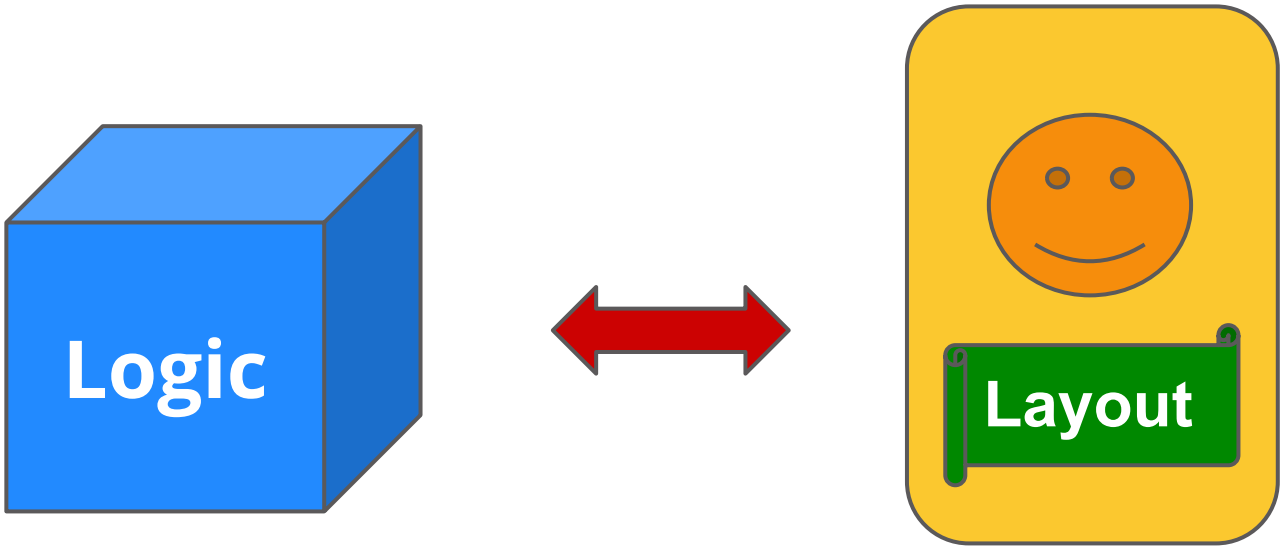
Modulo 6

Interfaccia grafica in Android

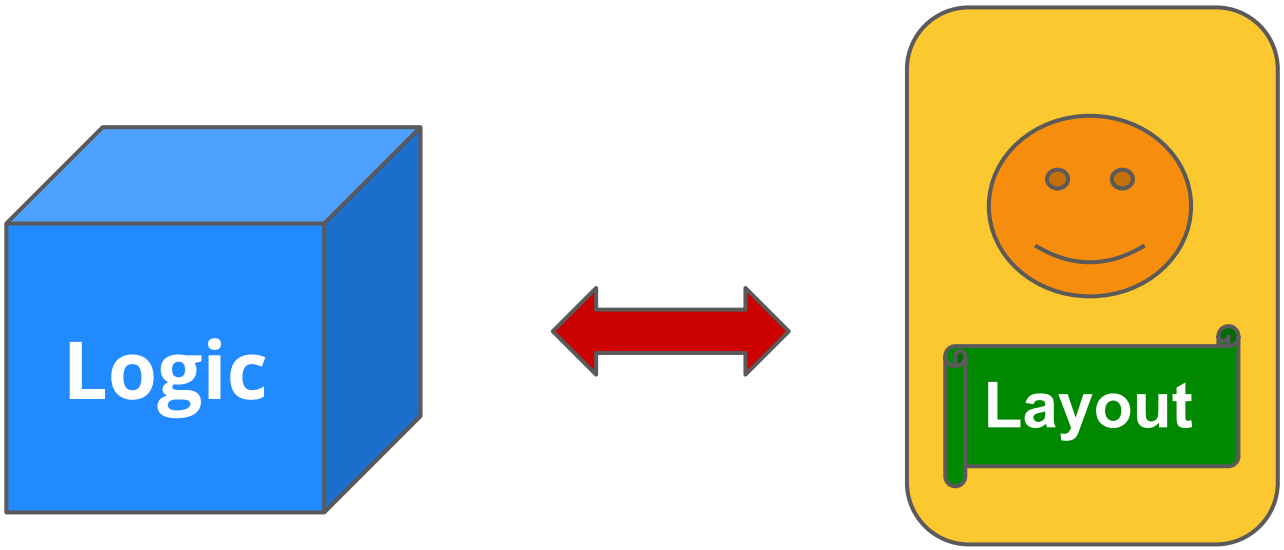
6.1

Principali layout

6.1.1 Layout Declaration



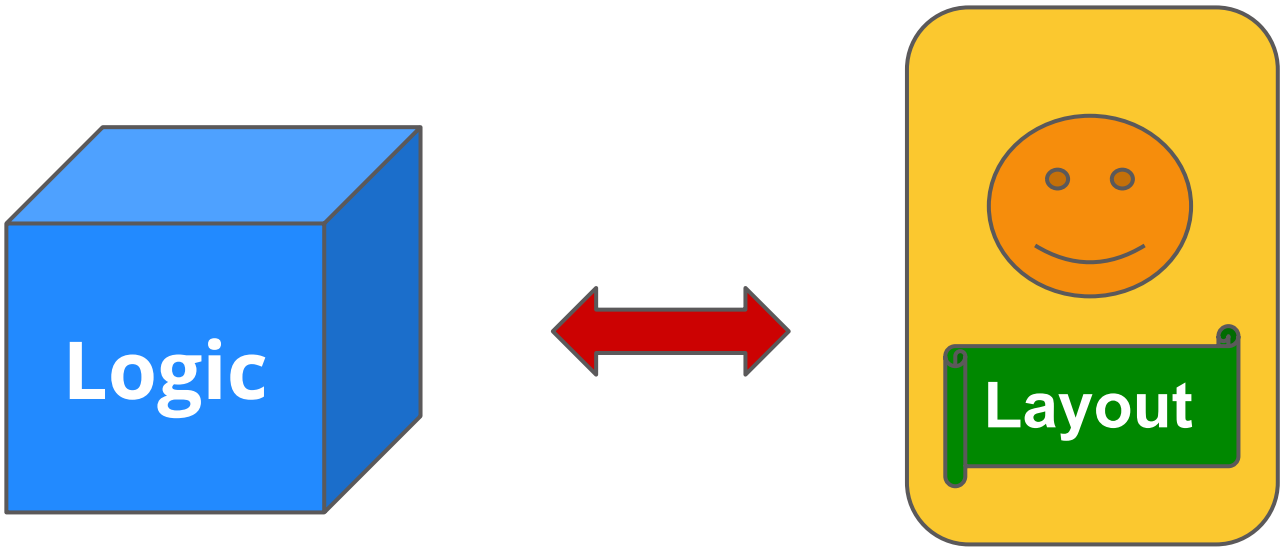
6.1.1 Layout Declaration



Java

XML

6.1.1 Layout Declaration



Java

XML / Java

6.1.1 Layout Declaration - XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
  <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
  <Button android:id="@+id/my_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Hello, I am a Button" />
</LinearLayout>
```

```
Button myButton = (Button) findViewById(R.id.my_button);
```

6.1.1 Layout Declaration -Java

```
ViewGroup linearLayout = (ViewGroup)findViewById(R.id.
linearLayoutID);


Button bt = new Button(this);


bt.setText("Un Bottone");
bt.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
                                    LayoutParams.WRAP_CONTENT));
linerLayout.addView(bt);
```

6.1.1 Layout Declaration - Inflater

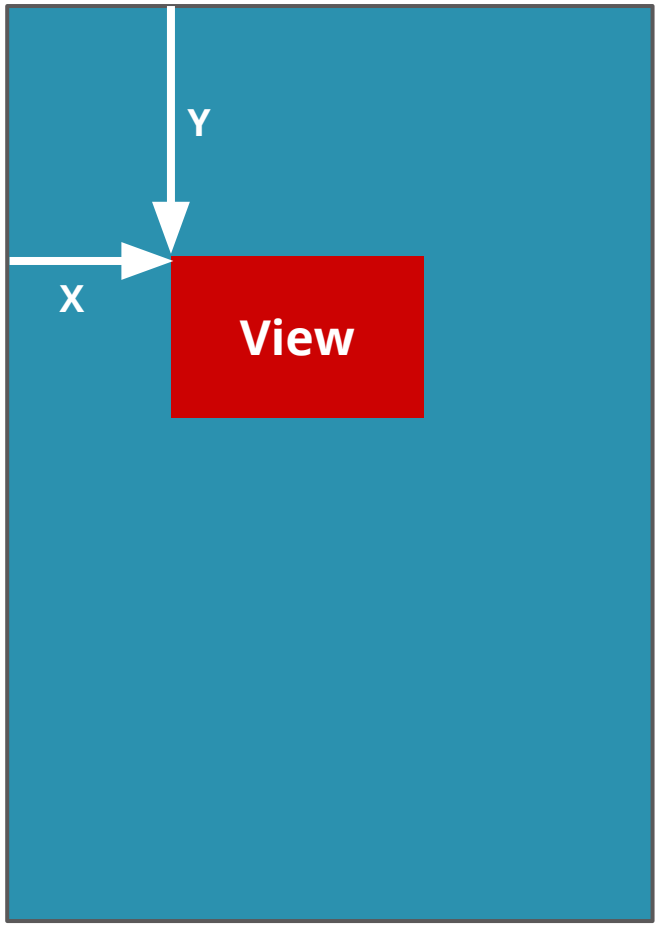
```
ViewGroup linearLayout = (ViewGroup) getContext()
    .getLayoutInflater().inflate(R.layout.contact_row, parent,
false);

Button bt = linearLayout.findViewById(R.id.my_button);

