Navigasi Antar Layar dengan Compose

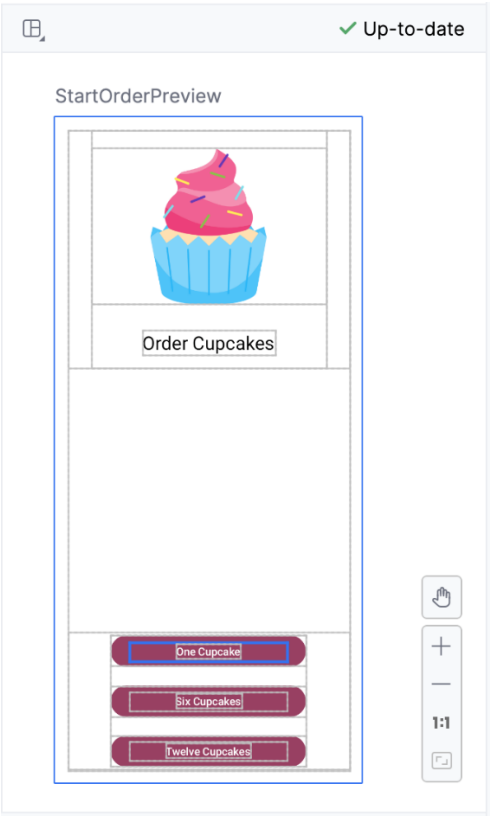
Unduhh kode awal di:

https://github.com/google-developer-training/basic-android-kotlin-compose-training-cupcake/archive/refs/heads/starter.zip

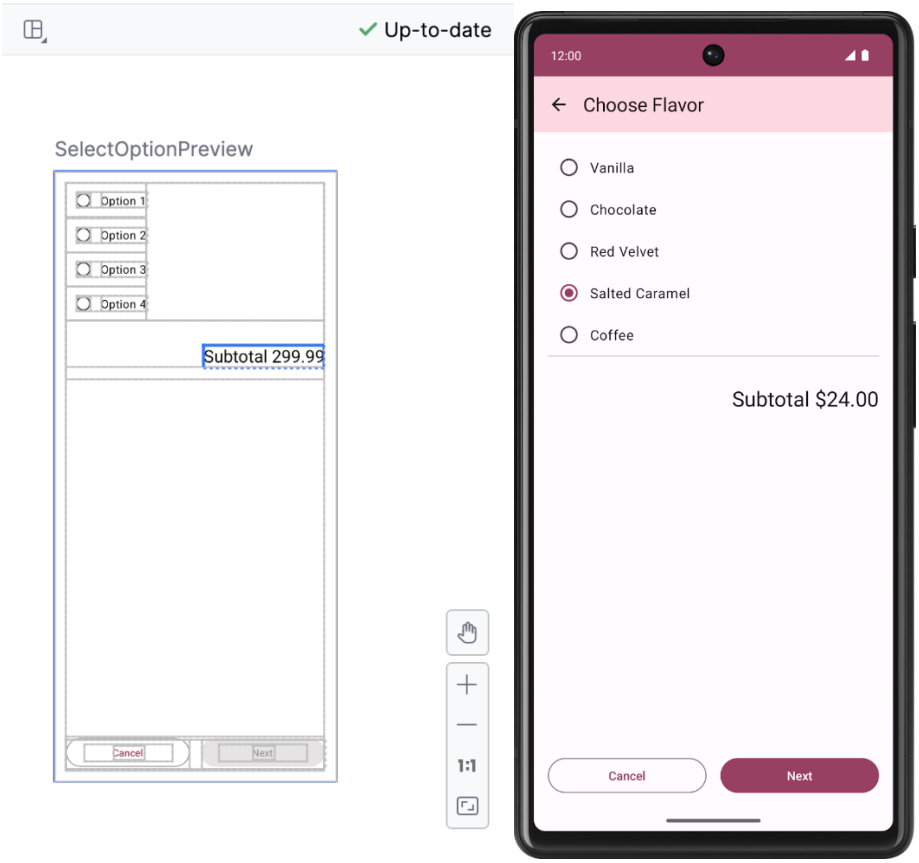
Aplikasi Cupcake

Layar untuk memulai pesanan

Dalam kode, layar ini diwakili oleh composable `StartOrderScreen` di `StartOrderScreen.kt`



