11- NAVIGATION

L'insieme delle informazioni contenute in un'applicazione, è contenuta in pagine diverse che hanno specifiche iterazioni utente.

- Users can move among these screens, according to a given set of routes, supported both by specific widgets in each screen as well as by other general purpose navigation items scaffolding the application, like the drawer menu or the top and bottom bars

Navigation fornisce **transizioni negli spostamenti** → importante per la percezione dell'utente.

```
dependencies {
  val navVersion = "2.7.6"

  // Kotlin standard views
  implementation("androidx.navigation:navigation-fragment-ktx:$navVersion")
  implementation("androidx.navigation:navigation-ui-ktx:$navVersion")

  // Jetpack Compose Integration
  implementation("androidx.navigation:navigation-compose:$navVersion")
}
```

Ci sono 3 idee **basi dietro navigation**:

- **NavController**: espone i metodi per switchare da una parte all'altra
- **NavHost**: composable, frame nei quali inserisco le mie pagine
- **navigation graph** → rappresenta come ci muoviamo da una pagina all'altra

Definire le route

Si può definire che ci sono delle parti che cambiano (body) e delle parti che non cambiano (topAppBar)
Inoltre c'è la **dialagBox** che serve ritornare le informazioni che l'utente inserisce
E activity → in alcuni casi bisogna buttare giù un'attività e mettere sopra un'altra attività corrente
Ogni **route consiste in un path separati da /**

```
@Composable
fun SomeApp() {
  val navController = rememberNavController()
  NavHost(navController = navController, startDestination = "home") {
    composable("home") {
      HomeScreen( /* params */ )
    }
    dialog("addProject") {
      AddProjectDialog( /* params */ )
    }
    composable("editProject/{projectId}",
               arguments = listOf( navArgument("projectId"){type=NavType.IntType}
    ) { entry ->
      EditProjectScreen( projectId = entry.arguments?.getInt("projectId"), /…/)
    }
  }
}
```

→ mostra home

→ quando vado in addProject, mostra sopra un composable con addProjectDialaog

→ in questo caso un integer

Se la nostra applicazione ha 20 screen → ho 20 cases

NAVIGARE TRA VARI SCREENS:

- **NavController** offre un insieme di metodi per tornare allo Screen precedente
  - Spesso conveniete non fare accesso diretto ma incapsulare le azioni di navigazione in funzioni
  - Funzioni propagate come proprietà ai figli dei composable

Incapsulando l'insieme delle possibile azioni che un'utente può fare in un ActionObject basato su un NavHostController.

Il navController gestisce un **BackStack** → struttura dati che ricorda la storia dei vari screen che sono stati navigati.

Si può fare **popBackStack()**

- specificando di andare indietro di 1
- specificando di andsre indietro fino ad Home (in modo tale da non fare una history lunghissima)
- si può anche fare popOffHome che mi butta fuori

```
class Actions(val navCont: NavHostController){
  val addProject: ()->Unit = {
     navCont.navigate("addProject")
  }
  val editProject:(Int)->Unit = { id ->
     navCont.navigate("editProject/${id}")
  }
  val navigateBack: ()->Unit = {
    navCont.popBackStack()
  }
}
@Composable
fun SomeApp() {
  val navController = rememberNavController()
  val actions = remember(navController) {
     Actions(navController)
  } /* … */
```

Gestire stato navigazione:

Ogni entry bnello stack può contenere un'istanza **SavedStateHandle** → key-value map where custom data may be written and retrieved, questi dati persistono anche dopo l'uccisione del processo da parte del sistema e sono disponibili finchè la entry non è speicificatamente rimossa dallo stack.

