Progettazione, Algoritmi e Computabilità
Sviluppo di Applicazioni Android

Michele Beretta

michele.beretta@unibg.it

Setup

Link Utili

- Android Studio: https://developer.android.com/studio/index.html
- Documentazione per sviluppatori: https://developer.android.com/docs/
- Link alla guida per lo sviluppo della prima app: https://developer.android.com/training/basics/firstapp

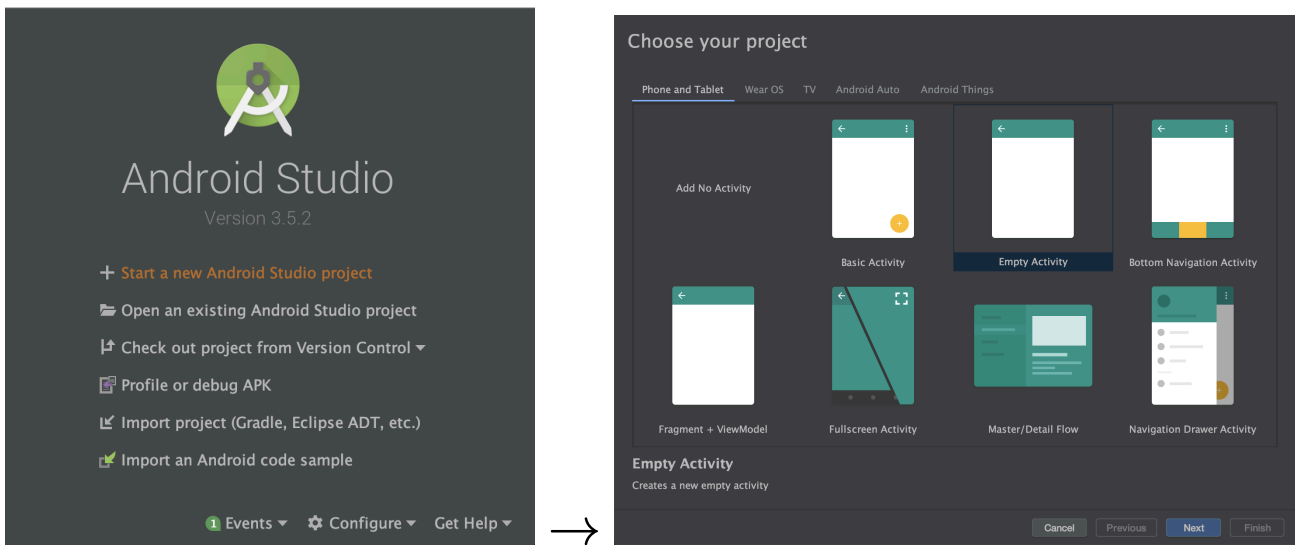
Ambiente di sviluppo

Per la realizzazione di applicazioni Android utilizzeremo l'ambiente *Android Studio*, basato su IntelliJ IDEA (funzionante anche su MacOS).

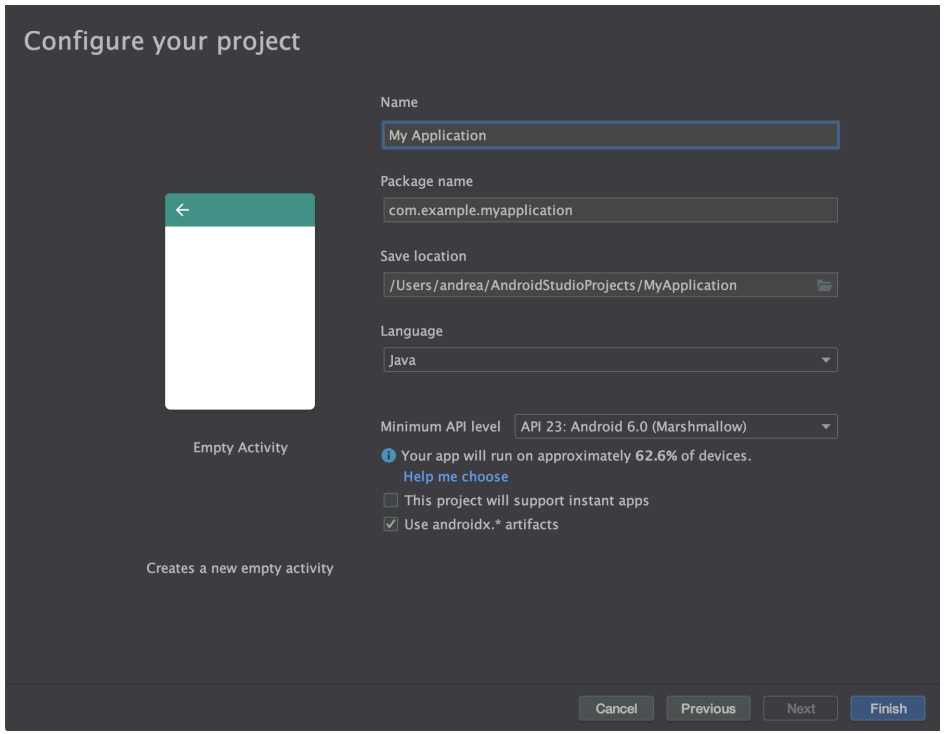
Installando l'IDE viene installando anche l'SDK che contiene:

- *SDK Manager*: permette di gestire i tool, le piattaforme, APIs ed altre componenti
- *AVD (Android Virtual Devices) Manager*: fornisce una interfaccia grafica per gestire e creare AVD che sono necessari per l'Android Emulator
- *Emulator*: emulatore Android per testare le app
- *Dalvik Debub Monitor Server*

Creazione di un nuovo progetto con Android Studio



Creazione di un nuovo progetto con Android Studio

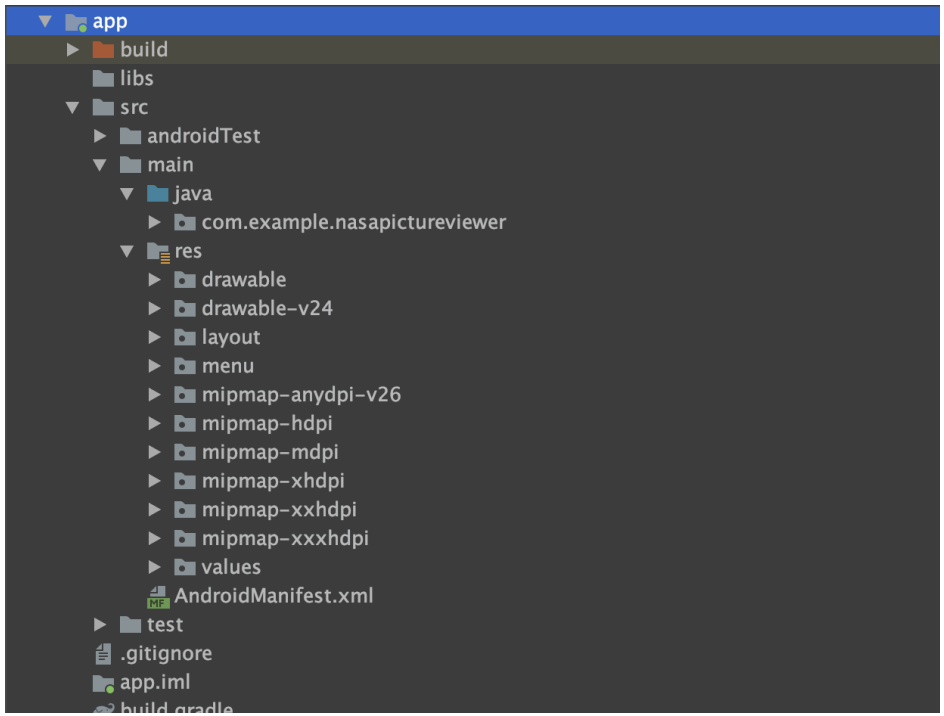


È possibile specificare:

- *Name*: nome dell'app
- *Package name*: namespace del package nell'app, deve essere univoco tra tutti i package installati nel sistema Android
- *Save Location*: directory nella quale il progetto sarà salvato
- *Min. API Level*: versione di Android minima necessaria per eseguire l'app

Grafica e Manifest

Directory structure di un progetto



Dentro la directory `src`:

- `main/java`: codice Java della app
- `res`: file di risorse, fra cui
  - layout
  - testi predefiniti
  - dati multimediali (suoni, immagini, etc)
- `main/AndroidManifest.xml`: informazioni essenziali che la app condivide con Android