bt.setText("Un Bottone");
bt.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
                                    LayoutParams.WRAP_CONTENT));
linerLayout.addView(bt);
```

6.1.2 Layout - Posizionamento / Dimensione

In principio...

era l'**AbsoluteLayout**

e le dimensioni in **px**



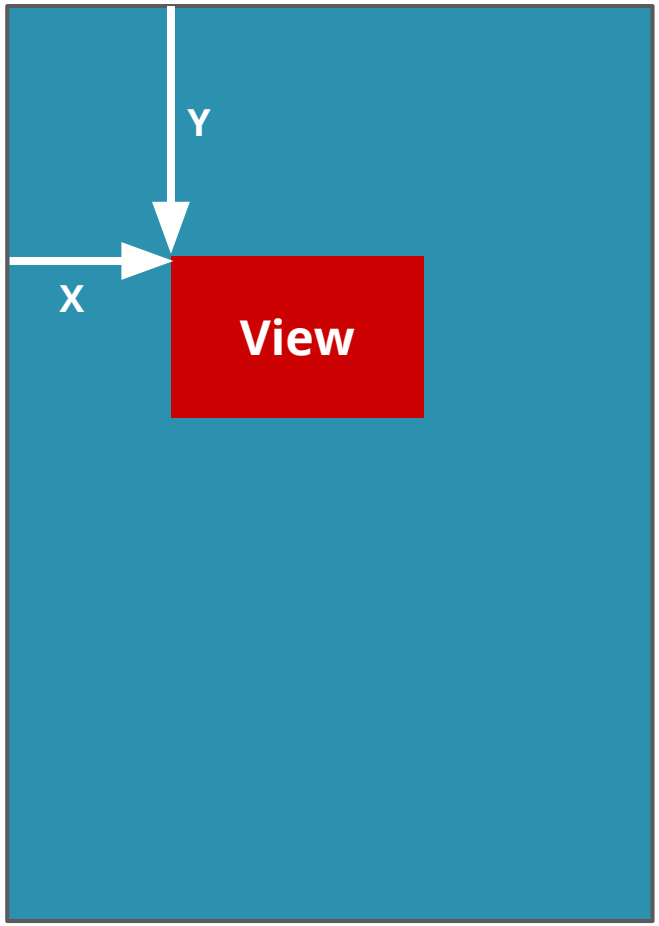
6.1.2 Layout - Posizionamento / Dimensione

In principio...

era l'**AbsoluteLayout**

e le dimensioni in **px**

Problemi:



6.1.2 Layout - Posizionamento / Dimensione

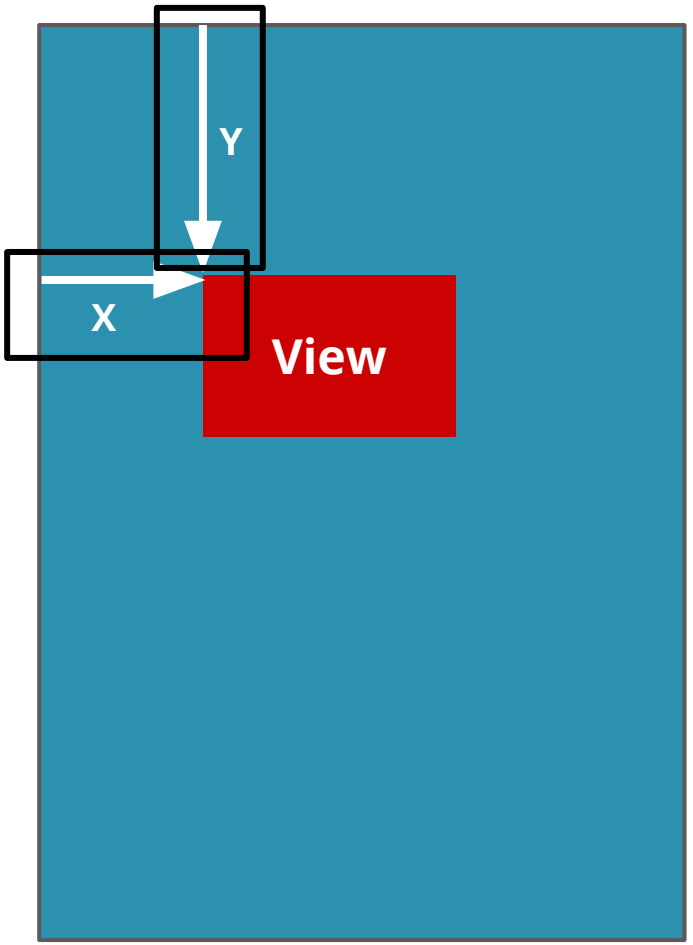
In principio...

era l'**AbsoluteLayout**

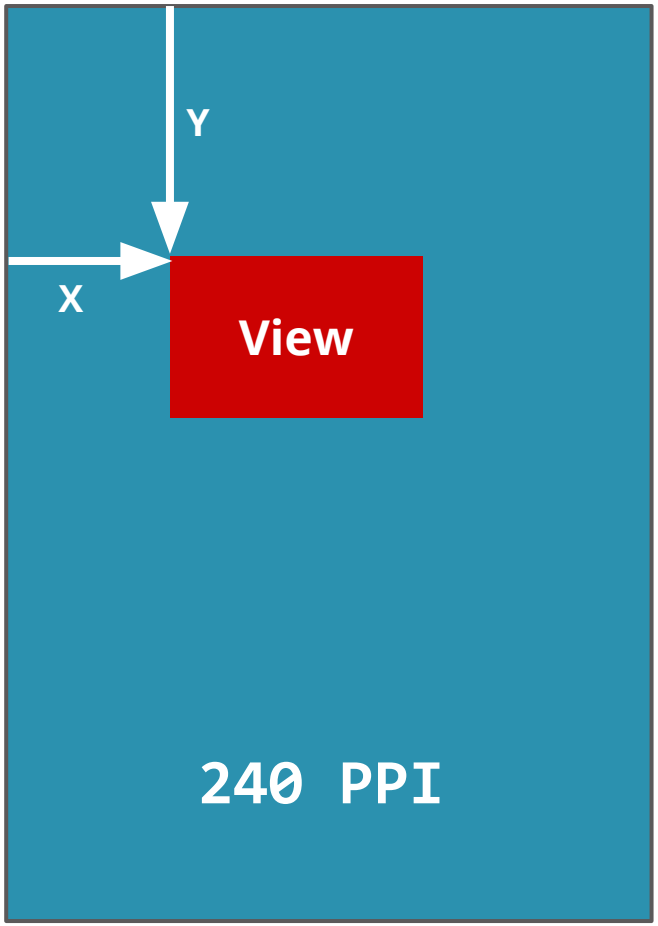
e le dimensioni in **px**

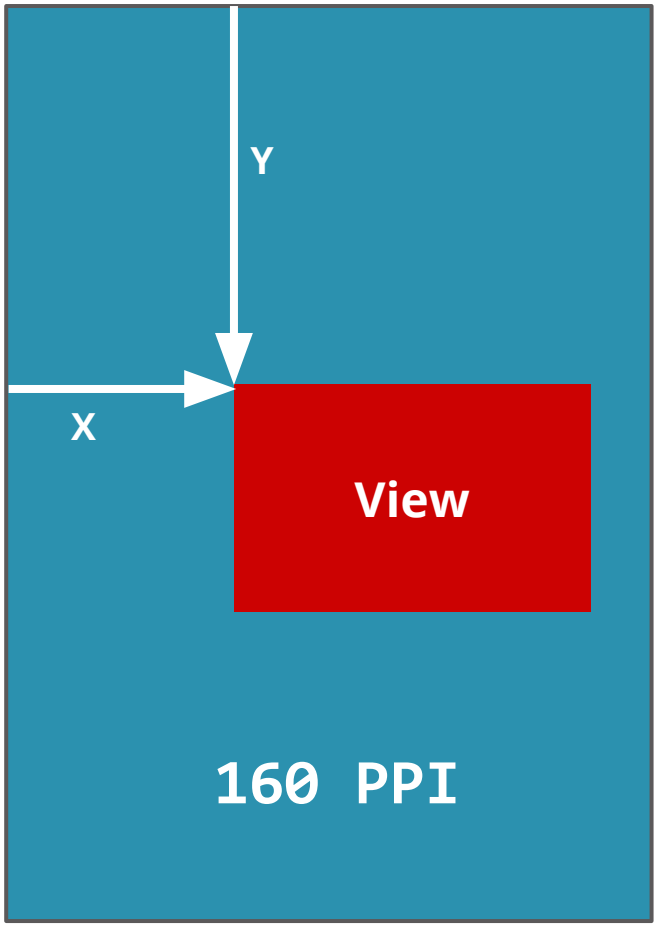
Problemi:

Differenza di densità (DPI/PPI).



6.1.2 Layout - Posizionamento / Dimensione



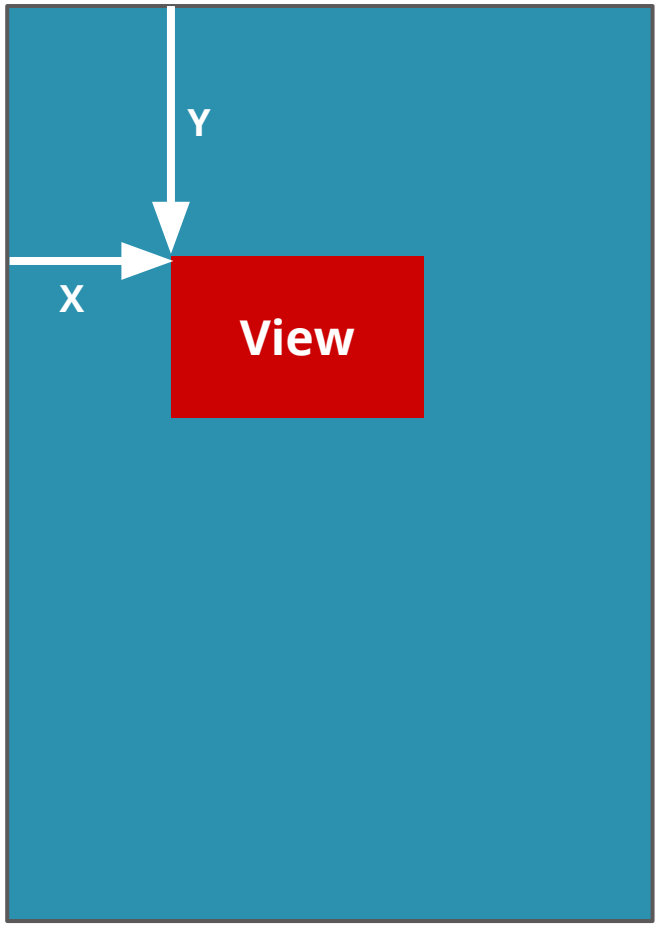


6.1.2 Layout - Posizionamento / Dimensione

In principio...

era l'**AbsoluteLayout**

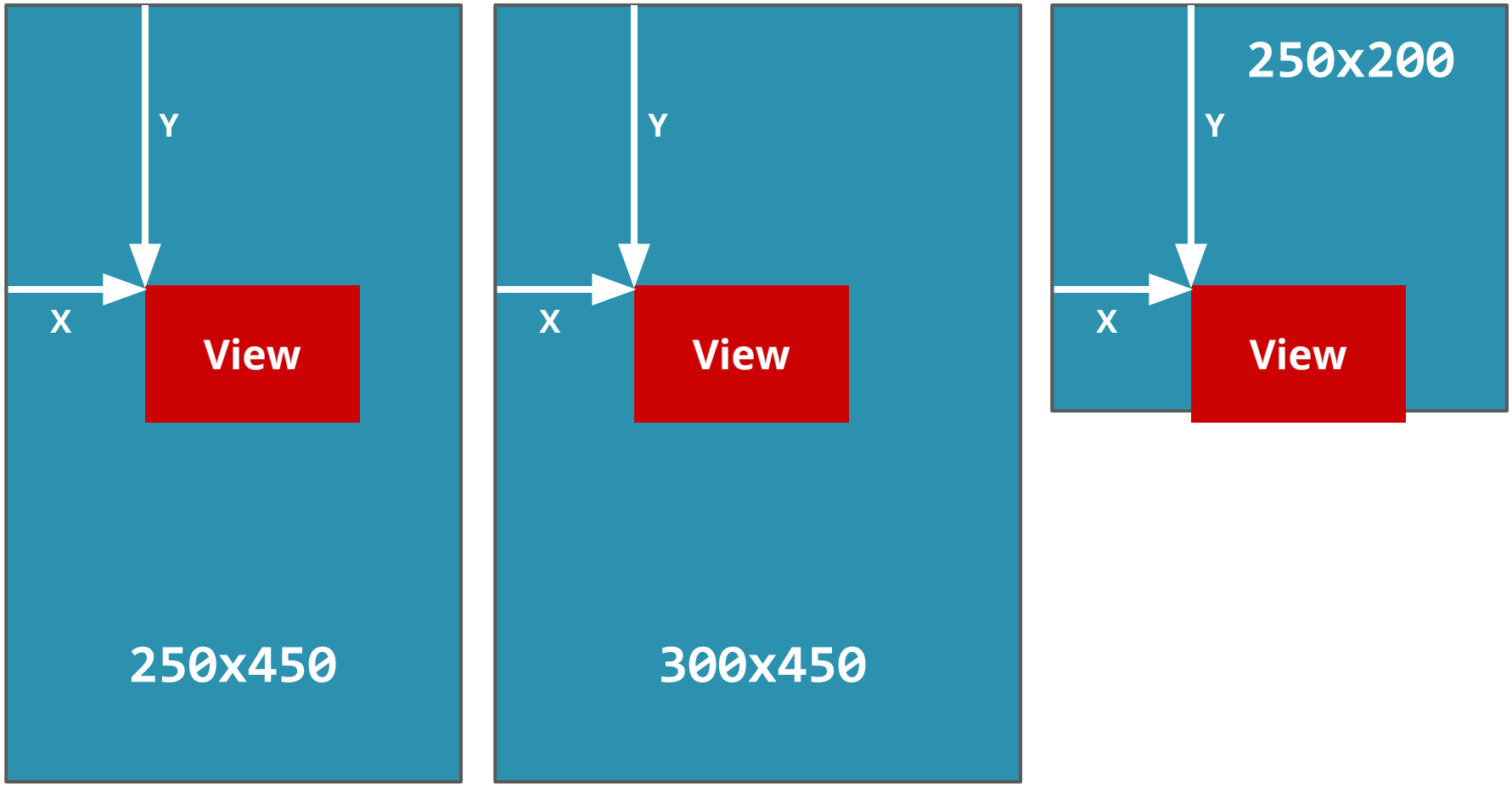
e le dimensioni in **px**



Problemi:

- Differenza di densità (DPI/PPI).
- Differenza di dimensioni (WxH).

6.1.2 Layout - Posizionamento / Dimensione

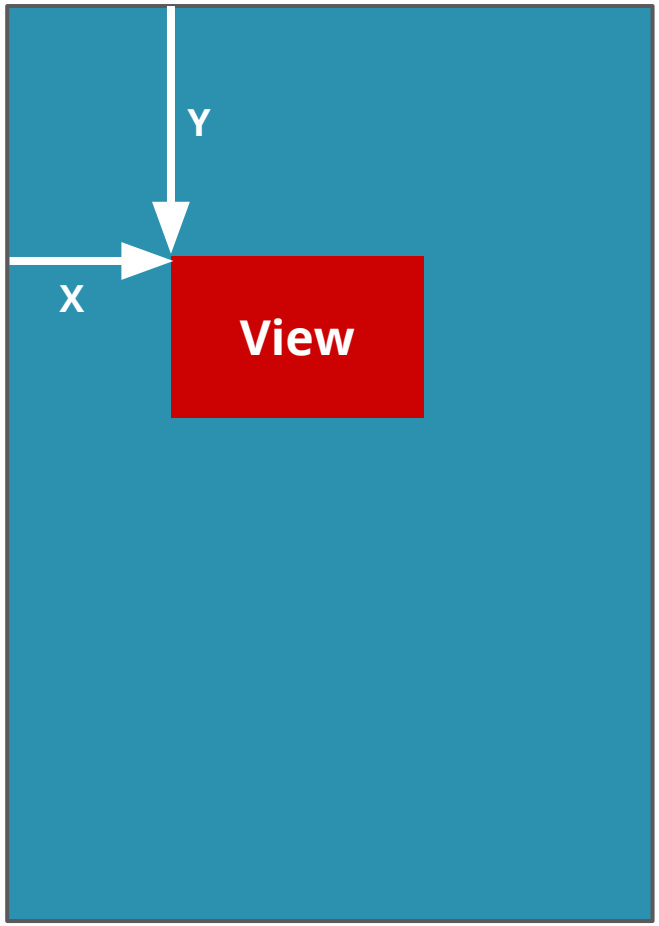


6.1.2 Layout - Posizionamento / Dimensione

In principio...

era l'**AbsoluteLayout**

e le dimensioni in **px**