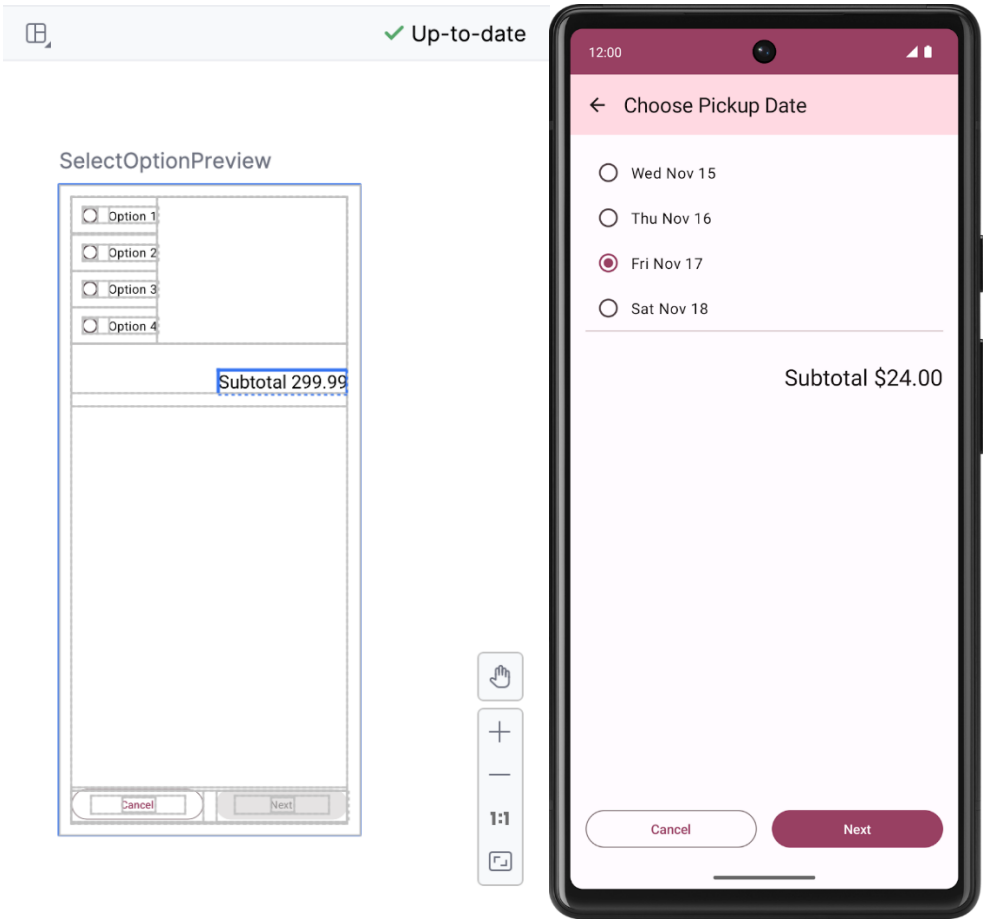
Layar untuk memilih rasa



Daftar kemungkinan rasa disimpan sebagai daftar ID resource string di `data.DataSource.kt`

Layar untuk memilih tanggal pengambilan

Setelah memilih rasa, aplikasi menampilkan serangkaian tombol pilihan lain kepada pengguna untuk memilih tanggal pengambilan. Opsi pengambilan berasal dari daftar yang ditampilkan oleh fungsi `pickupOptions()` di `OrderViewModel`.



Layar ini diimplementasikan oleh composable `OrderSummaryScreen` di `SummaryScreen.kt`.

Menentukan rute dan membuat NavHostController

Bagian dari Komponen Navigasi

Komponen Navigasi memiliki tiga bagian utama:

- **NavController:** Bertanggung jawab untuk menavigasi di antara tujuan—yaitu layar di aplikasi Anda.
- **NavGraph:** Memetakan tujuan composable untuk dinavigasi.
- **NavHost:** Composable yang bertindak sebagai container untuk menampilkan tujuan NavGraph saat ini.

Anda akan mulai dengan menentukan empat rute aplikasi Cupcake.

- **`Start`:** Pilih jumlah cupcake dari salah satu dari tiga tombol.
- **`Flavor`:** Pilih rasa dari daftar pilihan.
- **`Pickup`:** Pilih tanggal pengambilan dari daftar pilihan.
- **`Summary`:** Tinjau pilihan, lalu kirim atau batalkan pesanan.

