Corso di Laboratorio di Programmazione per Sistemi Mobile e Tablet

Docente: Dr. Mauro Dragoni

Dipartimento di Ingegneria e Scienze dell’Informazione
Anno Accademico 2022/2023

in questa lezione...

- Fragment
- Navigation Graph
- Layout adattabili

Fragment

- Il Fragment e’ una porzione di Activity.

- Può essere definito come una specie di sub-activity con un suo ruolo funzionale molto importante ed un suo ciclo di vita.

- Un Fragment non puo’ vivere senza una Activity

Fragment

- **Riutilizzo dei componenti di visualizzazione e logica.** I Fragment consentono il riutilizzo di parti dello schermo, comprese le View e la logica degli eventi, più e più volte in modi diversi in molte attività disparate. Ad esempio, utilizzando lo stesso elenco di informazioni all'interno di un'app.

- **Supporto tablet.** Spesso all'interno delle app, la versione tablet di un'attività ha un layout sostanzialmente diverso dalla versione smartphone che è diverso dalla versione TV. I Fragment consentono alle attività specifiche del dispositivo di riutilizzare gli elementi condivisi pur presentando differenze.

- **Orientamento dello schermo.** Spesso all'interno delle app, la versione verticale di un'attività ha un layout sostanzialmente diverso dalla versione orizzontale. I Fragment consentono a entrambi gli orientamenti di riutilizzare elementi condivisi pur presentando differenze.

Fragment

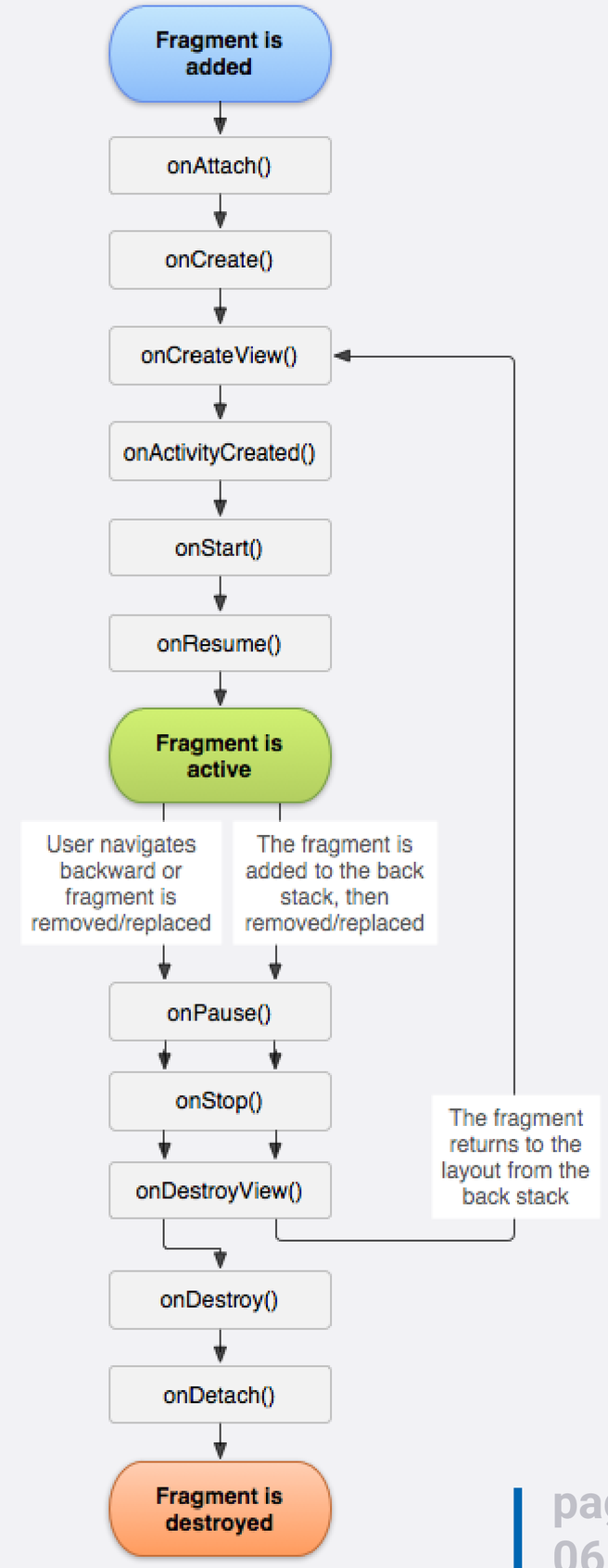
- All'interno di un'app ricca di Fragment, dobbiamo ricordarci di organizzare il nostro codice secondo le migliori pratiche ingegneristiche. All'interno di un'app che utilizza ampiamente i Fragment, dobbiamo tenere presente che il ruolo di un’Activity cambia.