Problemi:

Frammentazione

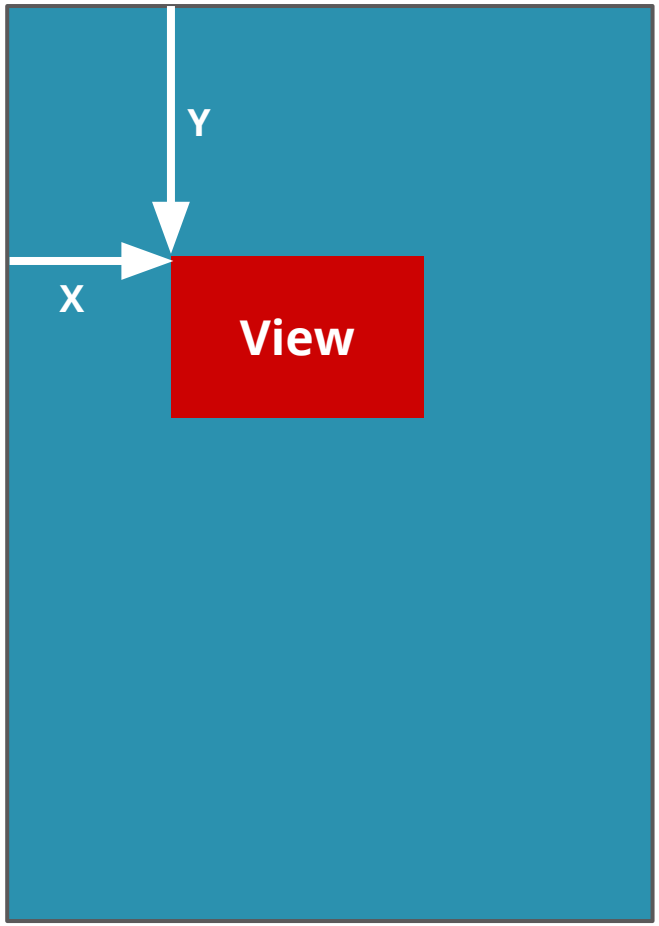
- Differenza di densità (DPI/PPI).
- Differenza di dimensioni (WxH).

6.1.2.1 Unità di misura

In principio...

era l'**AbsoluteLayout**

e le ~~dimensioni in~~ **~~px~~**



Problemi:

Frammentazione

- ~~Differenza di densità (DPI/PPI).~~
- Differenza di dimensioni (WxH).

6.1.2.1 Unità di misura

px Corrispondono a pixel reali sullo schermo.

in Basati sulla dimensione reale dello schermo.

pt Basati sulla dimensione reale dello schermo (1/72 inc).

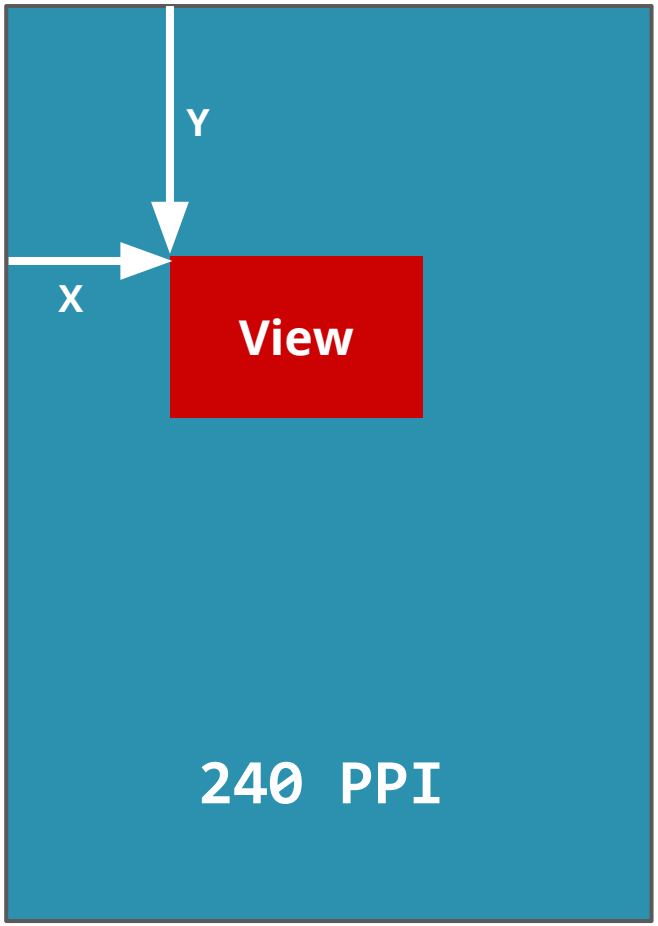
mm Basati sulla dimensione reale dello schermo

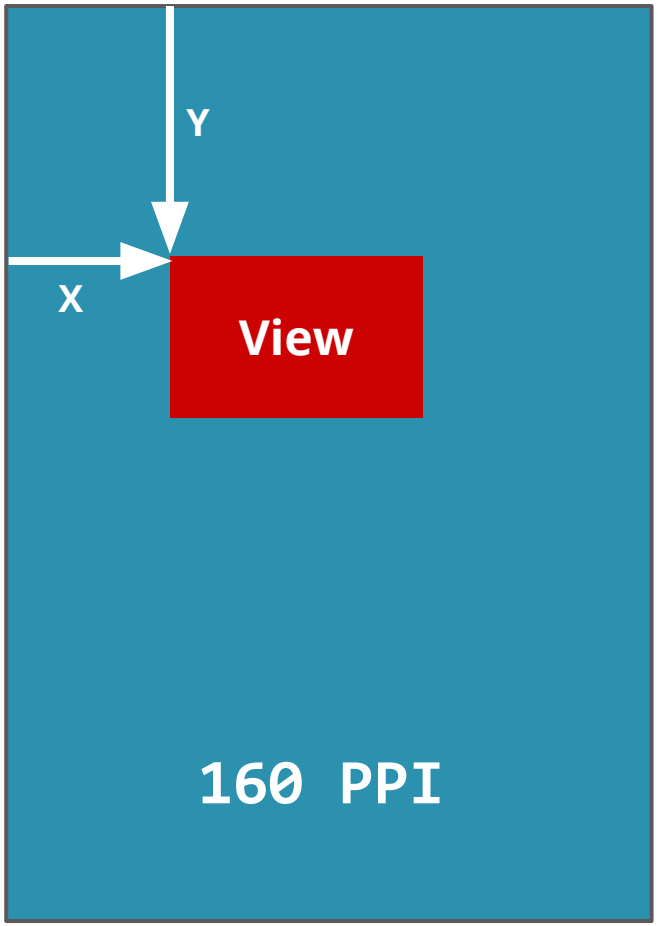
6.1.2.1 Unità di misura

dp (o **dip** Density-independent Pixels)
Un unità astratta, basata sulla effettiva densità dello schermo. Un 1dp a 160dp ~1px reale.

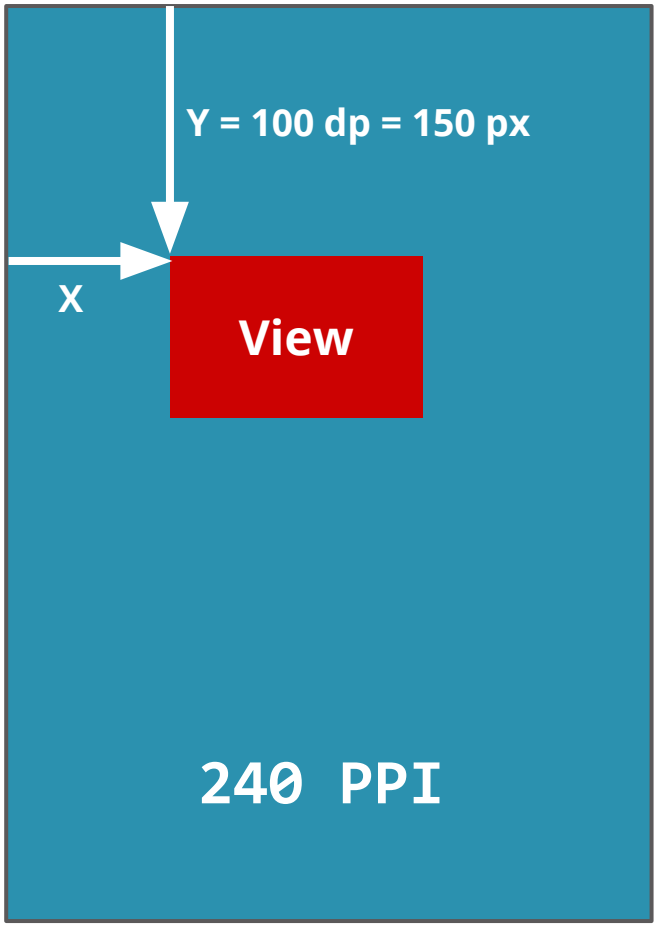
sp (Scale-independent Pixels) Come dp ma scalati in base alle preferenze dell'utente per la dimensione dei font.

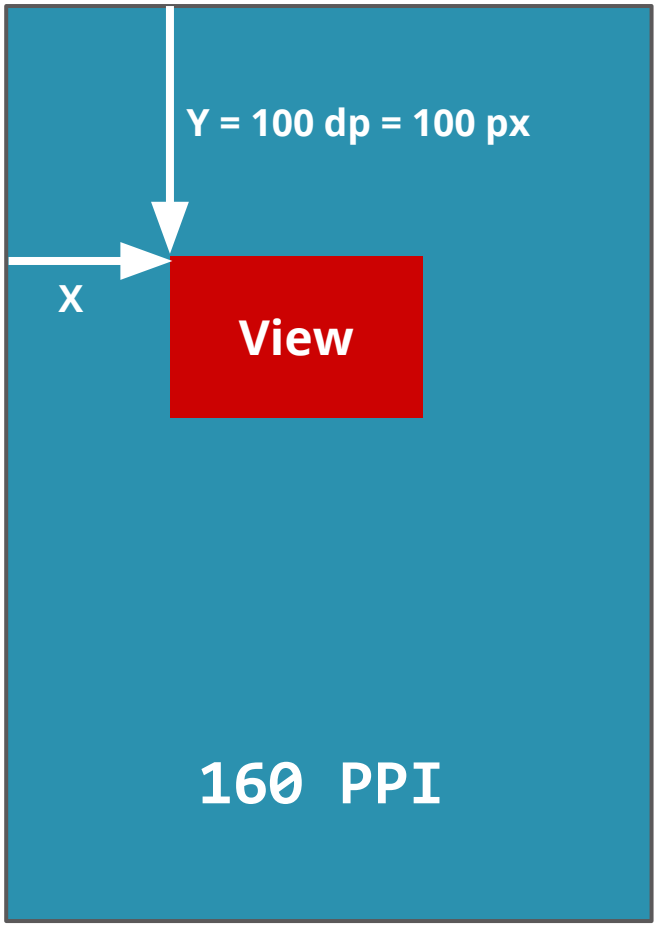
6.1.2.1 Unità di misura





6.1.2.1 Unità di misura



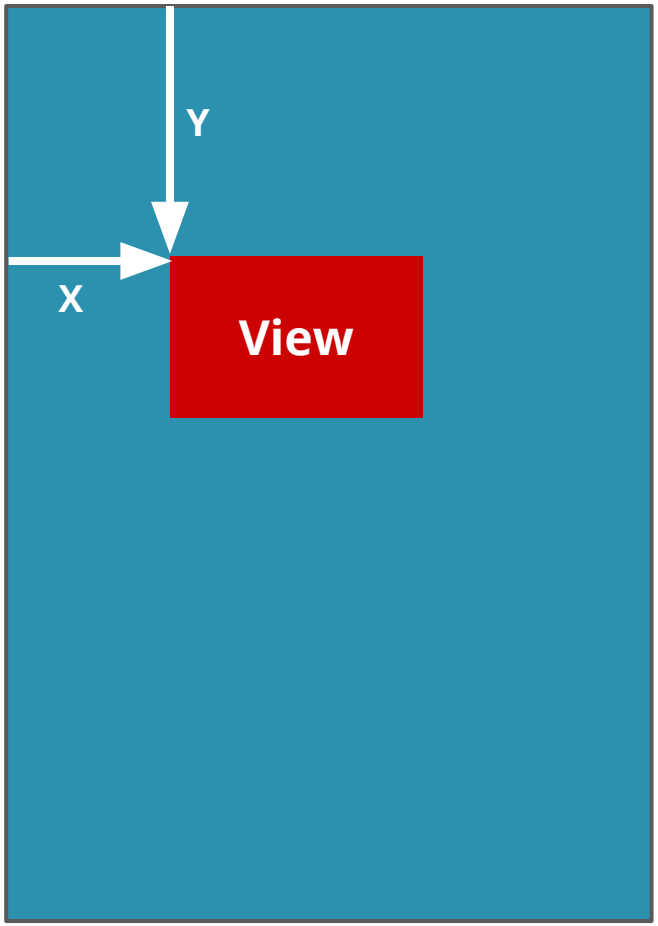


6.1.2.2 Layout

In principio...

era l'~~**AbsoluteLayout**~~

e le ~~dimensioni in **px**~~

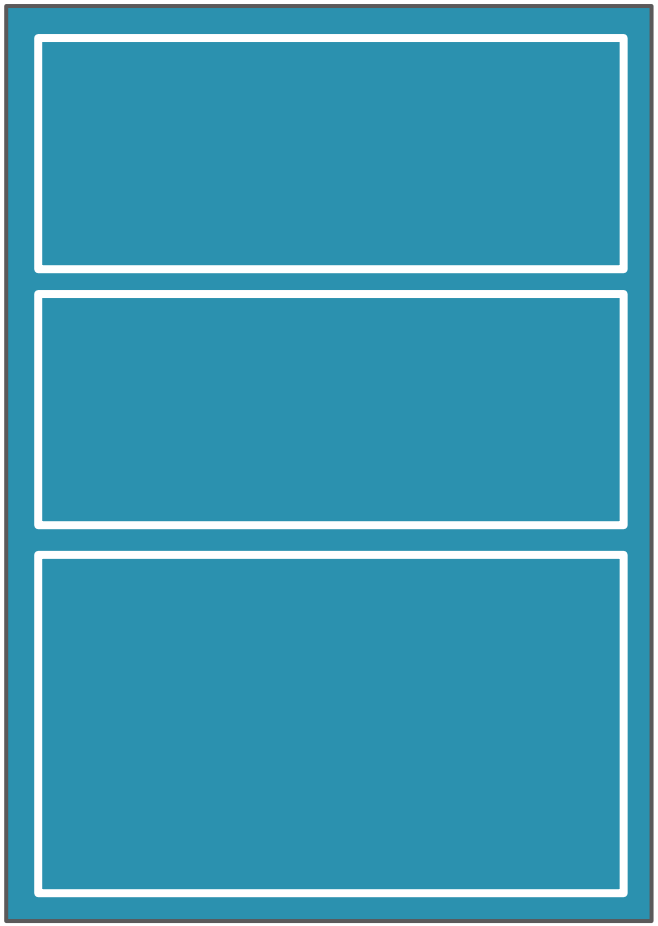


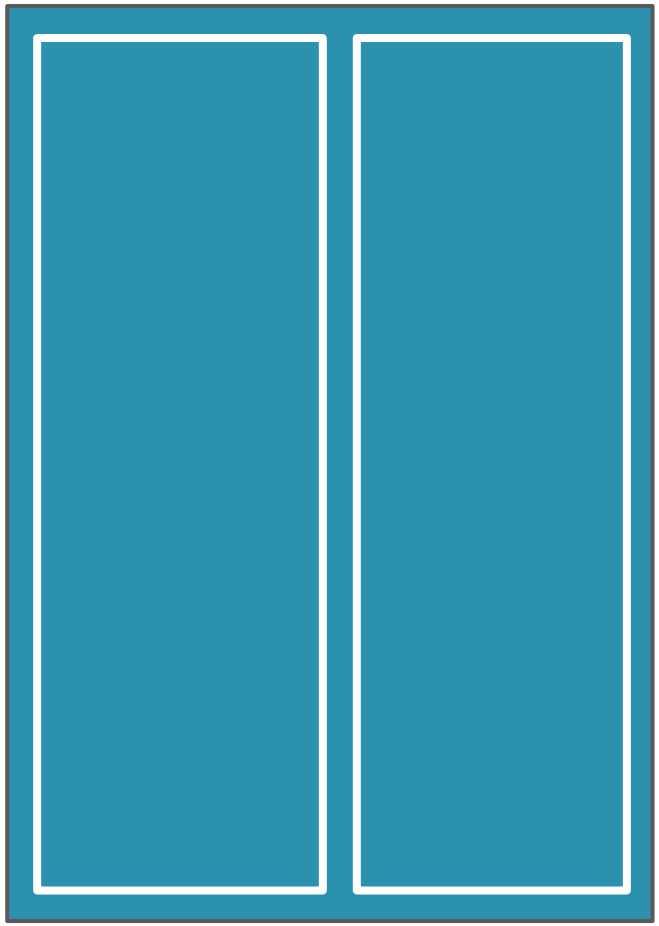