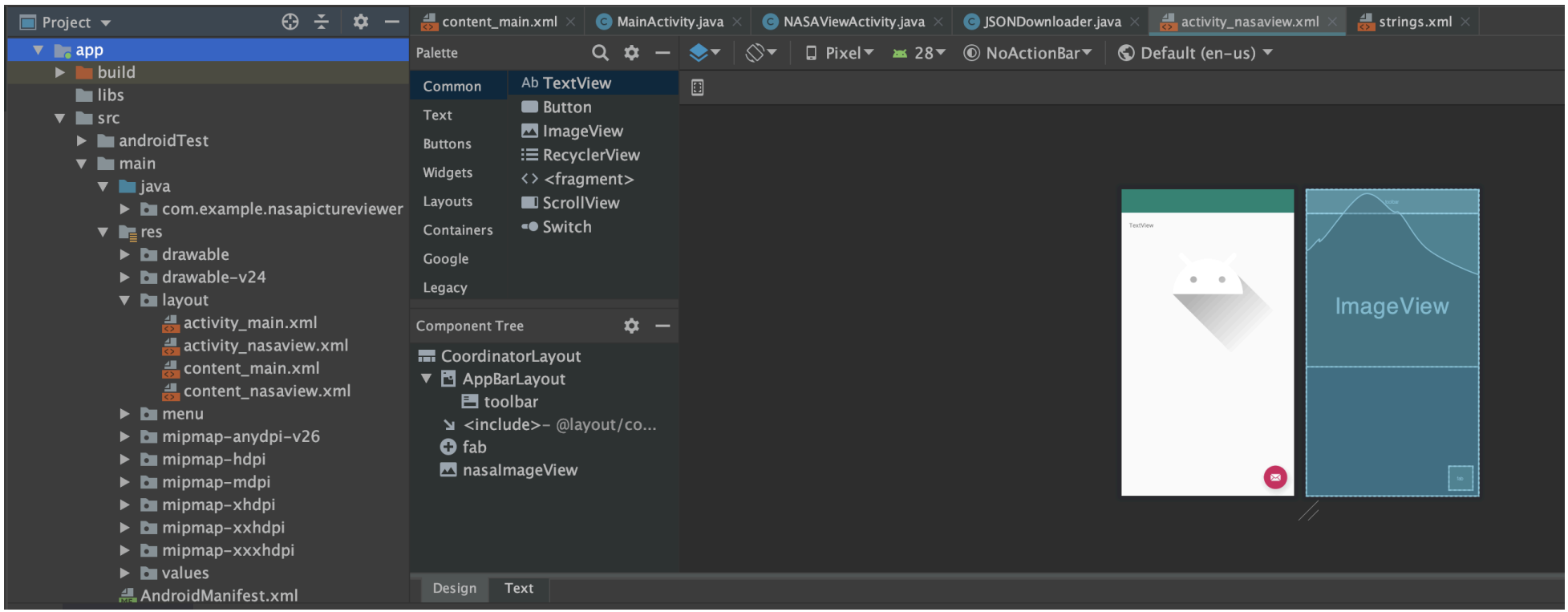
GUI

La GUI Android è costruita utilizzando una gerarchia di due tipologie di oggetti:

- `View`: Widget UI (bottoni, campi testo, immagini, etc)
- `ViewGroup`: containers invisibili che definiscono come devono essere mostrati i componenti in essi contenuti (*layout*)

Layout (1)

Il layout di una app può essere definito in modo grafico…



Layout (2)