- **Le Activity sono i controllori della navigazione** principalmente responsabili di:
  - Navigazione verso altre attività tramite intenti.
  - Presentare componenti di navigazione come il riquadro di navigazione o il visualizzatore.
  - Nascondere e mostrare frammenti rilevanti utilizzando il gestore frammenti.
  - Ricezione di dati da intenti e passaggio di dati tra frammenti.

- **I Fragment sono controller di contenuto** e contengono la maggior parte delle visualizzazioni, dei layout e della logica degli eventi, tra cui:
  - Layout e viste che mostrano il contenuto dell'app pertinente.
  - Logica di gestione degli eventi associata alle viste rilevanti.
  - Visualizza la logica di gestione dello stato come visibilità o gestione degli errori.
  - Attivazione della richiesta di rete tramite un oggetto client.
  - Recupero e archiviazione dei dati dalla persistenza tramite oggetti del modello.
  - Per ribadire, in un'architettura basata su frammenti, le attività sono per la navigazione ei frammenti sono per le viste e la logica.

Fragment

- **onAttach()** viene chiamato quando un Fragment è connesso a un'attività.
- **onCreate()** viene chiamato per eseguire la creazione iniziale del Fragment.
- **onCreateView()** viene chiamato da Android una volta che il Fragment istanzia le View.
- **onViewCreated()** viene chiamato dopo onCreateView() e si assicura che il contenitore non sia nullo. Qualsiasi configurazione del layout dovrebbe avvenire qui. Es., i listeners.
- **onActivityCreated()** viene chiamato quando l'Activity host ha completato il metodo onCreate().
- **onStart()** viene chiamato quando il Fragment è pronto per essere visualizzato sullo schermo.
- **onResume()** - Assegna risorse "costose" come la registrazione per la posizione, gli aggiornamenti dei sensori, ecc.
- **onPause()** - Rilascia risorse "costose". Conferma eventuali modifiche.
- **onDestroyView()** viene chiamato quando la vista del Fragment viene distrutta, ma il frammento viene comunque mantenuto.
- **onDestroy()** viene chiamato quando il Fragment non è più in uso.
- **onDetach()** viene chiamato quando il Fragment non è più connesso all'attività.



Fragment – Definizione layout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

</LinearLayout>
```

Fragment – La corrispondente classe Java

```java
import androidx.fragment.app.Fragment;

public class FooFragment extends Fragment {
    // The onCreateView method is called when Fragment should create its View object hierarchy,
    // either dynamically or via XML layout inflation.
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {
        // Defines the xml file for the fragment
        return inflater.inflate(R.layout.fragment_foo, parent, false);
    }

    // This event is triggered soon after onCreateView().
    // Any view setup should occur here.  E.g., view lookups and attaching view listeners.
    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        // Setup any handles to view objects here
        // EditText etFoo = (EditText) view.findViewById(R.id.etFoo);
    }
}
```

Fragment – Layout dell’Activity, inserimento statico

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <fragment
        android:name="com.example.android.FooFragment"
        android:id="@+id/fooFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Fragment – Layout dell’Activity, inserimento dinamico

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

  <FrameLayout
       android:id="@+id/your_placeholder"
       android:layout_width="match_parent"
       android:layout_height="match_parent">
  </FrameLayout>

</LinearLayout>
```

```java
// Begin the transaction
FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
// Replace the contents of the container with the new fragment
ft.replace(R.id.your_placeholder, new FooFragment());
// or ft.add(R.id.your_placeholder, new FooFragment());
// Complete the changes added above
ft.commit();
```

Comunicazione Fragment-Activity

- **Bundle** - Activity puo’ costruire un Fragment passando gli argomenti.

- **Methods** - Activity puo’ chiamare i metodi definiti in un Fragment su una sua istanza.

- **Listener** - Fragment puo’ attivare un Listener in un Activity tramite un’interfaccia.