Problemi:

Frammentazione

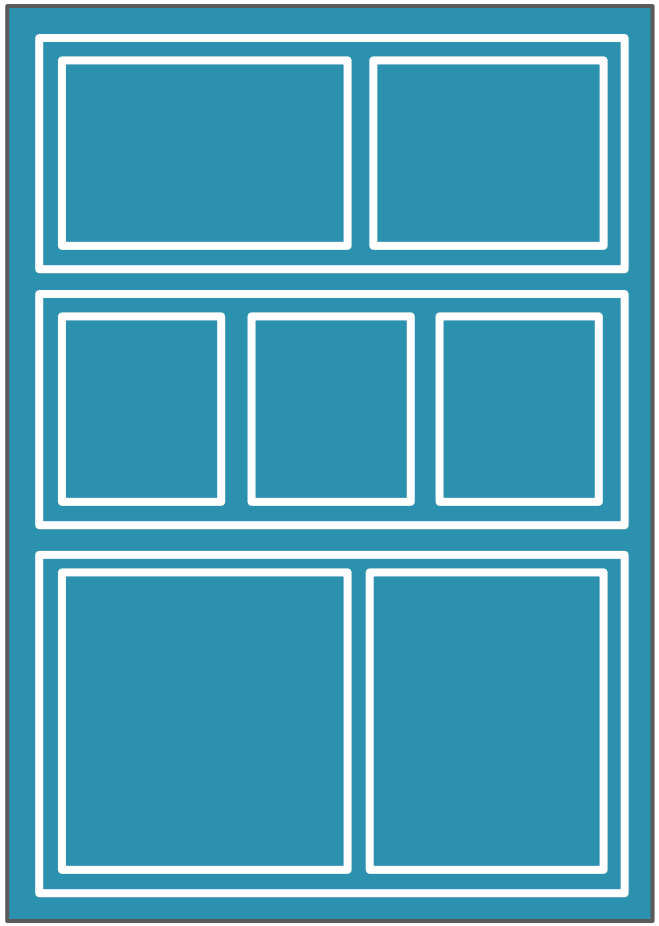
- ~~Differenza di densità (DPI/PPI).~~
- ~~Differenza di dimensioni (WxH).~~

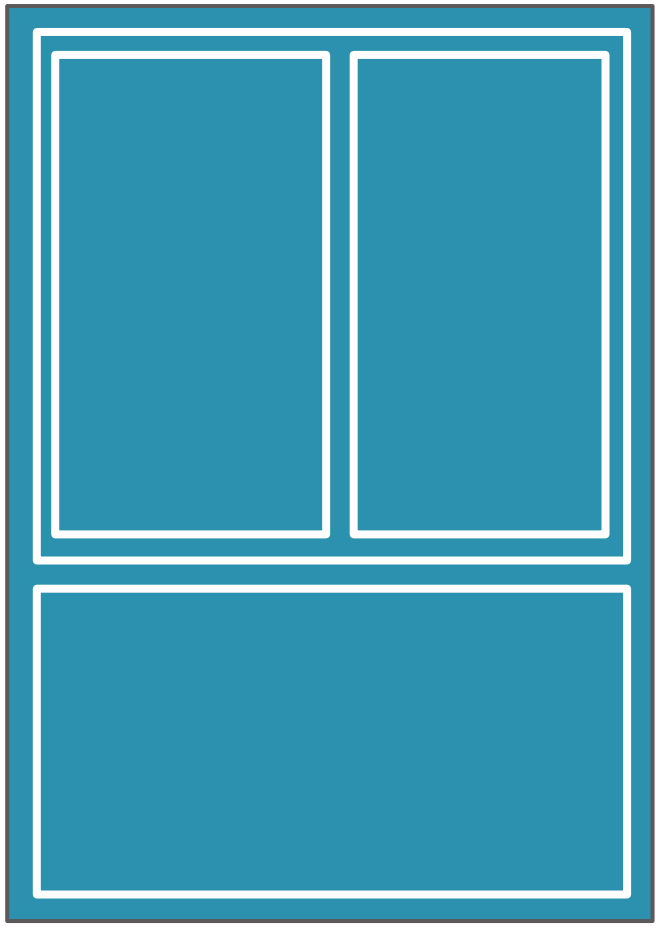
6.1.2.2 LinearLayout



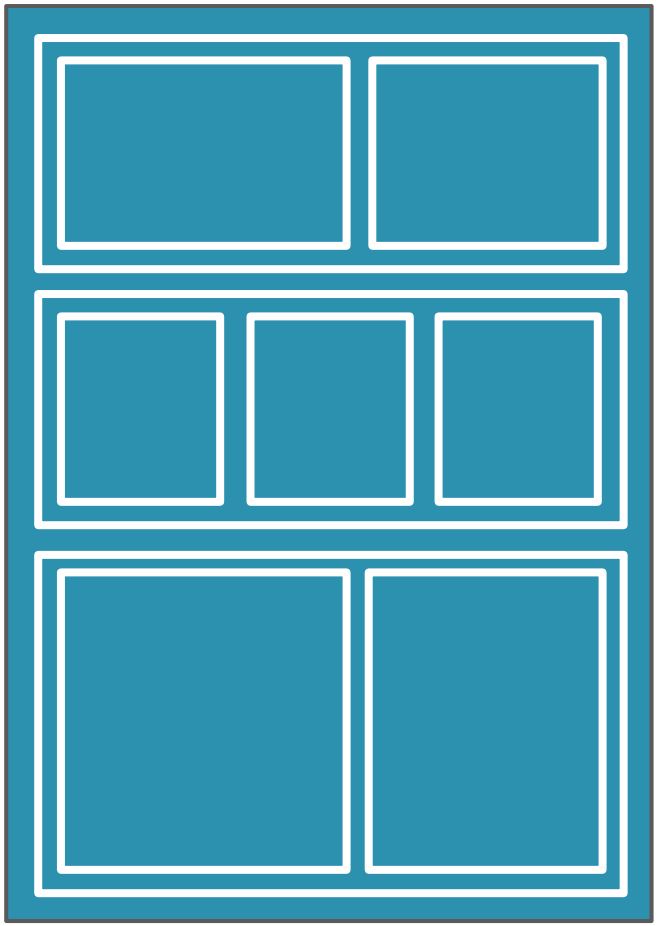


6.1.2.2 LinearLayout

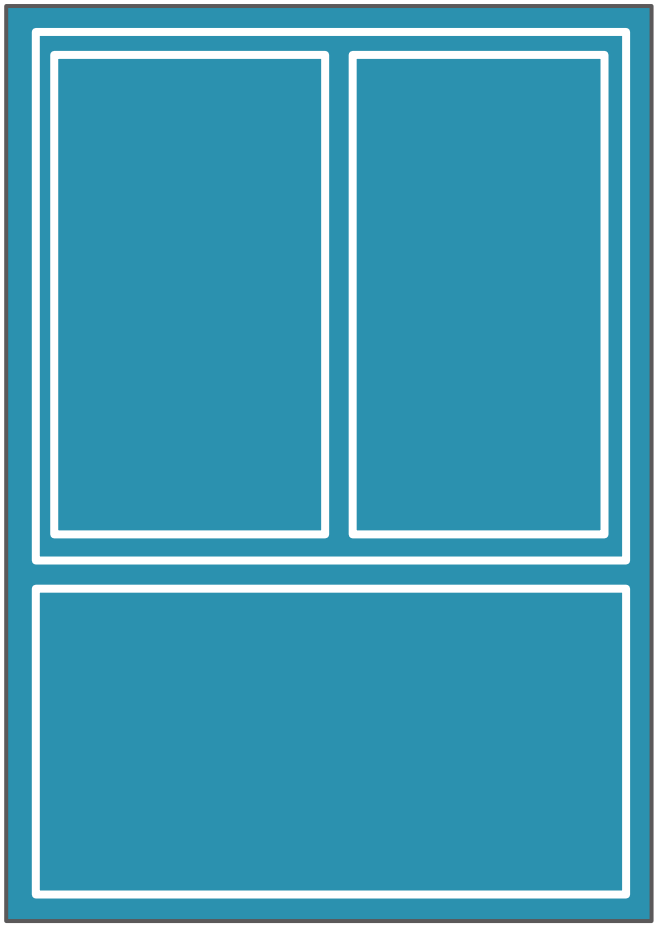


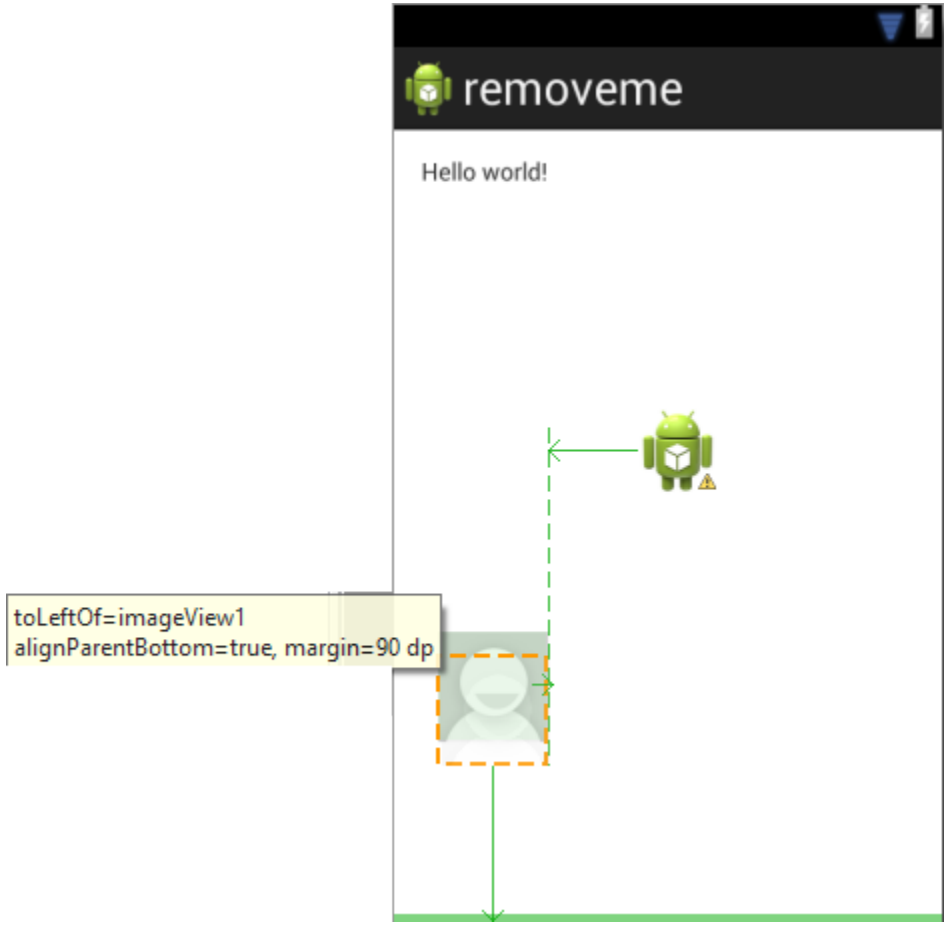


6.1.2.2 TableLayout

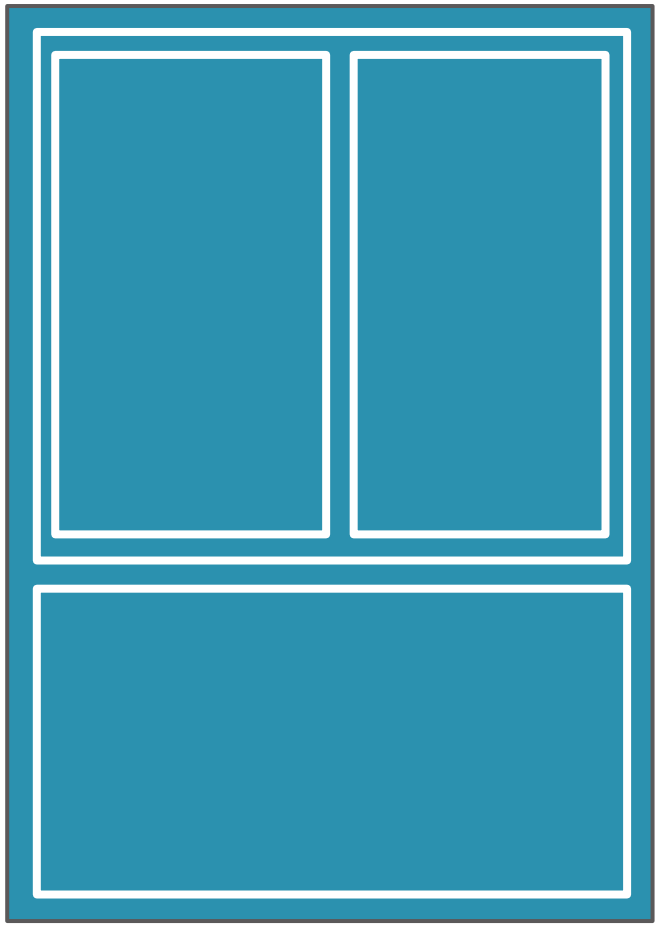


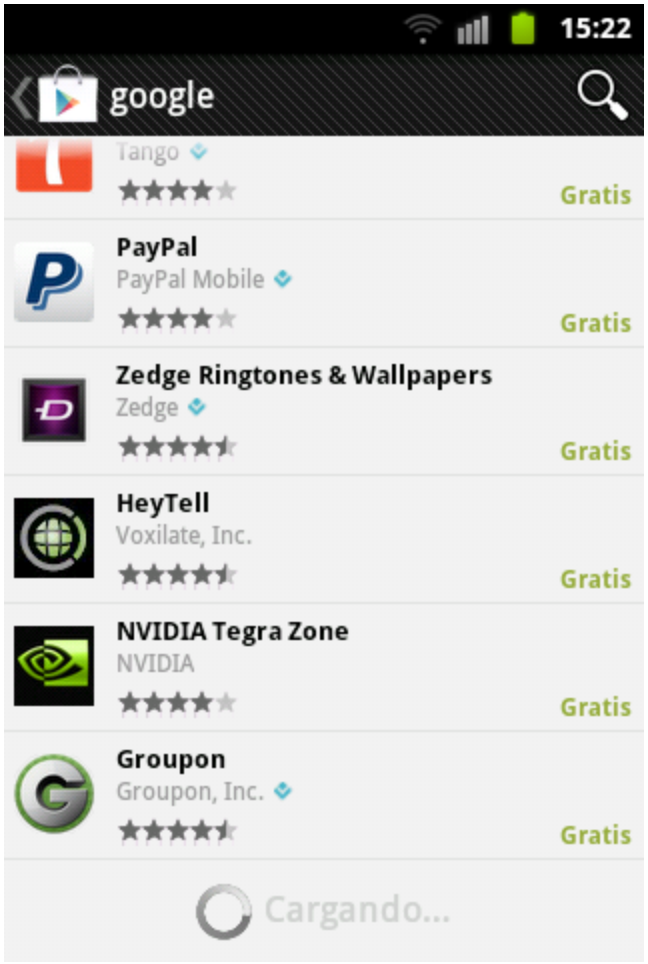
6.1.2.2 RelativeLayout





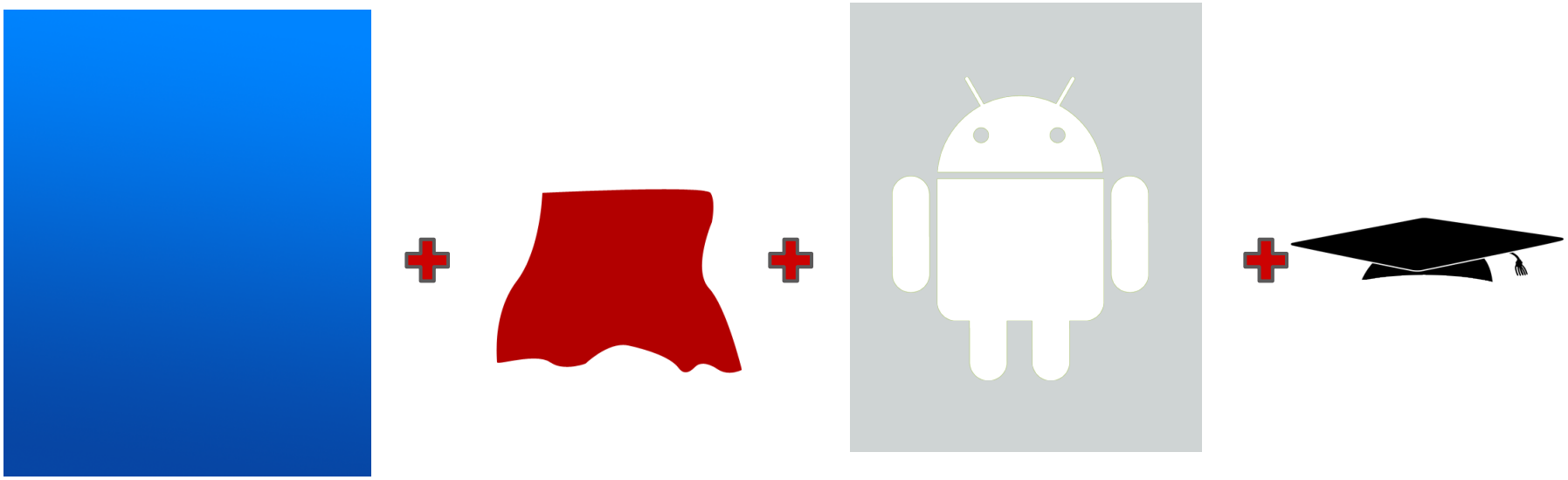
6.1.2.2 RelativeLayout



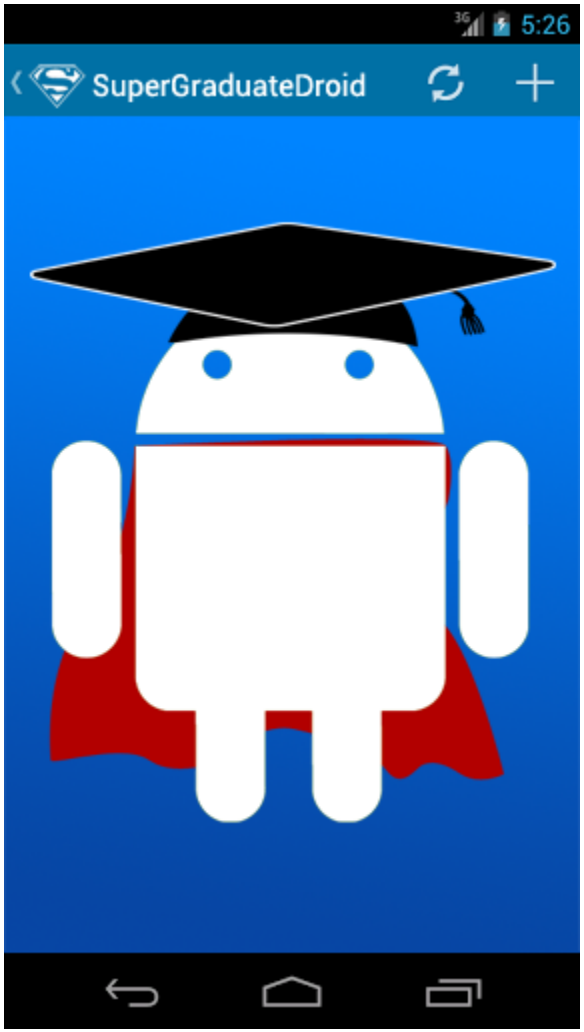


6.1.2.2 FrameLayout

6.1.2.2 FrameLayout

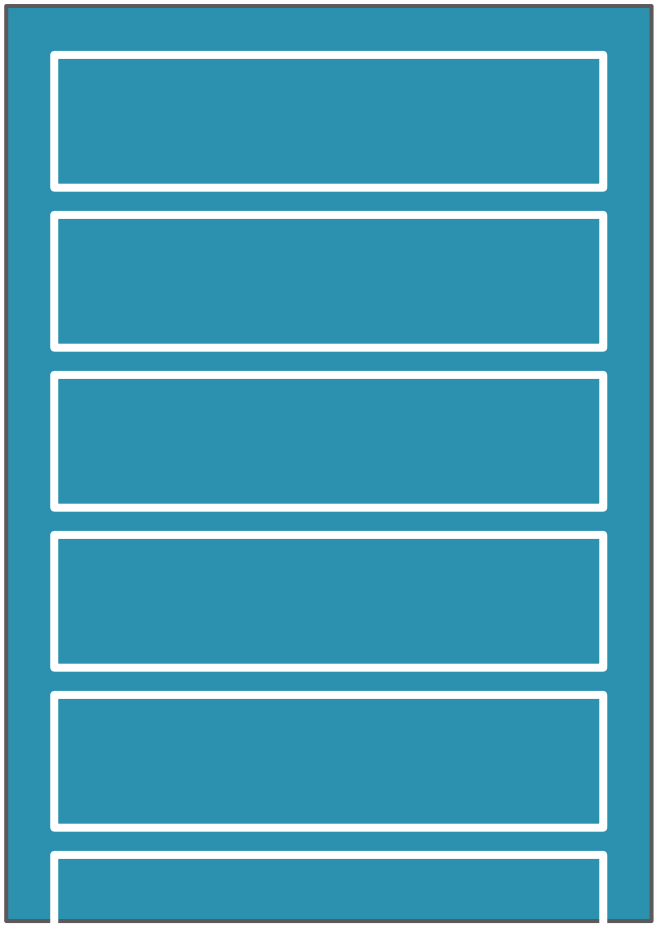


6.1.2.2 FrameLayout

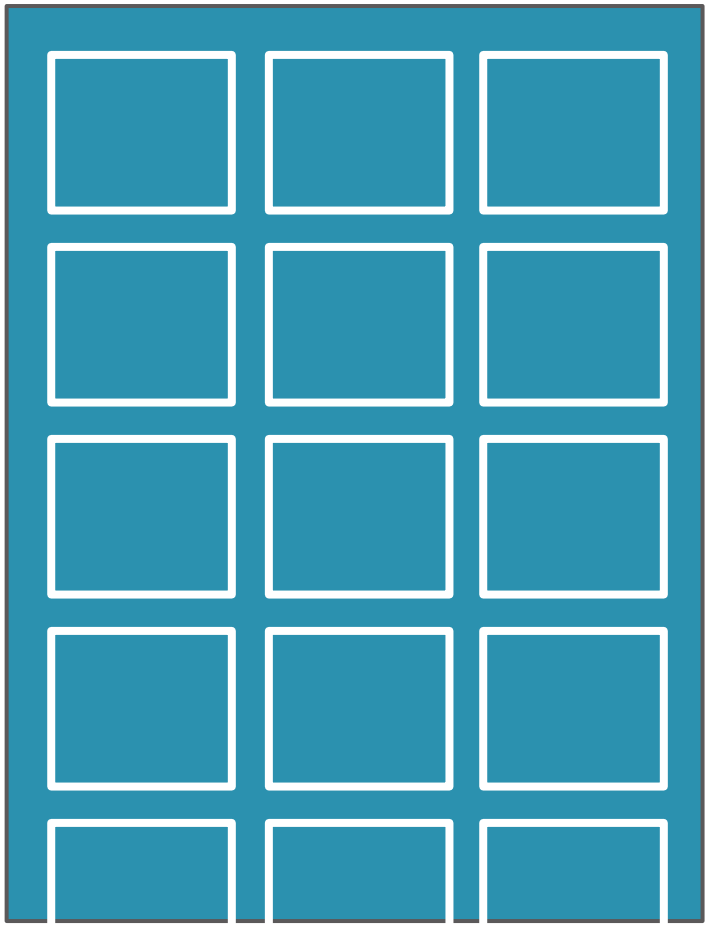


6.1.2.2 FrameLayout

6.1.2.2 ListView



6.1.2.2 GridView

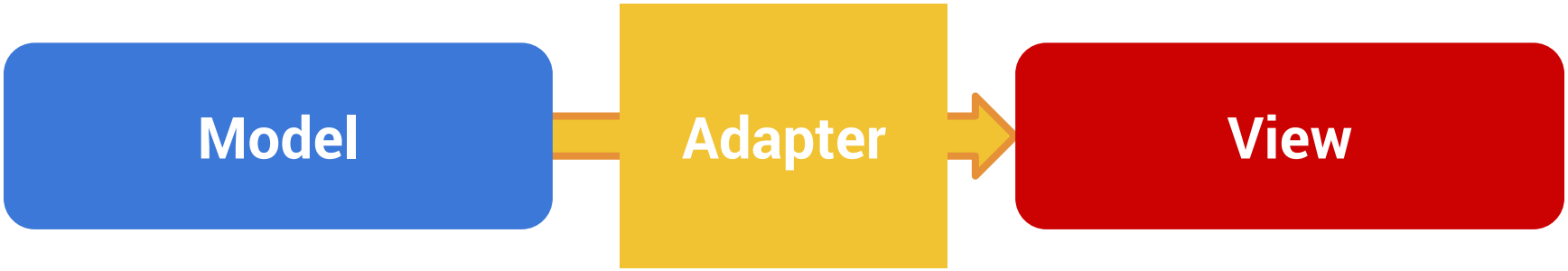


6.1.2.2 Adapter

Classe che traduce un'interfaccia in un'altra.

- Necessariamente parte del Presenter, riduce la purezza raggiungibile in codice UI.

In Android: tramite (di codice) tra dati e UI.

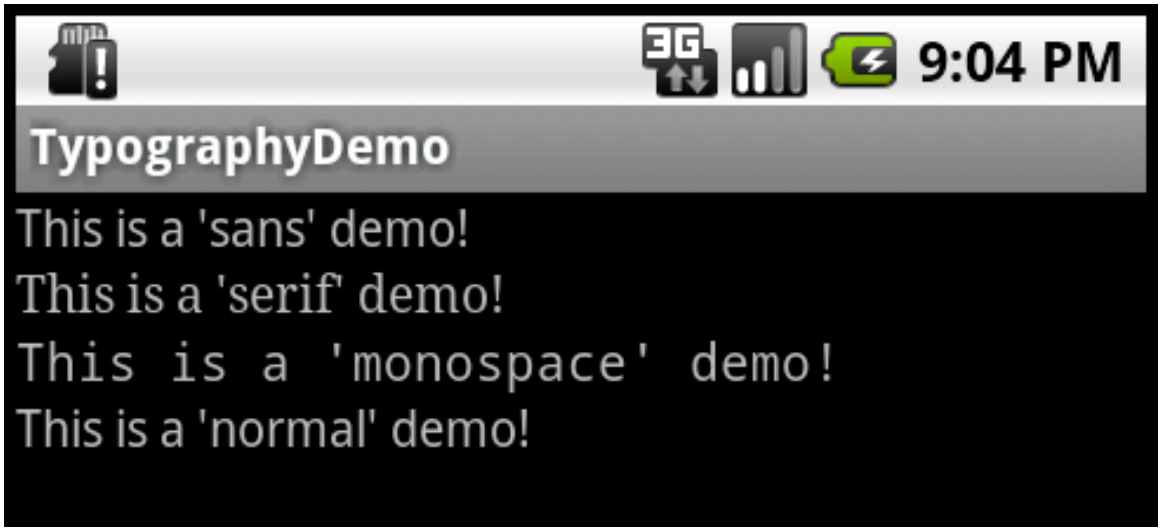


6.2

Widgets

6.2.1 TextView

Componente grafico più semplice: visualizza testo arbitrario.



6.2.1 TextView

Permette comunque una buona flessibilità.

- Auto-gestione dei link presenti nel testo