- Può essere usata per mandare i dati a quello che sta back nello stack

```
Button(onClick = {
   navController.previousBackStackEntry?.saveStateHandle?.set("someKey",someValue)
   navController.popBackStack()
}) { Text("Ok") }
```

Ci sono Widget per modificare dati e quelli per navigare:

Scaffold:

Componente che gestisce diversi sotto-componenti:

- Top App Bar
- Floating Action Button
- Bottom Navigation

```
@Composable
fun Material3Demo() {
    Scaffold(
        topBar = { MyTopBar() },
        floatingActionButtonPosition = FabPosition.End,
        floatingActionButton = {
            FloatingActionButton(onClick = { /*TODO*/ }) {
                Icon(Icons.Outlined.Create, "Create")
            }
        },
        bottomBar = { MyBottomBar() },
        content = { paddingValues ->
           MyContent(paddingValues = paddingValues)
        },
    )
}
```

TOP APP BAR:

- TopAppBar displays navigation, actions, and text at the top of a screen

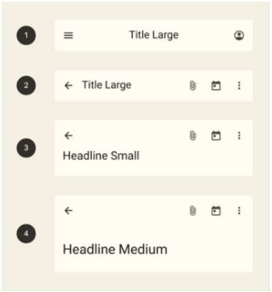
```
@Composable @OptIn(ExperimentalMaterial3Api::class)
fun SimpleTopAppBar() {
    Scaffold(topBar = {
            TopAppBar(
                title = {Text("Material 3")},
                navigationIcon = {
                    IconButton(onClick = {/*doSomething()*/}) {
                        Icon(imageVector = Icons.Filled.Favorite,

            contentDescription = "Favorite") }
                },
                colors = TopAppBarDefaults.smallTopAppBarColors(
                            containerColor = Color.cyan)
            )},
    ) { /* content here */
```

→ navigationICon

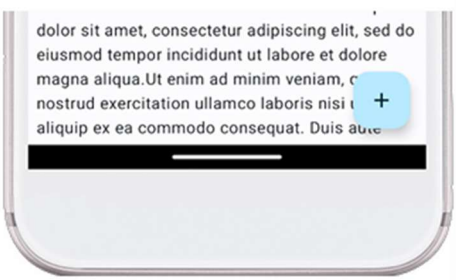
Frequente-important ma non frequent

- There are several versions of the TopAppBar that differ in the disposition of the elements:
  - The `CenterAlignedTopAppBar` has a header title that is horizontally aligned to the center. (1)
  - The `SmallTopAppBar` has a header title that is horizontally aligned to the right and on the same line as icons and actions. (2)
  - The `LargeTopAppBar` and `MediumTopAppBar` display the title in a second row below the navigation and actions when in its default expanded state. The title has more or less line spacing, respectively. (3 , 4)



Nello scaffold si può inserire un **floatingActionButton**, usabile per navigation purpose

```
@Composable
fun ShowFABSample() {
   Scaffold(
            topBar = { /**/ },
       content = { /**/ },
       floatingActionButton = {
           FloatingActionButton(onClick = { /**/ }) {
                  Icon(Icons.Filled.Add, "Add")
           }
       }
   )
}
```

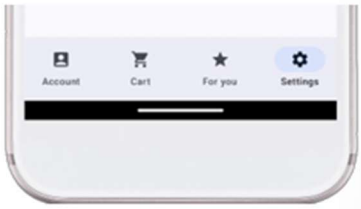


BottomBAr

alcune volte usata per navigation, altre volte per altro: navigation and key actions at the bottom of mobile and tablet screens that are easy to reach on a mobile device

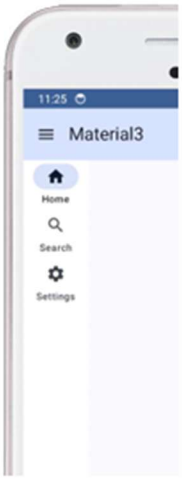
```
@Composable
fun BottomAppBarWithFAB() {
    BottomAppBar (
       actions = {
           IconButton(onClick = {}) {Icon(Icons.Outlined.Search, "Search")}
           IconButton(onClick = {}) {Icon(Icons.Outlined.Delete, "Delete")}
           IconButton(onClick = {}) {Icon(Icons.Outlined.AccountBox, "Account")}
       },
       floatingActionButton = {
           FloatingActionButton(onClick = { /*TODO*/ }) {
               Icon(Icons.Filled.Add, "Add")
           }
       },
   )
}
```


```
@Composable
fun NavigationBarSample() {
    var selectedItem by remember { mutableIntStateOf(0) }
    val items = listOf("Account", "Cart", "For you", "Settings")
    val icons = listOf(Icons.Filled.AccountBox, Icons.Filled.ShoppingCart,
                       Icons.Filled.Star, Icons.Filled.Settings)
    NavigationBar () {
       items.forEachIndexed { index, item ->
           NavigationBarItem(
               selected = index == selectedItem,
               onClick = { selectedItem = index },
               icon = { Icon(icons[index], item) },
               label = { Text(item) } )
       }
    }
}
```