Tambahkan class enum untuk menentukan rute.

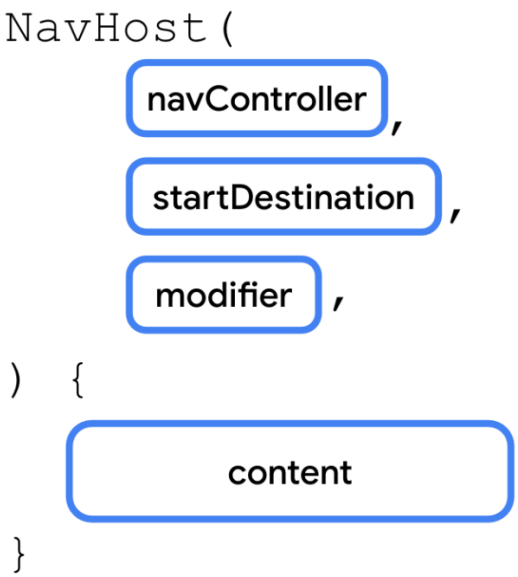
1. Di `CupcakeScreen.kt`, di atas composable `CupcakeAppBar`, tambahkan class enum bernama `CupcakeScreen`.

```
enum class CupcakeScreen() {
    Start,
    Flavor,
    Pickup,
    Summary
}
```

Menambahkan NavHost ke aplikasi Anda

NavHost adalah Composable yang menampilkan tujuan composable lainnya, berdasarkan rute tertentu. Misalnya, jika rutenya adalah `Flavor`, `NavHost` akan menampilkan layar untuk memilih rasa cupcake. Jika rutenya adalah `Summary`, aplikasi akan menampilkan layar ringkasan.

Sintaksis untuk `NavHost` sama seperti Composable lainnya.



Ada dua parameter penting.

- **`navController`:** Instance dari class `NavHostController`. Anda dapat menggunakan objek ini untuk berpindah antarlayar, misalnya, dengan memanggil metode `navigate()` untuk menuju ke tujuan lain. Anda dapat memperoleh `NavHostController` dengan memanggil `rememberNavController()` dari fungsi composable.

- **`startDestination`:** Rute string yang menentukan tujuan yang ditampilkan secara default saat aplikasi pertama kali menampilkan `NavHost`. Untuk aplikasi Cupcake, seharusnya ini adalah rute `Start`.

1. Buka `CupcakeScreen.kt`.
2. Dalam `Scaffold`, di bawah variabel `uiState`, tambahkan composable `NavHost`.

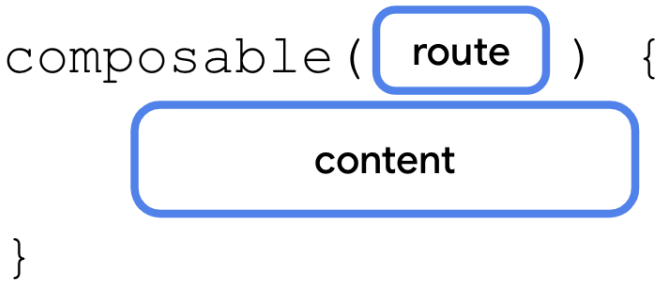
```
import androidx.navigation.compose.NavHost
import androidx.compose.foundation.layout.padding

NavHost(
    navController = navController,
    startDestination = CupcakeScreen.Start.name,
    modifier = Modifier.padding(innerPadding)
) {

}
```

Menangani rute di NavHost Anda

Seperti composable lainnya, `NavHost` menggunakan tipe fungsi untuk kontennya.



Dalam fungsi konten `NavHost`, Anda memanggil fungsi `composable()`. Fungsi `composable()` memerlukan dua parameter.

- **`route`:** String yang sesuai dengan nama rute. Ini dapat berupa string unik apa pun. Anda akan menggunakan properti nama konstanta enum `CupcakeScreen`.
- **`content`:** Di sini Anda dapat memanggil composable yang ingin ditampilkan untuk rute yang diberikan.

Anda akan memanggil fungsi `composable()` satu kali untuk masing-masing dari keempat rute.