  - proprietà *android:auto-link*
- Immagini opzionali ai 4 lati
  - proprietà *drawable-** (*top*, *right*, *bottom*, *left*)

6.2.1 TextView

Permette comunque una buona flessibilità.

- Gestione del testo
  - proprietà *android:ellipsize, android:singleLine, android:minLines, android:maxLines*
- Gestione del font
  - proprietà *android:text-color*, *android:text-size*, *android:text-style*, *android:font-family*

6.2.1 TextView

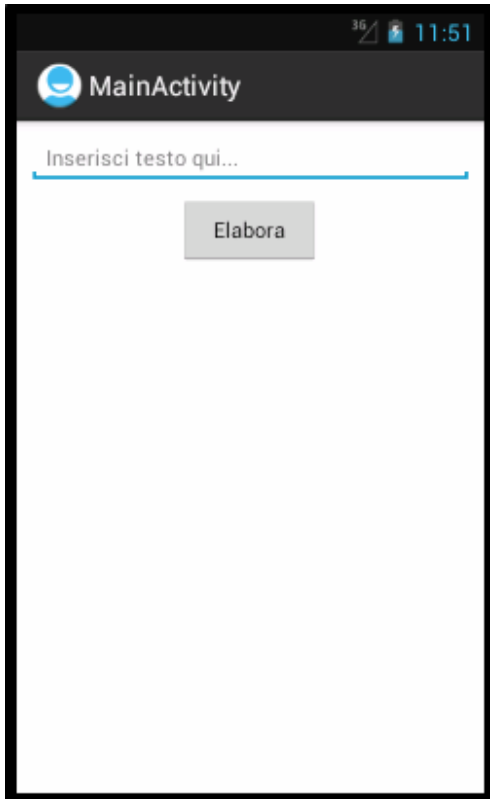
Permette comunque una buona flessibilità.

- Gestione dell'input
  - proprietà *android:editable*, *android:hint*, *android:inputType*

6.2.2 EditText

Sottoclasse di TextView che abilita di default la gestione dell'input:

- *android:hint*="Inserisci testo qui..."
- android:inputType=
  - **text**, **number**, **phone**, ...
  - **textPassword**
  - **textAutoComplete**
  - **textEmailAddress**
  - ...



6.2.2 EditText

Permette di specificare le azioni da associare alla tastiera software:

- *android:imeOptions=*
  - **actionGo**
  - **actionSearch**
  - **actionSend**
  - **actionNone**
  - **actionNext**: default se esitono EditText successive
  - **actionDone**: default se non esistono EditText successive

6.2.2 EditText

Permette di specificare le azioni da associare:

```
EditText editText = (EditText) findViewById(R.id.myEditText);
editText.setOnEditorActionListener(new OnEditorActionListener(){
        public boolean onEditorAction(TextView v, int actionId, KeyEvent evt){
                boolean handled = false;
                if(actionId == EditorInfo.IME_ACTION_SEND){
                        // TODO: inviamo la form
                        handled = true;
                }
                return handled;
        }
});
```

6.2.3 AutoCompleteTextView

Semplice implementare l'autocompletamento tramite la sottoclasse AutoCompleteTextView:

- Specificare un array di stringhe nelle risorse come sorgente dei suggerimenti
- Creare un adapter da questo array di stringhe
- Assegnare programmaticamente l'adapter alla TextView

6.2.3 AutoCompleteTextView

Specificare un array di stringhe nelle risorse come sorgente dei suggerimenti :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="regioni">
        <item>Abruzzo</item>
        <item>Basilicata</item>
        ...
    </string-array>
</resources>
```

6.2.3 AutoCompleteTextView

Creare un adapter da questo array di stringhe:

```
String[] regioni = getResources().getStringArray(R.array.regioni);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
                                                                    regioni);
```

Assegnare l'adapter alla TextView:

```
AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.autocomplete);
textView.setAdapter(adapter);
```

6.2.4 Button

Componente grafico che può avere testo, icona o entrambe e che reagisce alle azioni dell'utente:

6.2.4 Button

Button con solo testo:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/testoButton" />
```

6.2.4 Button

Button con solo icona:

```
<ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/iconaButton" />
```

6.2.4 Button

Button con testo ed icona:

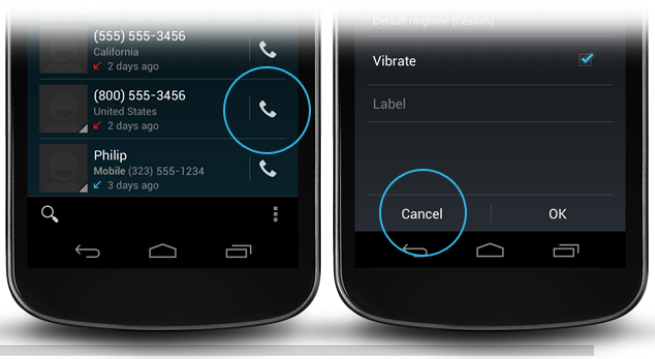
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/testoButton"
    android:drawableLeft="@drawable/iconaButton" />
```

6.2.4 Button

Button senza bordo, utile per raggruppamenti:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/testoButton"
    android:drawableLeft="@drawable/iconaButton"
    style="?android:attr/borderlessButtonStyle" />
```

Disponibile solo da API level 11 (Android 3.0)



6.2.4 Button

Reagire al click dell'utente - xml binding:

XML

```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/testoButton"
        android:onClick="onButtonClick" />
```

JAVA

```
public void onButtonClick(View v){
        // rispondiamo al click
}
```

6.2.4 Button

Reagire al click dell'utente - OnClickListener:

XML

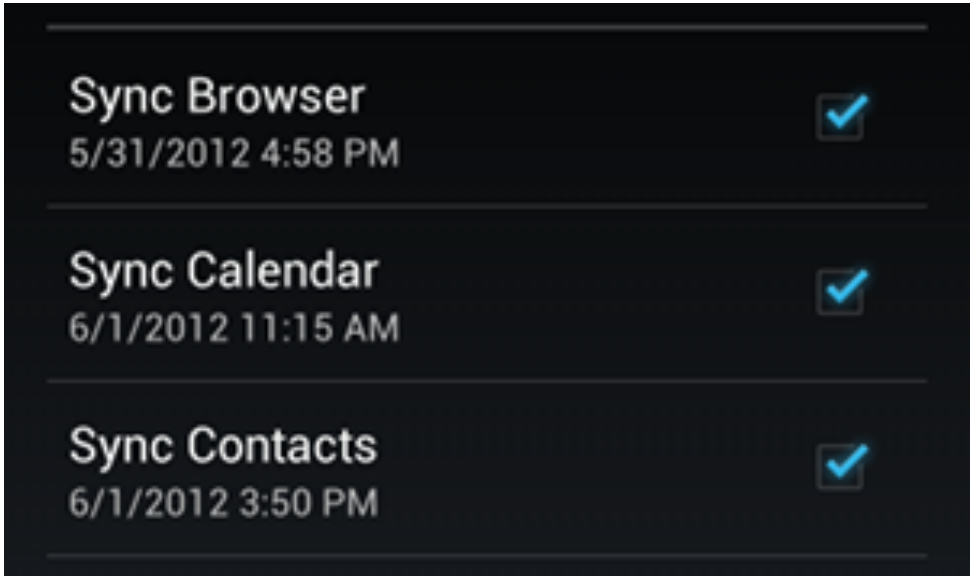
```
<Button
        android:id="@+id/Button" .. />
```

JAVA

```
Button button = (Button) findViewById(R.id.Button);
button.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v){
                // rispondiamo al click
        }
}
```

6.2.5 CheckBox

Componente grafico che permette all'utente di selezionare delle opzioni (tipicamente mostrate in verticale).



6.2.5 CheckBox

Ogni CheckBox è indipendente, quindi gli eventi devono essere gestiti in maniera indipendente.

```
<CheckBox android:id="@+id/cb1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/textCb1" />
```

6.2.5 CheckBox

Gestione click - XML binding:

XML

```
<CheckBox android:id="@+id/cb1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/textCb1"
          android:onClick="onCheckBoxClicked" />
```

JAVA

```
public void onCheckBoxClicked(View v){
    if(v.getId() == R.id.cb1)
        // reagiamo al click
}
```

6.2.5 CheckBox

Gestione click - OnCheckedChangeListener:

XML

```
<CheckBox android:id="@+id/cb1" ... />
```

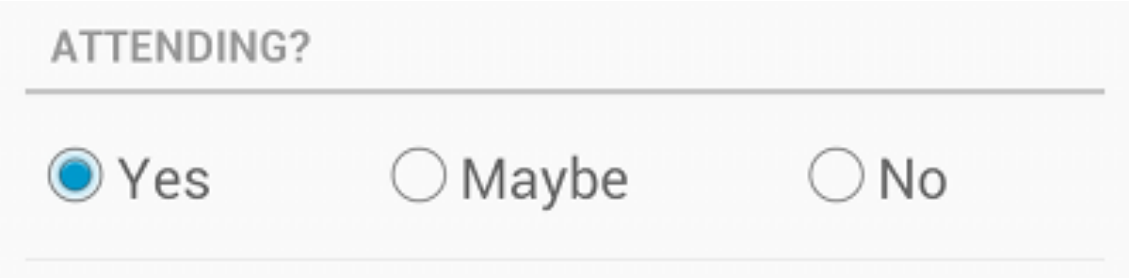
JAVA

```
CheckBox cb = (CheckBox) findViewById(R.id.cb1);
cb.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener(){
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){
                if(buttonView.getId() == R.id.cb1 && isChecked)
                        // reagiamo alla selezione della checkbox in oggetto
        }
});
```

6.2.6 RadioButton

Componente che permette di scegliere delle opzioni mutuamente esclusive:

```
...
<RadioButton android:id="@+id/radio_button"
        android:layout_width="wrap_content",
        android:layout_height="wrap_content"
        android:text="@string/radio_button" />
....
```



6.2.6 RadioButton

Ogni RadioButton deve essere contenuto all'interno di un RadioGroup:

```
<?xml version="1.0" encoding="utf-8" ?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <RadioButton android:id="@+id/radio_button1"
                ... />
        <RadioButton android:id="@+id/radio_button2"
                ... />
</RadioGroup>
```

6.2.6 RadioButton

Il RadioGroup è una sottoclasse di un

LinearLayout di default orientato in verticale:

```
<?xml version="1.0" encoding="utf-8" ?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/myRadioGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        ...
</RadioGroup>
```

6.2.6 RadioButton

Ottenere l'opzione attualmente selezionata:

```
...
Button button = (Button) findViewById(R.id.confirmButton);
button.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View v){
                RadioGroup rg = (RadioGroup) findViewById(R.id.myRadioGroup);
                int viewId = rg.getCheckedRadioButtonId();
                RadioButton  rb = (RadioButton) findViewById(viewId);
                Log.d(TAG, "Opzione selezionata: " + rb.getText());
        }
});
```

6.2.6 RadioButton

Reagire alla selezione di un opzione:

```
...
RadioGroup rg = (RadioGroup) findViewById(R.id.myRadioGroup);
rg.setOnCheckedChangeListener(new OnCheckedChangeListener(){

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId){
        RadioButton  rb = (RadioButton) findViewById(checkedId);
        Log.d(TAG, "Opzione selezionata: " + rb.getText());
    }

});
```

6.2.7 Spinner

Componente che permette di scegliere un elemento da un set: simile al RadioButton per funzionalità, ma più compatto.

