Views

Android Views è il paradigma standard per la creazione e manipolazione di tutti gli elementi grafici che compongano la *User Interface*. Una **View** definisce un concetto primitivo ma essenziale per lo sviluppo di applicazioni, contenitrice di tutti gli oggetti visualizzati a schermo. Pertanto una *View* è responsabile non solo della modellazione di tutti gli elementi grafici che caratterizzano l'applicazione di riferimento, ma anche della gestione degli eventi che li riguardano, come l'interazione con agenti esterni oppure l'acquisizione di dati.

Esempi più comuni di *View* sono distinti in:

- **Widgets**, descritte come delle etichette, da non confondere con il sistema software già in uso all'interno di dispositivi mobili
- **Layouts**, insieme di caratteristiche grafiche e comportamentali

Per determinare una *View* si adottano due metodi essenziali:

- **Declarative method**, metodo dichiarativo di una *View* tramite il file *XML*, simile a quanto definito a livello di *Activity*, in cui il *Manifest* assume un'importanza essenziale. Il loro accesso tramite linguaggi di programmazione, *Java* oppure *Kotlin*, avviene tramite il metodo **findViewById** a cui è passato come parametro l'elemento grafico contenuto all'interno della medesima *View*
- METTERE ESEMPIO DI SLIDE 5-6
- **Programmatic View**, in questo ambito le *View* sono direttamente create in *Java* o *Kotlin*, immettendo il *contesto* di riferimento. Tuttavia si tratta di un approccio non raccomandato, a causa del fatto che nella sezione *code* occorre dettare tutte le proprietà visive

Handling Events

La fase successiva ritrae la gestione degli eventi associati agli oggetti visivi. *Java*/*Kotlin* manipolano i possibili eventi attraverso la keyword **OnClick**; mediante la stessa definizione è possibile addirittura forzare l'avvenimento di un evento senza alcuna interazione.

Esistono tre differenti metodi per gestire gli eventi, suddivisi in:

- **XML**, il tutto gestito mediante le **callbacks**, direttamente indicate nel file; tuttavia il metodo descritto riguarda solamente un numero ristretto di componenti.
  METTERE ESEMPIO SLIDE 10
- **Event Handlers**, in questa sezione ogni *View* contiene un ammontare di metodi, richiamati qualora dovesse verificarsi un evento. Le differenti funzioni citate sono frutto di innumerevoli *extend* adattati nella classe *View*, riferiti agli oggetti grafici, per cui tramite questo approccio occorre compiere molteplici *override*; di conseguenza più articolata risulterà la *View*, maggiore sarà l'*impraticabilità*
- **Event Listeners**, in quest'ultima casistica, ogni *View* delega l'implementazione del comportamento, successivo ad un certo evento, ad un oggetto. In tal senso, ogni *listener* gestisce una singola tipologia di evento e contiene un unico metodo *callback*. Si evince

in questa breve descrizione la volontà di dividere nettamente ciò che è ritenuto dinamico, come l'interfaccia grafica, da elementi che costituiscono la logica dell'applicazione, ossia il comportamento successivo alla veridicità dell'evento, sovrapponendo un layer di astrazione in grado di isolare le due entità da modifiche e cambiamenti reciproci

```
Button button;
class MainActivity extends AppCompatActivity implements OnClickListener{
  @Override
  protected void onCreate(Bunflr savedInstanceState) {
    ...
    button = findViewById(R.id.button1);
    button.setOnClickListener(this);
  }

  @Override
  void onClick(View v) { }
}
```

Gestione evento tramite Event Listener

Di seguito sono definiti brevemente gli aspetti salienti della sezione di codice riportata:

- In questo esempio la *MainActivity* implementa la classe astratta *OnClickListener*, affinchè possa gestire correttamente gli eventi scaturiti da interazioni con agenti esterni
- All'interno del metodo *onCreate()*, viene acquisito il bottone immesso all'interno della *View*. In questo modo è possibile, attraverso il *listener*, definire il comportamento del bottone una volta creato; si ricorda che la gestione e il coordinamento degli eventi avviene tramite la keyword *OnClick*
- Infine all'interno del metodo *onClick()*, da cui si osserva la keyword *Override*, è sviluppata la vera logica del bottone, rispettando in questo modo la suddivisione in business logic e classi di alto livello descritta precedentemente. Nel caso in cui dovessero essere presenti più bottoni occorrerà implementare uno *switch* che possa distinguere le possibili casistiche

Layout

Un **Layout** deve estendere una **ViewGroup**. Una *ViewGroup* è un contenitore di *View* adottato per definire la struttura del loro *Layout*.

Ogni *View* in un *Layout* deve specificare:

- Una *lunghezza*, espressa in **android:layout_height**
- Una *larghezza*, espressa in **android:layout_width**
- Una *dimensione*, espressa numericamente oppure staticamente tramite la sintassi *match_parent* oppure *wrap_content*
- METTERE ESEMPIO SLIDE 22