…oppure in modo testuale editando il file XML.

```
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NASAViewActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_nasaview" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />

    <ImageView
        android:id="@+id/nasaImageView"
        android:layout_width="match_parent"
        android:layout_height="424dp"
        android:contentDescription="NASA Picture"
        app:srcCompat="@drawable/ic_launcher_foreground" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Layout (3)

Esistono diverse tipologie di Layout, tra cui:

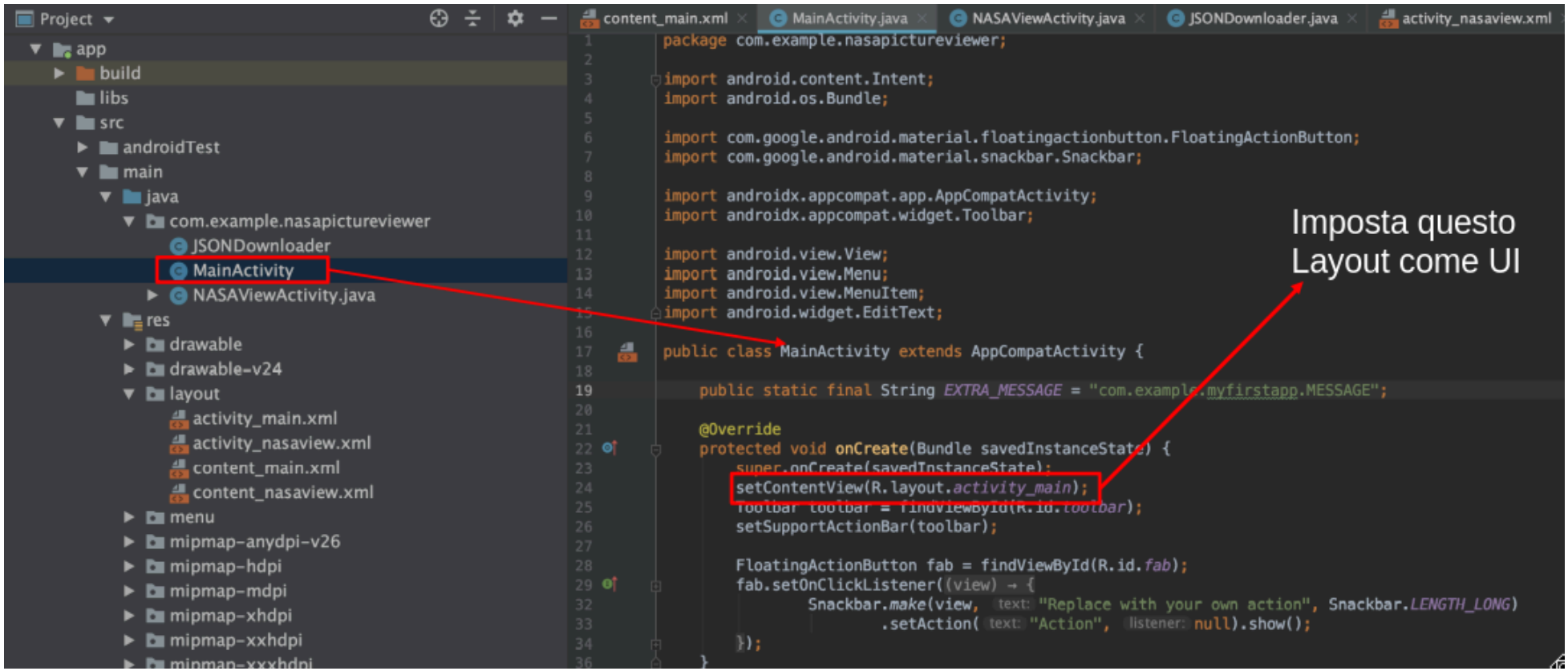
- *Linear Layout*: tutti gli elementi figli vengono allineati in una singola direzione, orizzontalmente o verticalmente
- *Relative Layout*: la posizione di un elemento figlio viene espressa in modo relativo alla posizione di un altro elemento figlio
- *List-View*: gli elementi vengono messi in una lista che si può scorrere
- *Table-Layout*: gli elementi vengono mostrati in una griglia bidimensionale scorribile

View

Esistono diverse tipologie di componenti View che possono essere inserite:

- `TextView`
- `ImageViimages/ew`
- `Button`
- `EditText`
- `CheckBox`
- `RadioGroup` e `RadioButton`
- `ToggleButton` (pulsanti che rimangono premuti)
- `Spinner` (menù a tendina)
- `Switch` (classici per attivare-disattivare una funzionalità)

View – Codice Java



Android Manifest

https://developer.android.com/guide/topics/manifest/manifest-intro.html

Il file Android Manifest permette di comunicare informazioni essenziali al sistema Android. In particolare:

- Nome del package dell'app
- Componenti dell'app
- Quali permessi richiede l'app (e.g. l'accesso alla fotocamera, ad internet, scrittura su SD)
- Quali permessi devono richiedere le altre applicazioni per interagire con i componenti della nostra app
- Librerie richieste
- Minima versione delle API Android