```
...
<Spinner
        android:id="@+id/MySpinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
...
```

6.2.7 Spinner

Per creare il set di opzioni come al solito è necessario creare un adapter. Cominciamo dal file delle risorse:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
    <string-array name="myOptions">
        <item>Home</item>
        ...
    </string-array>
</resources>
```

6.2.7 Spinner

E finiamo con l'adapter:

```
Spinner spinner = (Spinner) findViewById(R.id.mySpinner);
// Pattern factory per la creazione dell'adapter.
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.myOptions,
                                                android.R.layout.simple_spinner_item);
// Imposta il layout del vista drop-down.
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

6.2.7 Spinner

Per reagire alla selezione registriamo un listener:

```
Spinner spinner = (Spinner) findViewById(R.id.mySpinner);
spinner.setOnItemSelectedListener(new OnItemSelectedListener(){
        public void onItemSelected(AdapterView<?> parent, View view, int pos, int id){
                String s = (String) parent.getItemAtPosition(pos);
                Log.i(TAG, "Selezionato l'item " + s);
        }
        public void onNothingSelected(AdapterView<?> parent){
        }
});
```

6.2.8 ImageView

Componente che permette di mostrare un'immagine da diverse sorgenti:

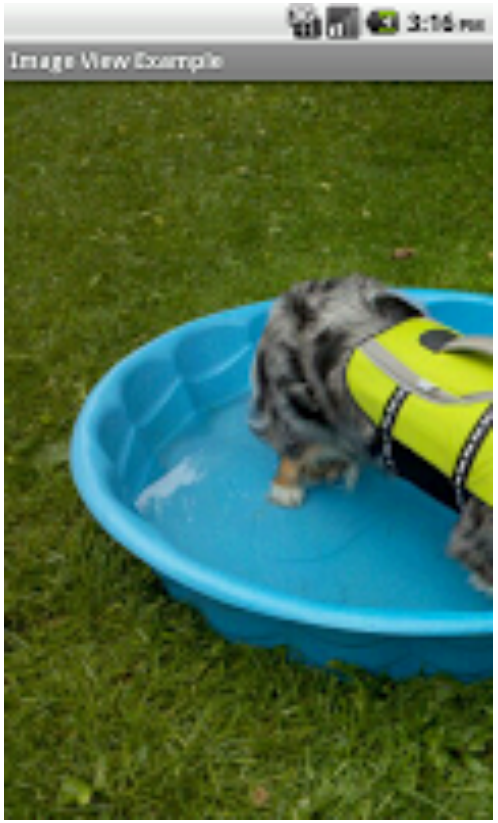
```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/icon" />
```

6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

matrix

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="matrix"
        android:src="@drawable/icon" />
```

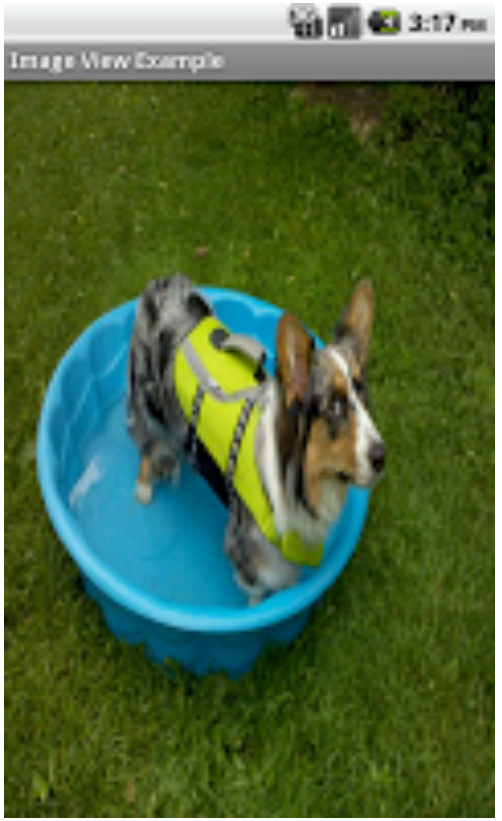


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

fitXY

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitXY"
        android:src="@drawable/icon" />
```

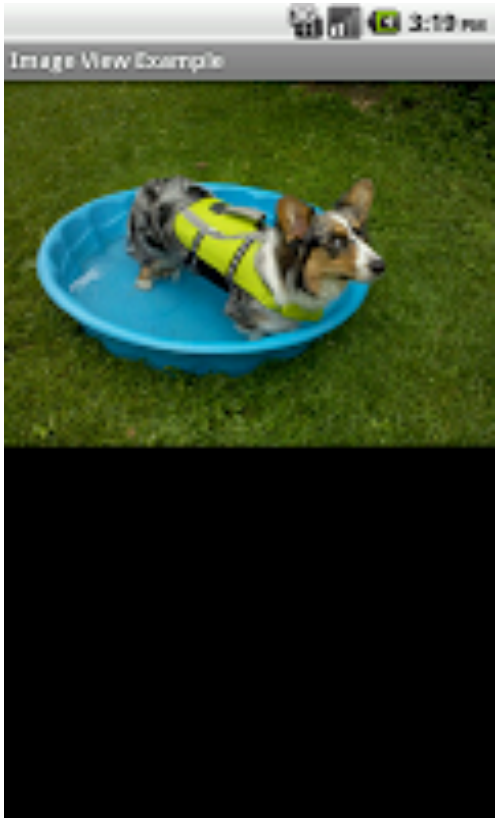


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

fitStart

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitStart"
        android:src="@drawable/icon" />
```

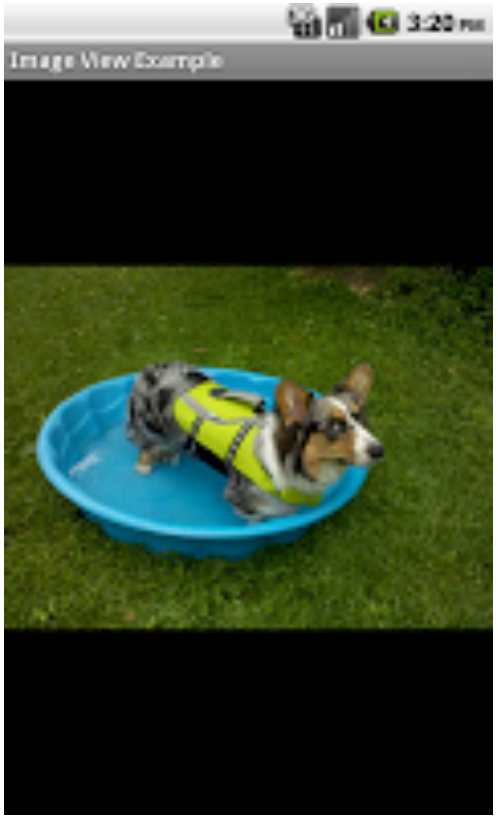


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

fitCenter

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitCenter"
        android:src="@drawable/icon" />
```

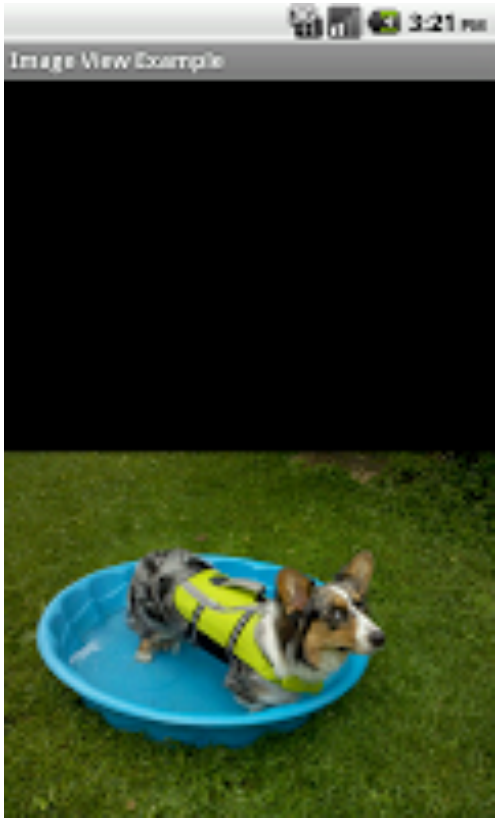


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

fitEnd

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="fitEnd"
        android:src="@drawable/icon" />
```

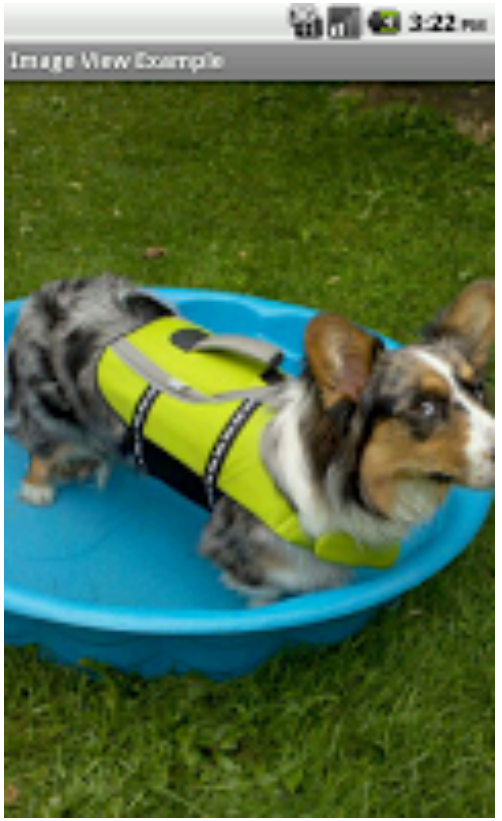


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

center

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="center"
        android:src="@drawable/icon" />
```

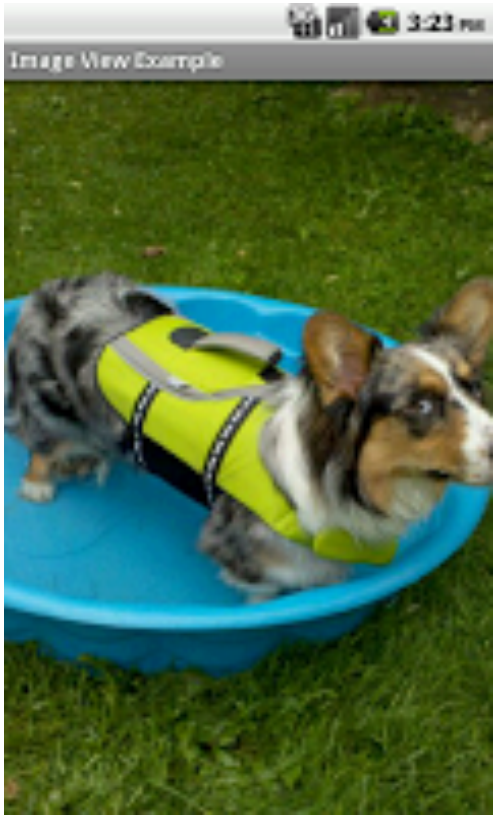


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

centerCrop

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:src="@drawable/icon" />
```

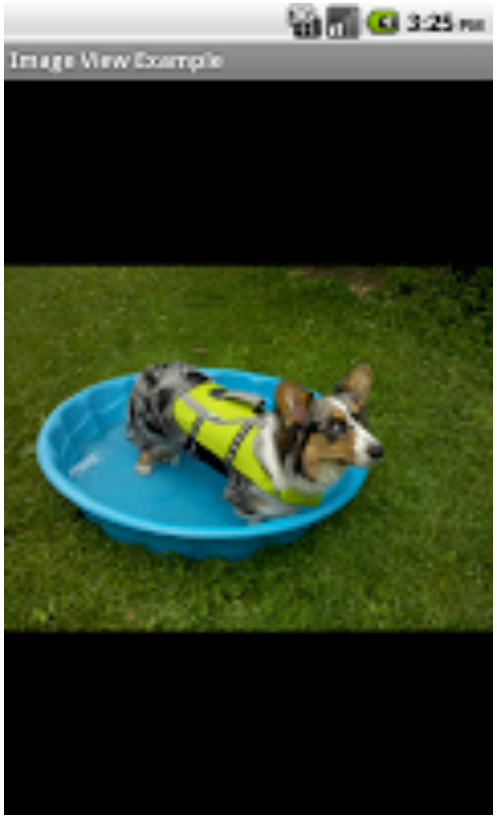


6.2.8 ImageView

Varie opzioni per lo scaling dell'immagine:

centerInside

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerInside"
        android:src="@drawable/icon" />
```



6.2.8 ImageView

Opzione per il resize automatico per mantenere l'aspect-ratio dell'immagine:

```
<ImageView
        android:id="@+id/myImageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:src="@drawable/icon" />
```

6.2.8 ImageView

Tramite Java esistono metodi che accettano diversi tipi di immagini renderizzabili:

DRAWABLE

```
..
ImageView iv = (ImageView) findViewById(R.id.myImageView);
Drawable d = getResources().getDrawable(R.drawable.icon);
iv.setImageDrawable(d);
...
```

6.2.8 ImageView

Tramite Java esistono metodi che accettano diversi tipi di immagini renderizzabili:

BITMAP