Comunicazione Fragment-Activity

- **Bundle** - Activity puo' costruire un Fragment passando gli argomenti.

```
public class DemoFragment extends Fragment {
    // Creates a new fragment given an int and title
    // DemoFragment.newInstance(5, "Hello");
    public static DemoFragment newInstance(int someInt, String someTitle) {
        DemoFragment fragmentDemo = new DemoFragment();
        Bundle args = new Bundle();
        args.putInt("someInt", someInt);
        args.putString("someTitle", someTitle);
        fragmentDemo.setArguments(args);
        return fragmentDemo;
    }
}
```

```
public class DemoFragment extends Fragment {
   @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       // Get back arguments
       int SomeInt = getArguments().getInt("someInt", 0);
       String someTitle = getArguments().getString("someTitle", "");
   }
}
```

Comunicazione Fragment-Activity

- **Bundle** - Activity puo’ costruire un Fragment passando gli argomenti.

```
// Within the activity
FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
DemoFragment fragmentDemo = DemoFragment.newInstance(5, "my title");
ft.replace(R.id.your_placeholder, fragmentDemo);
ft.commit();
```

Comunicazione Fragment-Activity

- **Methods** - Activity puo’ chiamare i metodi definiti in un Fragment su una sua istanza.

```
public class DemoFragment extends Fragment {
  public void doSomething(String param) {
      // do something in fragment
  }
}
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        DemoFragment fragmentDemo = (DemoFragment)
            getSupportFragmentManager().findFragmentById(R.id.fragmentDemo);
        fragmentDemo.doSomething("some param");
    }
}
```

Comunicazione Fragment-Activity

- **Listener** - Fragment puo’ attivare un Listener in un Activity tramite un’interfaccia.

```
import androidx.fragment.app.Fragment;

public class MyListFragment extends Fragment {
  // ...
  // Define the listener of the interface type
  // listener will the activity instance containing fragment
  private OnItemSelectedListener listener;

  // Define the events that the fragment will use to communicate
  public interface OnItemSelectedListener {
    // This can be any number of events to be sent to the activity
    public void onRssItemSelected(String link);
  }

  // Store the listener (activity) that will have events fired once the fragment is attached
  @Override
  public void onAttach(Context context) {
      super.onAttach(context);
      if (context instanceof OnItemSelectedListener) {
        listener = (OnItemSelectedListener) context;
      } else {
        throw new ClassCastException(context.toString()
            + " must implement MyListFragment.OnItemSelectedListener");
      }
  }

  // Now we can fire the event when the user selects something in the fragment
  public void onSomeClick(View v) {
     listener.onRssItemSelected("some link");
  }
}
```

Comunicazione Fragment-Activity

- **Listener** - Fragment puo’ attivare un Listener in un Activity tramite un’interfaccia.

```
import androidx.appcompat.app.AppCompatActivity;

// Activity implements the fragment listener to handle events
public class RssfeedActivity extends AppCompatActivity implements MyListFragment.OnItemSelectedListener {
    // Can be any fragment, `DetailFragment` is just an example
    DetailFragment fragment;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rssfeed);
        // Get access to the detail view fragment by id
        fragment = (DetailFragment) getSupportFragmentManager()
            .findFragmentById(R.id.detailFragment);
    }

    // Now we can define the action to take in the activity when the fragment event fires
    // This is implementing the `OnItemSelectedListener` interface methods
    @Override
    public void onRssItemSelected(String link) {
        if (fragment != null && fragment.isInLayout()) {
            fragment.setText(link);
        }
    }
}
```

Fragment – Hello World

- creiamo il layout per la Activity in cui ricaviamo il posto per alloggiare il Fragment;

- definiamo il layout del Fragment che conterra' il vero aspetto della Activity e, di conseguenza, anche la stringa "Hello World";

- definiamo la classe Fragment che essenzialmente servira' a caricare il layout di cui al punto precedente;

- creiamo la Activity che svolgera' per lo più il ruolo di bacino di Fragment.

Fragment – Hello World

- file: `res/layout/fragment_main.xml`

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="@dimen/activity_vertical_margin">

    <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />

</RelativeLayout>
```

Fragment – Hello World

- file: `res/layout/activity_main.xml`

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/container"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

Fragment – La classe HelloFragment

```
public class HelloFragment extends Fragment
{
        public HelloFragment() {}

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                         Bundle savedInstanceState)
        {
                View rootView = inflater.inflate(R.layout.fragment_main, container, false);
                return rootView;
        }
}
```

Fragment – La classe MainActivity