Android Manifest – Esempio


ID univoco del package
Componenti
Permessi richiesti dall'App

Durante l’esecuzione

Esecuzione

L’esecuzione di una applicazione può essere fatta:

- Tramite l’emulatore installato insieme a Android Studio e l’SDK di Android
- Collegando uno smartphone Android (in modalità sviluppatore) attaccato al PC sul quale si sta sviluppando l’applicazione via cavo USB

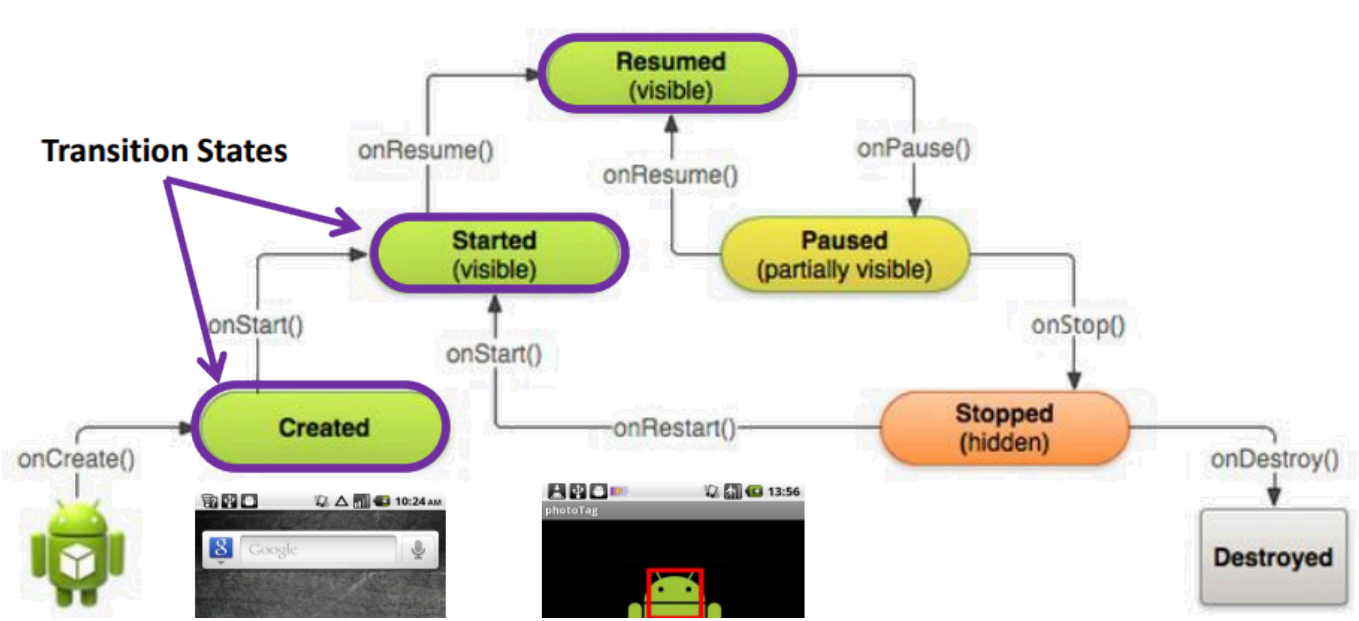
Activity

Una **Activity** è una schermata che viene mostrata sul dispositivo in un certo istante di tempo.

Una app, quindi, ha normalmente più di una Activity e l'utente deve essere in grado di passare da una Activity alla successiva, o viceversa.

Essendo l'Activity la schermata visualizzata, su ogni activity lo sviluppatore può disegnare una specifica interfaccia grafica.

Ciclo di vita di una Activity



Lo stato *paused* si ha quando una Activity che non ricopre l'intero schermo si sovrappone ad un'altra (e.g. un messaggio di richiesta conferma).

Funzioni di callback (1)

Android crea delle nuove istanze di Activity utilizzando la chiamata del metodo `onCreate()`. È quindi necessario che ogni Activity implementi questo metodo:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Set the user interface layout for this Activity
    // The layout file is defined in the project res/layout/main_activity.xml file
    setContentView(R.layout.main_activity);

    // Initialize member TextView so we can manipulate it later
    mTextView = (TextView) findViewById(R.id.text_message);

    // Make sure we're running on Honeycomb or higher to use ActionBar APIs
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        // For the main activity, make sure the app icon in the action bar
        // does not behave as a button
        ActionBar actionBar = getActionBar();
        actionBar.setHomeButtonEnabled(false);
    }
}
```

