Navigation Component

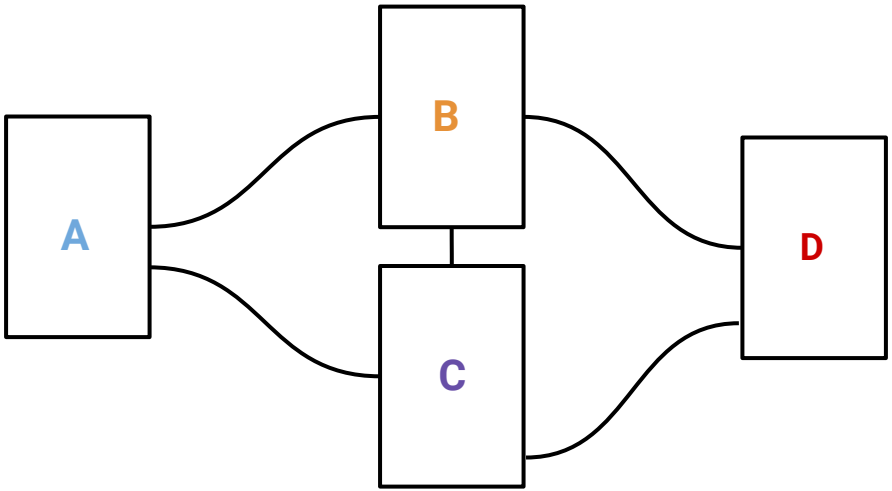
Per iniziare

Navigation Component presentazione google

- collezione di librerie che **simplifica** la navigazione all'interno di un'app nello sviluppo Android
- sostituiscono gli intent/fragment transactions
- progettato per situazioni più complesse
- migliore gestione della complessità
- permette il **passaggio di argomenti** in modo sicuro
- include **animazioni**
- **rappresentazione grafica** del progetto
- nativamente, supporta fragments/activities, ma è possibile creare anche un'implementazione personalizzata
- ha **3 componenti chiave** che comunicano fra di loro

Navigation Graph

- file xml che definisce tutti i percorsi che un utente può intraprendere all'interno dell'applicazione
- mostra **visivamente** tutte le destinazioni che si possono raggiungere da un'altra destinazione



NavHost

- container, in cui le destinazioni vengono aggiunte o rimosse mentre l'utente scorre all'interno dell'applicazione
- **NavHostFragment:** implementazione di default
- va incluso all'interno del layout della MainActivity dimostrazione

NavController

- oggetto che tiene traccia della posizione corrente all'interno del NavGraph e gestisce lo spostamento
- chiamare **findNavController().navigate(id)** per cambiare destinazione all'interno del fragment
- da richiamare ad esempio all'interno del setOnClickListener di un bottone
- l'id da specificare può essere sia l'azione che lo schermo di arrivo
- è anche possibile passare argomenti fra 2 destinazioni: dimostrazione
    - creare un **Bundle** e passargli dentro gli argomenti
    - infine, passare il bundle al metodo **navigate()**

Aggiungere e connettere destinazioni

- ogni destinazione deve avere:
  - **id:** identificativo da richiamare come destinazione nel Navigation Graph
  - **nome:** percorso della classe e relativo package
  - **etichetta:** nome della destinazione sotto forma di stringa
  - **layout:** collegamento a layout xml
- designare 1 schermo come **punto di partenza**, ovvero quello che l'utente vede per primo all'apertura dell'applicazione (icona casa)
- connettere le destinazioni tramite frecce
  - in particolare, viene aggiunta un'**azione** con una destinazione di partenza e una di arrivo
  - all'azione si possono aggiungere anche animazioni

Differenze e analogie con Jetpack Compose doc

- anche qui, 3 componenti chiave
- in particolare, NavHost e NavController sono 2 oggetti supportati nativamente, mentre il **Navigation Graph è una classe** (non più file risorsa xml) da creare manualmente, alla quale passare per ogni destinazione lo schermo corrispondente
- le **destinazioni** sono contrassegnate da delle **stringhe**, alle quali corrisponde uno schermo, e non più da un id
- tramite codice e non tramite editor grafico. **Più facile da gestire**, date le molteplici frecce che entrano ed escono da una destinazione
- il navController viene passato come parametro agli schermi che poi chiamano la funzione **navigate()**, esattamente come con il Navigation Component