1. Panggil fungsi `composable()`, dengan meneruskan `CupcakeScreen.Start.name` untuk `route`.

```
import androidx.navigation.compose.composable
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.ui.res.dimensionResource
```

```
import com.example.cupcake.ui.StartOrderScreen
import com.example.cupcake.data.DataSource

NavHost(
    navController = navController,
    startDestination = CupcakeScreen.Start.name,
    modifier = Modifier.padding(innerPadding)
) {
    composable(route = CupcakeScreen.Start.name) {
        StartOrderScreen(
            quantityOptions = DataSource.quantityOptions,
            modifier = Modifier
                .fillMaxSize()
                .padding(dimensionResource(R.dimen.padding_medium))
        )
    }
}
```

Sampai di sini jalankan aplikasi. Seharus tampil layar StartOrderScreen.

2. Di bawah panggilan pertama ke `composable()`, panggil `composable()` lagi, dengan meneruskan `CupcakeScreen.Flavor.name` untuk route.

```
import androidx.compose.ui.platform.LocalContext
import com.example.cupcake.ui.SelectOptionScreen
import androidx.compose.foundation.layout.fillMaxHeight

composable(route = CupcakeScreen.Flavor.name) {
    val context = LocalContext.current
    SelectOptionScreen(
        subtotal = uiState.price,
        options = DataSource.flavors.map { id ->
context.resources.getString(id) },
        onSelectionChanged = { viewModel.setFlavor(it) },
        modifier = Modifier.fillMaxHeight()
    )
}
```

3. Panggil lagi fungsi `composable()` dengan meneruskan `CupcakeScreen.Pickup.name` untuk parameter `route`.

```
composable(route = CupcakeScreen.Pickup.name) {
    SelectOptionScreen(
        subtotal = uiState.price,
        options = uiState.pickupOptions,
        onSelectionChanged = { viewModel.setDate(it) },
        modifier = Modifier.fillMaxHeight()
    )
}
```

4. Panggil `composable()` sekali lagi, dengan meneruskan `CupcakeScreen.Summary.name` untuk `route`.

```
import com.example.cupcake.ui.OrderSummaryScreen

composable(route = CupcakeScreen.Summary.name) {
    OrderSummaryScreen(
        orderUiState = uiState,
        modifier = Modifier.fillMaxHeight()
    )
}
```

Seperti itulah cara menyiapkan `NavHost`.

Menambahkan pengendali tombol ke `StartOrderScreen`

Anda akan memulai dengan menambahkan parameter jenis fungsi yang dipanggil saat salah satu tombol kuantitas ditekan di layar pertama. Fungsi ini diteruskan ke dalam composable `StartOrderScreen` dan bertanggung jawab untuk memperbarui model tampilan dan membuka layar berikutnya.

1. Buka `StartOrderScreen.kt`.
2. Di bawah parameter `quantityOptions`, dan sebelum parameter `modifier`, tambahkan parameter bernama `onNextButtonClicked` dari tipe `() -> Unit`.

```
@Composable
fun StartOrderScreen(
    quantityOptions: List<Pair<Int, Int>>,
    onNextButtonClicked: () -> Unit,
    modifier: Modifier = Modifier
){
    ...
}
```

3. Setelah composable `StartOrderScreen` mendapatkan nilai untuk `onNextButtonClicked`, cari `StartOrderPreview` lalu teruskan isi lambda kosong ke parameter `onNextButtonClicked`.

```
@Preview
@Composable
fun StartOrderPreview() {
    CupcakeTheme {
        StartOrderScreen(
            quantityOptions = DataSource.quantityOptions,
            onNextButtonClicked = {},
            modifier = Modifier
                .fillMaxSize()
                .padding(dimensionResource(R.dimen.padding_medium))
        )
```

```
    }
}
```

4. Ubah tipe parameter `onNextButtonClicked` untuk mengambil parameter `Int`.

```
onNextButtonClicked: (Int) -> Unit,
```

5. Temukan ekspresi lambda kosong untuk parameter `onClick` dari `SelectQuantityButton`. Dalam ekspresi lambda, panggil `onNextButtonClicked` dengan meneruskan `item.second`, yaitu jumlah cupcake.

```
quantityOptions.forEach { item ->
    SelectQuantityButton(
        labelResourceId = item.first,
        onClick = { onNextButtonClicked(item.second) }
    )
}
```