Funzioni di callback (2)

Il metodo `onResume()` è chiamato tutte le volte che l'Activity viene richiamata in primo piano:

```
@Override
public void onResume() {
    super.onResume();  // Always call the superclass method first

    // Get the Camera instance as the activity achieves full user focus
    if (mCamera == null) {
        initializeCamera(); // Local method to handle camera init
    }
}
```

Funzioni di callback (3)

Il metodo `onPause()` è normalmente utilizzato per:

- Fermare animazioni o azioni che sono in esecuzione
- Confermare modifiche non ancora salvate
- Rilasciare risorse di sistema, e.g. come sensori, fotocamera

```
@Override
public void onPause() {
    super.onPause();  // Always call the superclass method first

    // Release the Camera because we don't need it when paused
    // and other activities might need to use it.
    if (mCamera != null) {
        mCamera.release()
        mCamera = null;
    }
}
```

Funzioni di callback (4)

Quando una activity riceve la chiamata del metodo `onStop()` passa a non essere più visibile e dovrebbe rilasciare tutte le risorse che possedeva.

Rispetto a `onPause()`, il metodo `onStop()` esegue operazioni di *shutdown* più intensive, come ad esempio operazioni di scrittura sul database.

```
@Override
protected void onStop() {
    super.onStop();  // Always call the superclass method first

    // Save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    getContentResolver().update(
            mUri,    // The URI for the note to update.
            values,  // The map of column names and new values to apply to them.
            null,    // No SELECT criteria are used.
            null     // No WHERE columns are used.
            );
}
```

Funzioni di callback (5)

Il metodo `onStart()` è chiamato tutte le volte che una Activity passa ad essere visibile. Un tipico utilizzo di questo metodo è quello di controllare che le risorse di sistema necessarie siano abilitate.

```
@Override
protected void onStart() {
    super.onStart();  // Always call the superclass method first

    // The activity is either being restarted or started for the first time
    // so this is where we should make sure that GPS is enabled
    LocationManager locationManager =
            (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    boolean gpsEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

    if (!gpsEnabled) {
        // Create a dialog here that requests the user to enable GPS, and use an intent
        // with the android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS action
        // to take the user to the Settings screen to enable GPS when they click "OK"
    }
}
```

Activity Manager

Lanciare una Activity è un procedimento abbastanza costoso in termini di risorse:

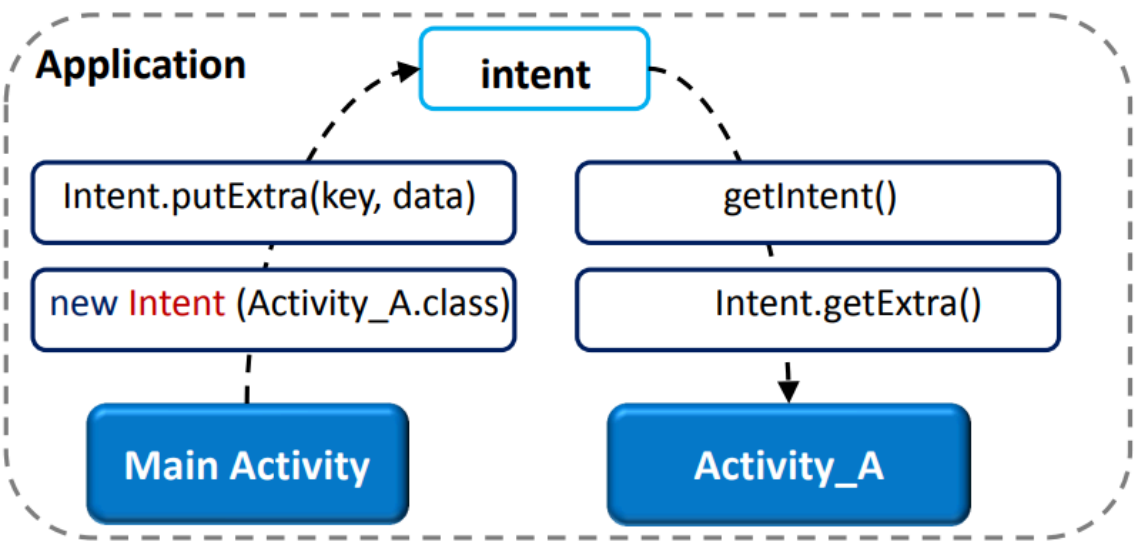
- Bisogna creare un nuovo processo Linux
- Bisogna allocare le risorse e la memoria per gli oggetti della UI
- Bisogna impostare correttamente lo schermo
- E così via