```
public class MainActivity extends ActionBarActivity
{
	@Override
	protected void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		if (savedInstanceState == null)
		{
			getSupportFragmentManager().beginTransaction()
			.add(R.id.container, new HelloFragment()).commit();
		}
	}
}
```

Fragment – Le FragmentTransactions

- **add** Aggiunge un Fragment alla Activity
- **remove** Rimuove un Fragment precedentemente aggiunto
- **replace** Sostituisce un Fragment con un altro
- **hide** Nasconde un Fragment
- **show** Mostra un Fragment precedentemente nascosto

Altri metodi di interesse legati al FragmentManager:

- addOnBackStackChangedListener()
- beginTransaction()
- findFragmentById(int id)
- findFragmentByTag(String tag)
- popBackStack()
- executePendingTransactions()

Fragment: hide vs. replace

- **Replace**

```
// Within an activity

private FragmentA fragmentA;
private FragmentB fragmentB;
private FragmentC fragmentC;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    if (savedInstanceState == null) {
        fragmentA = FragmentA.newInstance("foo");
        fragmentB = FragmentB.newInstance("bar");
        fragmentC = FragmentC.newInstance("baz");
    }
}

protected void displayFragmentA() {
    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    // removes the existing fragment calling onDestroy
    ft.replace(R.id.flContainer, fragmentA);
    ft.commit();
}
```

Fragment: hide vs. replace

- **Hide**