Menambahkan pengendali tombol ke SelectOptionScreen

1. Di bawah parameter `onSelectionChanged` composable `SelectOptionScreen` di `SelectOptionScreen.kt`, tambahkan parameter bernama `onCancelButtonClicked` dari tipe `() -> Unit` dengan nilai default `{}`.

```
@Composable
fun SelectOptionScreen(
    subtotal: String,
    options: List<String>,
    onSelectionChanged: (String) -> Unit = {},
    onCancelButtonClicked: () -> Unit = {},
    modifier: Modifier = Modifier
)
```

2. Di bawah parameter `onCancelButtonClicked`, tambahkan parameter lain dari tipe `() -> Unit` bernama `onNextButtonClicked` dengan nilai default `{}`.

```
@Composable
fun SelectOptionScreen(
    subtotal: String,
    options: List<String>,
    onSelectionChanged: (String) -> Unit = {},
    onCancelButtonClicked: () -> Unit = {},
    onNextButtonClicked: () -> Unit = {},
    modifier: Modifier = Modifier
)
```

3. Teruskan `onCancelButtonClicked` untuk parameter `onClick` tombol **Cancel**.

```
OutlinedButton(
    modifier = Modifier.weight(1f),
    onClick = onCancelButtonClicked
) {
    Text(stringResource(R.string.cancel))
}
```

4. Teruskan `onNextButtonClicked` untuk parameter `onClick` tombol Next.

```
Button(
    modifier = Modifier.weight(1f),
    enabled = selectedValue.isNotEmpty(),
    onClick = onNextButtonClicked
) {
    Text(stringResource(R.string.next))
}
```

Menambahkan pengendali tombol ke SummaryScreen

Terakhir, tambahkan fungsi pengendali tombol untuk tombol **Cancel** dan **Send** pada layar ringkasan.

1. Pada composable `OrderSummaryScreen` di `SummaryScreen.kt`, tambahkan parameter bernama `onCancelButtonClicked` dari tipe `() -> Unit`.

```
@Composable
fun OrderSummaryScreen(
    orderUiState: OrderUiState,
    onCancelButtonClicked: () -> Unit,
    modifier: Modifier = Modifier
){
    ...
}
```

2. Tambahkan parameter lain dari jenis `(String, String) -> Unit` lalu beri nama `onSendButtonClicked`.

```
@Composable
fun OrderSummaryScreen(
    orderUiState: OrderUiState,
    onCancelButtonClicked: () -> Unit,
    onSendButtonClicked: (String, String) -> Unit,
    modifier: Modifier = Modifier
){
    ...
}
```

3. Sekarang composable `OrderSummaryScreen` mendapatkan nilai untuk `onSendButtonClicked` dan `onCancelButtonClicked`. Cari `OrderSummaryPreview`, teruskan isi lambda kosong dengan dua parameter `String` ke `onSendButtonClicked` dan isi lambda kosong ke parameter `onCancelButtonClicked`.

```
@Preview
@Composable
fun OrderSummaryPreview() {
    CupcakeTheme {
        OrderSummaryScreen(
            orderUiState = OrderUiState(0, "Test", "Test", "$300.00"),
            onSendButtonClicked = { subject: String, summary: String -> },
            onCancelButtonClicked = {},
            modifier = Modifier.fillMaxHeight()
        )
    }
}
```

4. Teruskan `onSendButtonClicked` untuk parameter `onClick` dari tombol **Send**. Teruskan `newOrder` dan `orderSummary`, dua variabel yang ditentukan sebelumnya di `OrderSummaryScreen`. String ini terdiri dari data aktual yang dapat dibagikan pengguna kepada aplikasi lain.

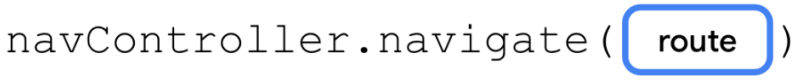
```
Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = { onSendButtonClicked(newOrder, orderSummary) }
) {
    Text(stringResource(R.string.send))
}
```

5. Teruskan `onCancelButtonClicked` untuk parameter `onClick` pada tombol **Cancel**.

```
OutlinedButton(
    modifier = Modifier.fillMaxWidth(),
    onClick = onCancelButtonClicked
) {
    Text(stringResource(R.string.cancel))
}
```

Membuka rute lain

Untuk membuka rute lain, cukup panggil metode `navigate()` pada instance `NavHostController` Anda.