Per questi motivi, l'intero processo viene gestito dal componente *Activity Manager*, che lo esegue nel modo più ottimizzato possibile, e non dallo sviluppatore.

Intent

http://developer.android.com/guide/components/intents-filters.html

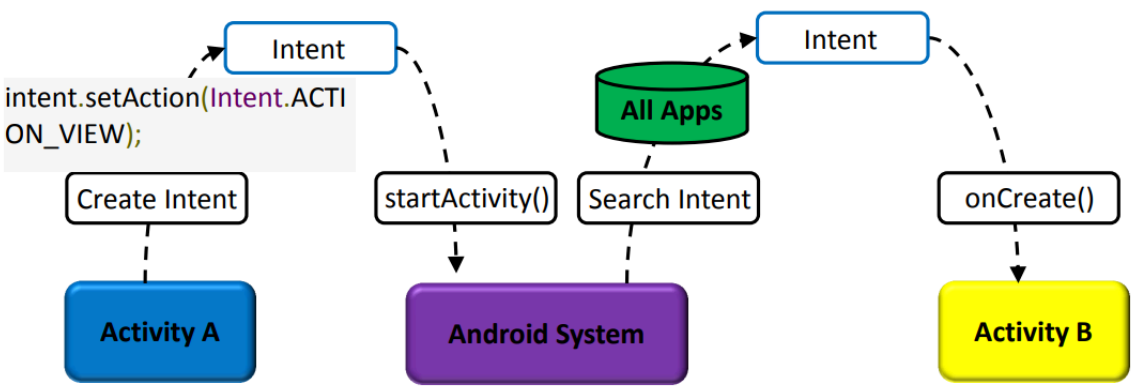
Un *Intent* è un oggetto utilizzato per lo scambio di messaggi, che permette di facilitare la comunicazione tra Activity.



Intent

Esistono due tipologie di Intent:

- *Intent espliciti*: il componente viene avviato specificando il suo nome (viene usato all'interno di una app)
- *Intent impliciti*: il componente non viene avviato specificando il suo nome, ma dichiarando solamente l'azione che deve essere eseguita



Avviare una Activity con passaggio di dati

```java
public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
public void sendMessage(View view)
{
    Intent intent = new Intent(this, DisplayMessageActivity.class);

    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get Intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    // Create the text view
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);

    // Set the text view as the activity layout
    setContentView(textView);
}
```

Esercizi

Esercizio 1

Creare un progetto vuoto in Android Studio e realizzare la prima app introduttiva spiegata al link https://developer.android.com/codelabs/build-your-first-android-app.

Se durante il task 9 avete dei problemi con gradle, trovate la soluzione qui https://developer.android.com/guide/navigation/navigation-pass-data#Safe-args.

Esercizio 1

In particolare, il file `build.gradle` (Project) deve avere la seguente sezione

```
buildscript {
  dependencies {
    classpath "com.android.tools.build:gradle:3.5.4"
    classpath "androidx.navigation:navigation-safe-args-gradle-
plugin:2.5.3"
  }
}
```

Esercizio 1

Mentre il file `build.gradle` (Module) dev'essere strutturato così

```
plugins {
  id "com.android.application"
  id "androidx.navigation.safeargs"
}
...
dependencies {
  implementation "androidx.lifecycle:lifecycle-viewmodel-
savedstate:2.4.1"
  ...
}
```

Esercizio 2

1. Creare una view di login
2. Fare in modo che alla pressione del pulsante *Login* si avvii una nuova Activity
3. In tale activity mostrare il nome utente e la password inseriti nella view di login (passati tramite Intent) e l'immagine del giorno della Nasa