ini, tambahkan content padding bottom ke komponen list kita.

```
LazyColumn(  
    modifier = modifier.fillMaxSize(),  
    contentPadding = PaddingValues(bottom = 84.dp)  
) {  
    items(data) {  
        ...  
    }  
}
```

Jalankan aplikasi lagi dan scroll hingga data terakhir. Pastikan data terakhir sudah tidak terhalangi FAB dan tampilan aplikasi terlihat seperti pada gambar berikut ini.



Jika aplikasi sudah berjalan dengan baik, lakukan commit sesuai judul task.. Selanjutnya cek daftar commit di tab Git. Pastikan jumlah commit yang dilakukan sudah sesuai jumlah task.

Berikutnya, lakukan push kode project ke repo Github menggunakan menu Git > Push.

4. Summary

Pada modul kali ini kita telah membuat sebuah aplikasi Android yang menampilkan daftar catatan dalam bentuk list menggunakan komponen LazyColumn. Kita menangani keadaan jika data yang akan ditampilkan ternyata kosong. Kita juga telah belajar bagaimana membuat item pada list bisa diklik dan aplikasi memberi respon menggunakan Toast.. Terakhir, kita belajar bagaimana menambahkan FAB untuk memudahkan pengguna kita melakukan aksi utama di aplikasi.

5. Self-Reflection

Apakah praktik kita di modul ini telah berhasil..? Silahkan isi tabel berikut.

No.	Pertanyaan	Ya / Tidak
1	Saya dapat membuat aplikasi berbentuk list	
2	Saya dapat menangani keadaan jika list kosong	
3	Saya dapat membuat item pada list bisa diklik	
4	Saya dapat membuat Floating Action Button (FAB)	

6. Challenge

Buatlah branch baru dengan nama “07-challenge”. Selanjutnya, ubah project praktikum kalian agar menjadi aplikasi yang menampilkan list berisi data mahasiswa. Tampilan list item-nya bebas, boleh seperti contoh berikut, atau yang lainnya sesuai kreativitas kalian. Yang penting di setiap list item ada informasi **nama, NIM dan kelas** mahasiswanya dan memenuhi aturan design yang dibahas di modul.

Aplikasi WAJIB memenuhi kriteria berikut:

- a. Bisa di-scroll, jadi tambahkan data dummy yang banyak.
- b. Terdapat empty state jika data mahasiswanya kosong.
- c. Ketika data mahasiswa diklik, muncul Toast berisi nama.
- d. Terdapat FAB yang ketika diklik, akan muncul Toast juga.
- e. FAB tidak menutupi item terakhir karena diberi padding.

Note: Pastikan kode kalian rapi, bersih, tanpa warning dan tidak ada sisa-sisa aplikasi Catatan.

Selamat mengerjakan 😊

Mobpro 1		
Rizza Indah Mega Mandasari	6706244601	D3IF-46-01
Indra Azimi	6706244602	D3IF-46-02
Reza Budiawan	6706244612	D3IF-46-02
Dwiko Indrawansyah	6706244622	D3IF-46-02
Cahyana	6706244603	D3IF-46-03
Indra Azimi	6706244604	D3IF-46-04
Erna Hikmawati	6706244605	D3IF-46-05

+