se ci sono tante destinazioni possibili, si può usare ***navigationRails*** per switchare a destinazioni

```
@Composable
fun NavigationRailSample(visible: Boolean) {
    var selectedItem by remember { mutableIntStateOf(0) }
    val items = listOf("Home", "Search", "Settings")
    val icons = listOf(Icons.Filled.Home, Icons.Filled.Search,
                       Icons.Filled.Settings)
    Row() {
        AnimatedVisibility(visible = visible) {
            NavigationRail(containerColor = Color.White) {
                items.forEachIndexed { index, item ->
                    NavigationRailItem(
                        icon = { Icon(icons[index], item) },
                        label = { Text(item) },
                        selected = selectedItem == index,
                        onClick = { selectedItem = index } ) }
}   }    }   }
```



Navigation Drawer → non compare, quando clicchi compare e mostra vari posti in cui puoi andare

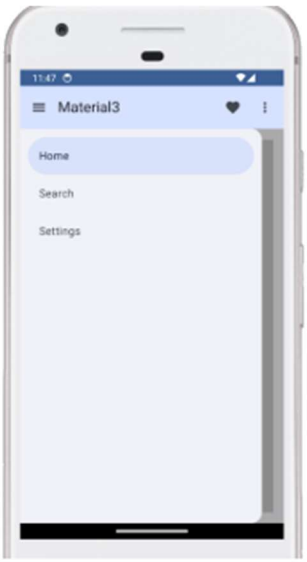
Due tipi:

- **Standarddrawers**, that share space within a screen with other content
- **Modaldrawers**, that appear over the top of other content

Non per forza per navigare, anche per altri scopi. → comune per applicazione.

```
@Composable
fun NavDrawerSample(drawerState: DrawerState, scope: CoroutineScope) {

    var selectedItem by remember { mutableIntStateOf(0) }
    val items = listOf("Home", "Search", "Settings")
    ModalNavigationDrawer(
        drawerState = drawerState,
        drawerContent = {
            ModalDrawerSheet {
                Spacer(modifier = Modifier.height(12.dp))
                items.forEachIndexed { index, item ->
                    NavigationDrawerItem(
                        label = { Text(item) },
                        selected = index==selectedItem ,
                        onClick = { selectedItem = index
                            scope.launch { drawerState.close() } },
                        modifier = Modifier.padding(
                         NavigationDrawerItemDefaults.ItemPadding))
}   } } ) { /* page content goes here */ } }
```

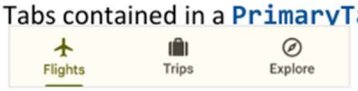


Tabs: per cose non importanti, insieme di icone sul top, ognuna delle quali evidenziata corrispondente al page attuale

Utili per raggruppare in categorie:

- Si dividono in primari e secondari
- Possono scrollare orrizzontalmente

- Primary tabs are typically placed at the top of the content pane under a top app bar
  - They allow to quickly switch between related content destinations
  - Tabs contained in a `PrimaryTabRow` are evenly spaced and fill the entire available space



- Secondary tabs tabs are used within a content area to further separate related content and establish hierarchy

Alcune volte, tab per intera area, altre volte per luna soecifica area.

```
@Composable
@OptIn(ExperimentalMaterial3Api::class)
fun TabDemo(modifier: Modifier) {
    var state by remember { mutableStateOf(0) }
    val icons = listOf(Icons.Filled.Home, Icons.Filled.ShoppingCart, Icons.Filled.AccountCircle)
    val titles = listOf("Home", "Shopping Cart", "Profile")
    Column(modifier = modifier) {
        TabRow(selectedTabIndex = state) {
            titles.forEachIndexed { index, title ->
                Tab(selected = state == index,
                    onClick = { state = index },
                    text = { Text(text = title) },
                    icon = { Icon(icons[index],title) })
            }
        }
        Text(modifier = Modifier.align(Alignment.CenterHorizontally).padding(8.dp),
            text = titles[state],
            style = MaterialTheme.typography.headlineLarge)
    }
}
```