Metode navigasi mengambil satu parameter: `String` yang sesuai dengan rute yang ditentukan di `NavHost`. Jika rute cocok dengan salah satu panggilan ke `composable()` di `NavHost`, aplikasi akan membuka layar tersebut.

Anda akan meneruskan fungsi yang memanggil `navigate()` saat pengguna menekan tombol pada layar `Start`, `Flavor`, dan `Pickup`.

1. Di `CupcakeScreen.kt`, temukan panggilan ke `composable()` untuk layar mulai. Untuk parameter `onNextButtonClicked`, teruskan ekspresi lambda.

```
StartOrderScreen(
    quantityOptions = DataSource.quantityOptions,
    onNextButtonClicked = {
        viewModel.setQuantity(it)
    },
)
```

2. Panggil `navigate()` pada `navController`, yang meneruskan `CupcakeScreen.Flavor.name` untuk `route`.

```
onNextButtonClicked = {
    viewModel.setQuantity(it)
    navController.navigate(CupcakeScreen.Flavor.name)
}
```

3. Untuk parameter `onNextButtonClicked` di layar rasa, cukup teruskan lambda yang memanggil `navigate()`, dengan meneruskan `CupcakeScreen.Pickup.name` untuk `route`.

```
composable(route = CupcakeScreen.Flavor.name) {
    val context = LocalContext.current
    SelectOptionScreen(
        subtotal = uiState.price,
        onNextButtonClicked = { navController.navigate(CupcakeScreen.Pickup.name)
},
        options = DataSource.flavors.map { id -> context.resources.getString(id)
},
        onSelectionChanged = { viewModel.setFlavor(it) },
        modifier = Modifier.fillMaxHeight()
    )
}
```

4. Teruskan lambda kosong untuk `onCancelButtonClicked` yang akan Anda implementasikan berikutnya.

```
SelectOptionScreen(
    subtotal = uiState.price,
    onNextButtonClicked = { navController.navigate(CupcakeScreen.Pickup.name) },
    onCancelButtonClicked = {},
    options = DataSource.flavors.map { id -> context.resources.getString(id) },
    onSelectionChanged = { viewModel.setFlavor(it) },
```

```
    modifier = Modifier.fillMaxHeight()
)
```

5. Untuk parameter `onNextButtonClicked` di layar pengambilan, teruskan lambda yang memanggil `navigate()`, dengan meneruskan `CupcakeScreen.Summary.name` untuk `route`.

```
composable(route = CupcakeScreen.Pickup.name) {
    SelectOptionScreen(
        subtotal = uiState.price,
        onNextButtonClicked = { navController.navigate(CupcakeScreen.Summary.name)
},
        options = uiState.pickupOptions,
        onSelectionChanged = { viewModel.setDate(it) },
        modifier = Modifier.fillMaxHeight()
    )
}
```

6. Sekali lagi, teruskan lambda kosong untuk `onCancelButtonClicked()`.

```
SelectOptionScreen(
    subtotal = uiState.price,
    onNextButtonClicked = { navController.navigate(CupcakeScreen.Summary.name) },
    onCancelButtonClicked = {},
    options = uiState.pickupOptions,
    onSelectionChanged = { viewModel.setDate(it) },
    modifier = Modifier.fillMaxHeight()
)
```

7. Untuk `OrderSummaryScreen`, teruskan lambda kosong untuk `onCancelButtonClicked` dan `onSendButtonClicked`. Tambahkan parameter untuk `subject` dan `summary` yang diteruskan ke `onSendButtonClicked`, yang akan segera Anda implementasikan.

```
composable(route = CupcakeScreen.Summary.name) {
    OrderSummaryScreen(
        orderUiState = uiState,
        onCancelButtonClicked = {},
        onSendButtonClicked = { subject: String, summary: String ->

        },
        modifier = Modifier.fillMaxHeight()
    )
}
```

Sekarang Anda dapat membuka setiap layar aplikasi. Perhatikan bahwa dengan memanggil `navigate()`, layar tidak hanya berubah, tetapi juga ditempatkan di atas data sebelumnya. Selain itu, saat menekan tombol kembali sistem, Anda dapat kembali ke layar sebelumnya.

sumber:

https://developer.android.com/courses/pathways/android-basics-compose-unit-4-pathway-2?hl=id

Codelab nomor 2