```
// ...onCreate stays the same

// Replace the switch method
protected void displayFragmentA() {
    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    if (fragmentA.isAdded()) { // if the fragment is already in container
        ft.show(fragmentA);
    } else { // fragment needs to be added to frame container
        ft.add(R.id.flContainer, fragmentA, "A");
    }
    // Hide fragment B
    if (fragmentB.isAdded()) { ft.hide(fragmentB); }
    // Hide fragment C
    if (fragmentC.isAdded()) { ft.hide(fragmentC); }
    // Commit changes
    ft.commit();
}
```

Navigation

- **La navigazione si riferisce alle interazioni che consentono agli utenti di navigare, entrare e uscire dai diversi contenuti all'interno dell'app.**

- La navigazione all’interno delle applicazioni Android è costituits da tre parti chiave:

  - **Navigation Graph**: una risorsa XML che contiene tutte le informazioni relative alla navigazione in un'unica posizione centralizzata. Ciò include tutte le singole aree di contenuto all'interno dell'app, denominate destinazioni, nonché i possibili percorsi che un utente può intraprendere attraverso l'app.

  - **NavHost**: un contenitore vuoto che visualizza le destinazioni dal grafico di navigazione. Il componente Navigation contiene un'implementazione NavHost predefinita, NavHostFragment, che visualizza le destinazioni dei frammenti.

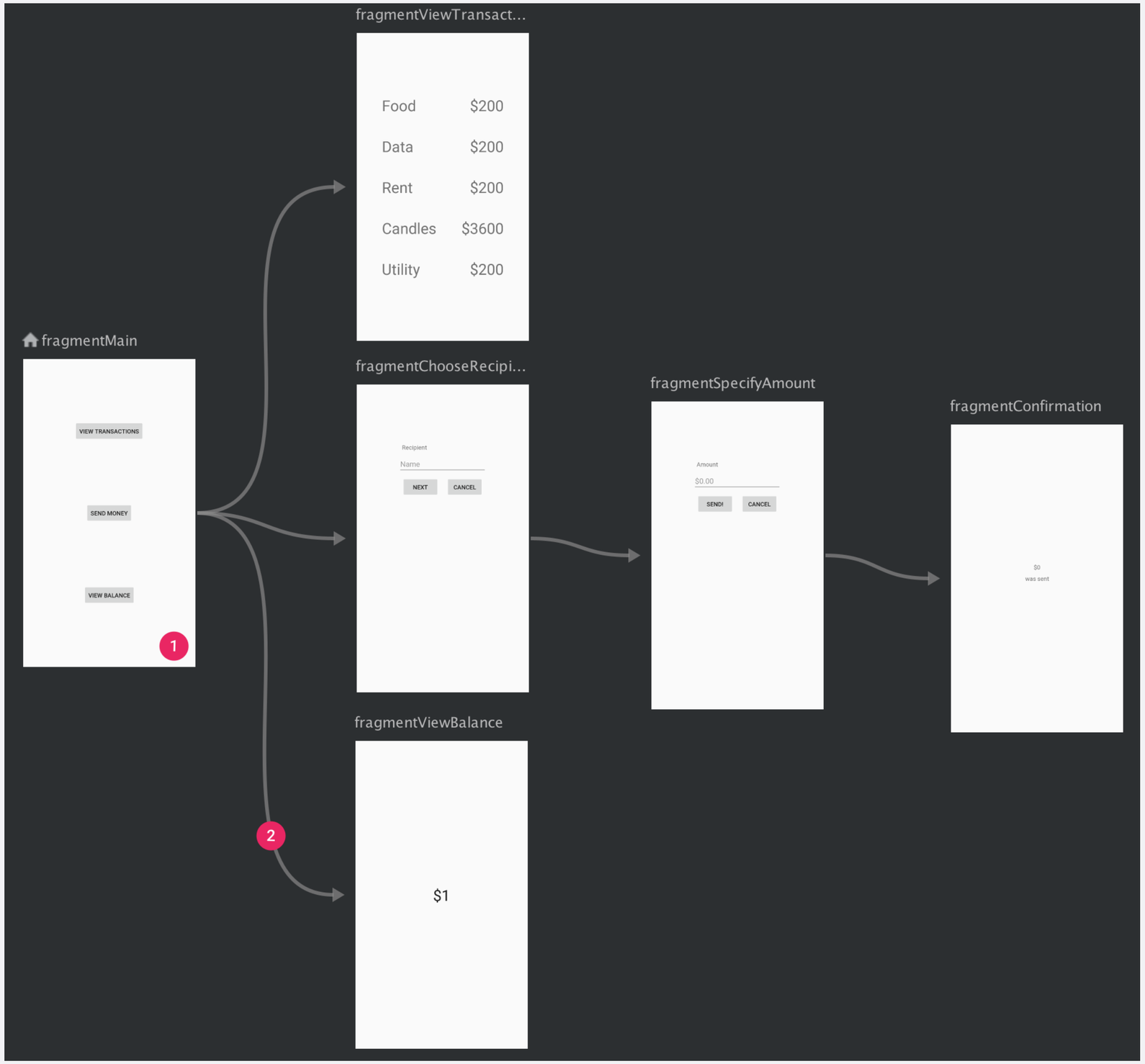
  - **NavController**: un oggetto che gestisce la navigazione dell'app all'interno di un **NavHost**. Il **NavController** orchestra lo scambio del contenuto di destinazione nel **NavHost** mentre gli utenti si spostano nell'app.

Navigation

- **La navigazione si riferisce alle interazioni che consentono agli utenti di navigare, entrare e uscire dai diversi contenuti all'interno dell'app.**

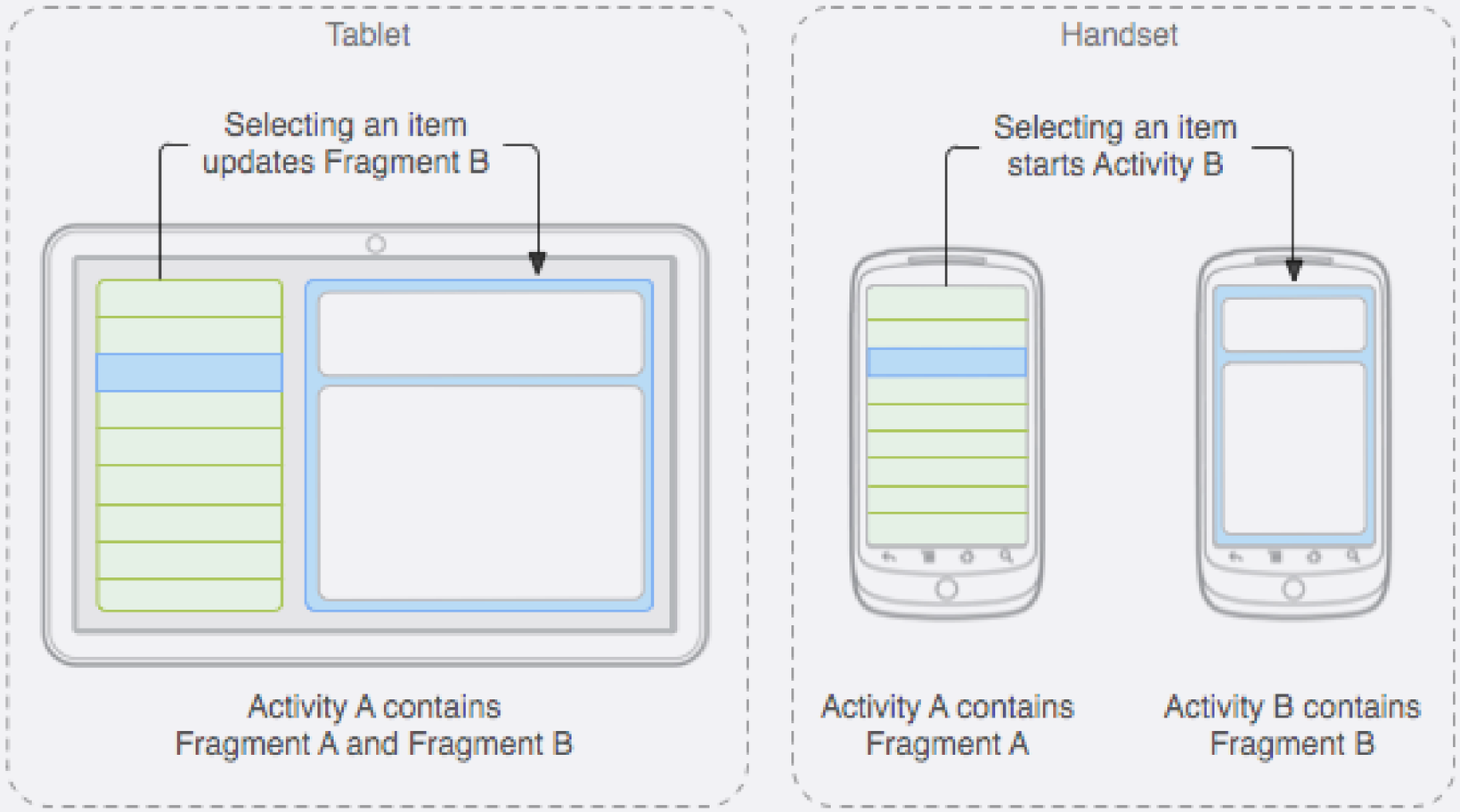
- La gestione della navigazione offre una serie di altri vantaggi:
  - Gestione delle transazioni di Fragments.
  - Gestire correttamente le azioni Up e Back per impostazione predefinita.
  - Fornire risorse standardizzate per animazioni e transizioni.
  - Implementazione e gestione dei deep linking.
  - Utilizzo di patterns per la creazione e gestione della navigazione stessa
  - Safe Args: un plug-in Gradle che fornisce la sicurezza del tipo durante la navigazione e il passaggio di dati tra le destinazioni.
  - Supporto al ViewModel: è possibile definire lo scope di un ViewModel in un grafico di navigazione per condividere i dati relativi all'interfaccia utente tra le destinazioni del grafico.

Navigation Graph



1) Destinations
2) Actions

Fragments e Layout Adattabili



- i Fragments rappresentano una soluzione al problema della frammentazione dei dispositivi

Layout adattabili – Configurazione delle risorse

- ***layout-large-land*** : verra’ usato solo per dispositivi con display **large** in posizione **landscape**;

- ***layout*** : la cartella di default. Verra’ chiamata in causa per tutte le altre situazioni.

Layout adattabili – Configurazione delle risorse

- File 1: res/layout/activity_main.xml:

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/container"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

Layout adattabili – Configurazione delle risorse

- File 2: res/layout-large-land/activity_main.xml:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <fragment android:name="it.unitn.lpsmt.gui.fragments.Fragment1"
    android:id="@+id/fragment1"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"/>

    <fragment android:name="it.unitn.lpsmt.gui.fragments.Fragment2"
    android:id="@+id/fragment2"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="2"/>

</LinearLayout>
```

Layout adattabili – Comunicazione tra Fragments

- La Activity svolge il ruolo di snodo funzionale e di comunicazione tra i due Fragments.

- Per poter riutilizzare i Fragments in piu’ contesti, e’ necessario che non si “conoscano” tra loro ne’ che conoscano la Activity alla quale devono essere collegati.

- All’interno dei Fragments non viene mai menzionata esplicitamente la classe di appartenenza della Activity.

- Viene definita un’interfaccia che viene implementata dalla Activity stessa.

Layout adattabili – Osservazioni

- Fragments collegati tra loro il minimo possibile per permetterne un maggior riutilizzo.

- Activity usata come snodo di comunicazione tra Fragments.

- Risorse in grado di separare all’origine i layout distinguendo con id diversi i contenitori dei Fragments.

- Ogni classe Fragment deve avere sin dalla nascita uno scopo ben preciso e contenere tutta la logica necessaria per raggiungerlo in maniera indipendente.