```
..
ImageView iv = (ImageView) findViewById(R.id.myImageView);
Bitmap b = BitmapFactory.decodeFile("icon.png");
iv.setImageBitmap(b);
...
```

6.2.8 ImageView

Tramite Java esistono metodi che accettano diversi tipi di immagini renderizzabili:

RESOURCE

```
..
ImageView iv = (ImageView) findViewById(R.id.myImageView);
iv.setImageResource(R.drawable.icon);
...
```

6.2.8 ImageView

Tramite Java esistono metodi che accettano diversi tipi di immagini renderizzabili:

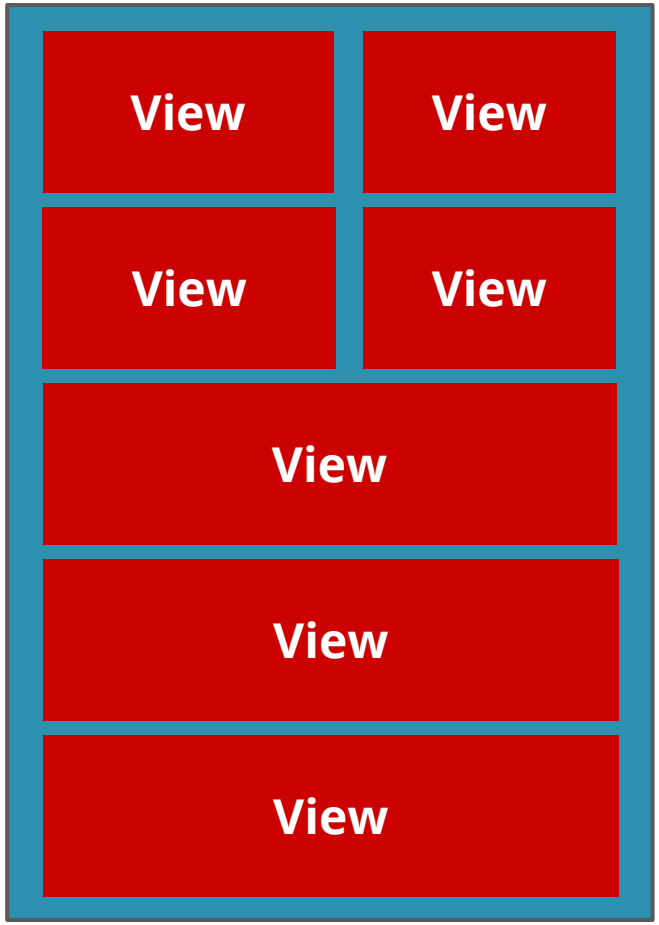
URI

```
..
ImageView iv = (ImageView) findViewById(R.id.myImageView);
iv.setImageURI(R.drawable.icon);
...
```

6.3

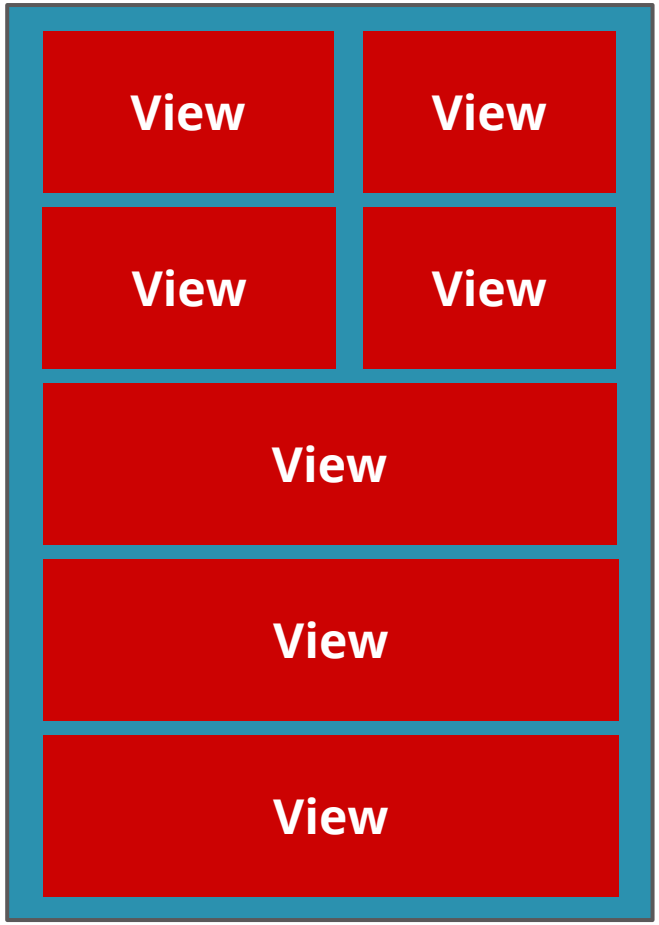
Style, Theme, State-Color, ecc...

6.3.1 Style/Theme




```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomText" parent="@style/Text">
      <item name="android:textSize">20sp</item>
      <item name="android:textColor">#008</item>
   </style>
</resources>
```

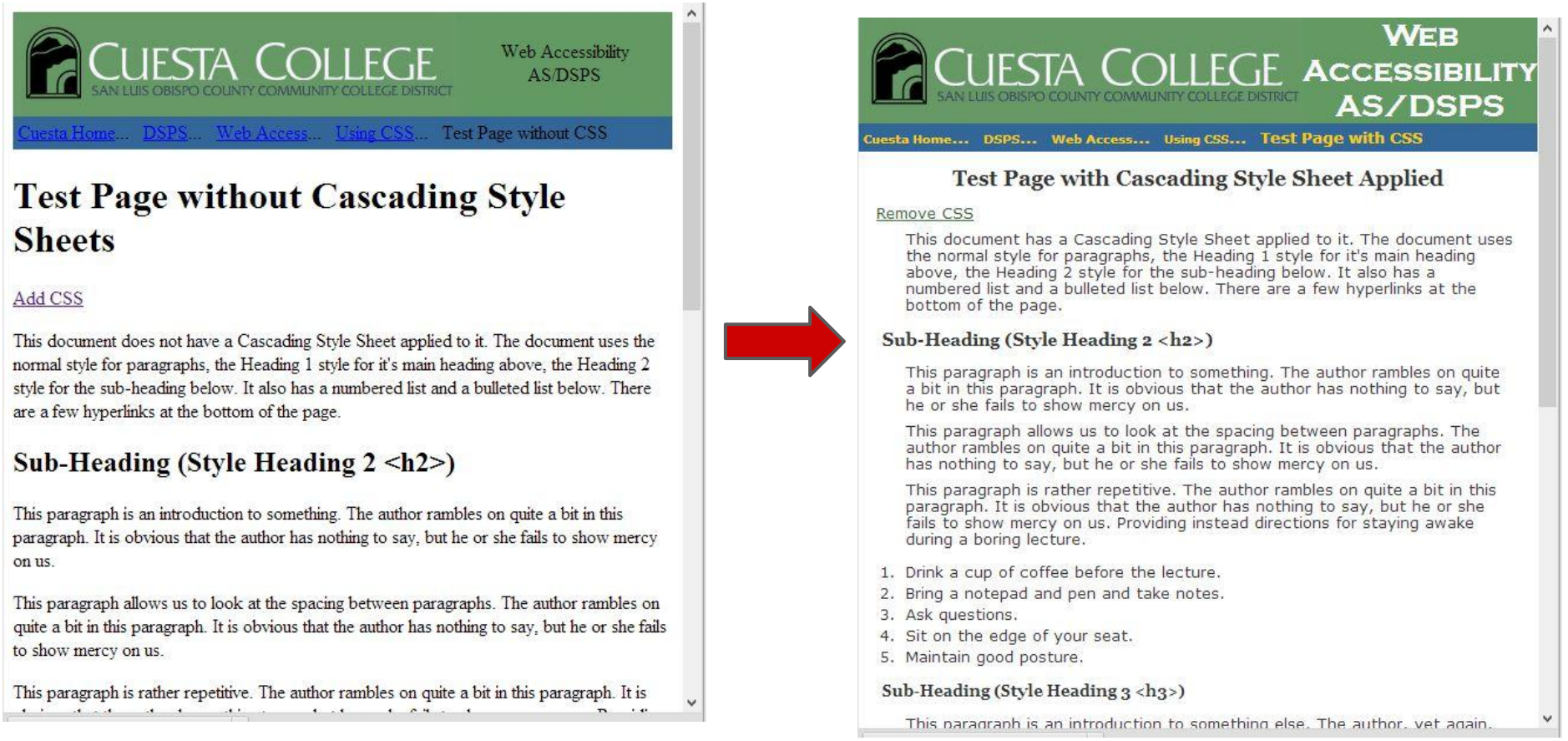
6.3.1 Style/Theme



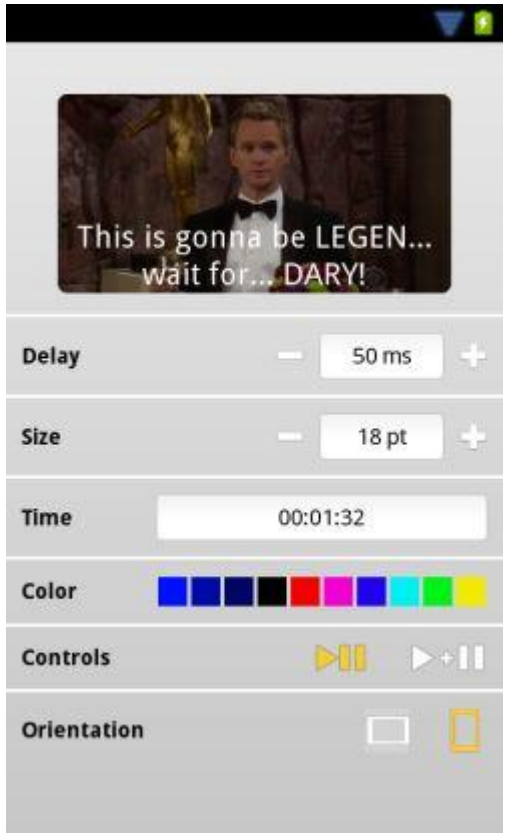

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomText" parent="@style/Text">
      <item name="android:textSize">20sp</item>
      <item name="android:textColor">#008</item>
   </style>
</resources>
```

HTML **CSS**

6.3.1 Style/Theme

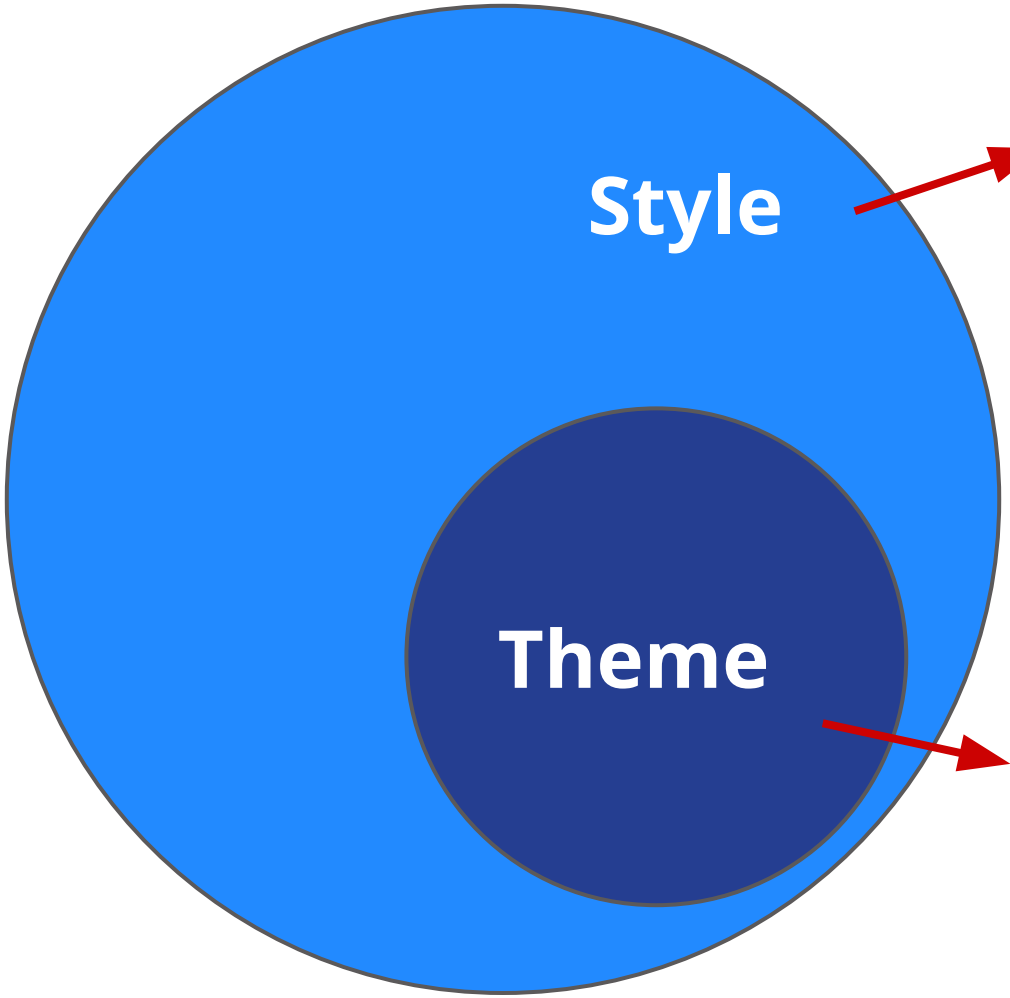


6.3.1 Style/Theme



6.3.1 Style/Theme

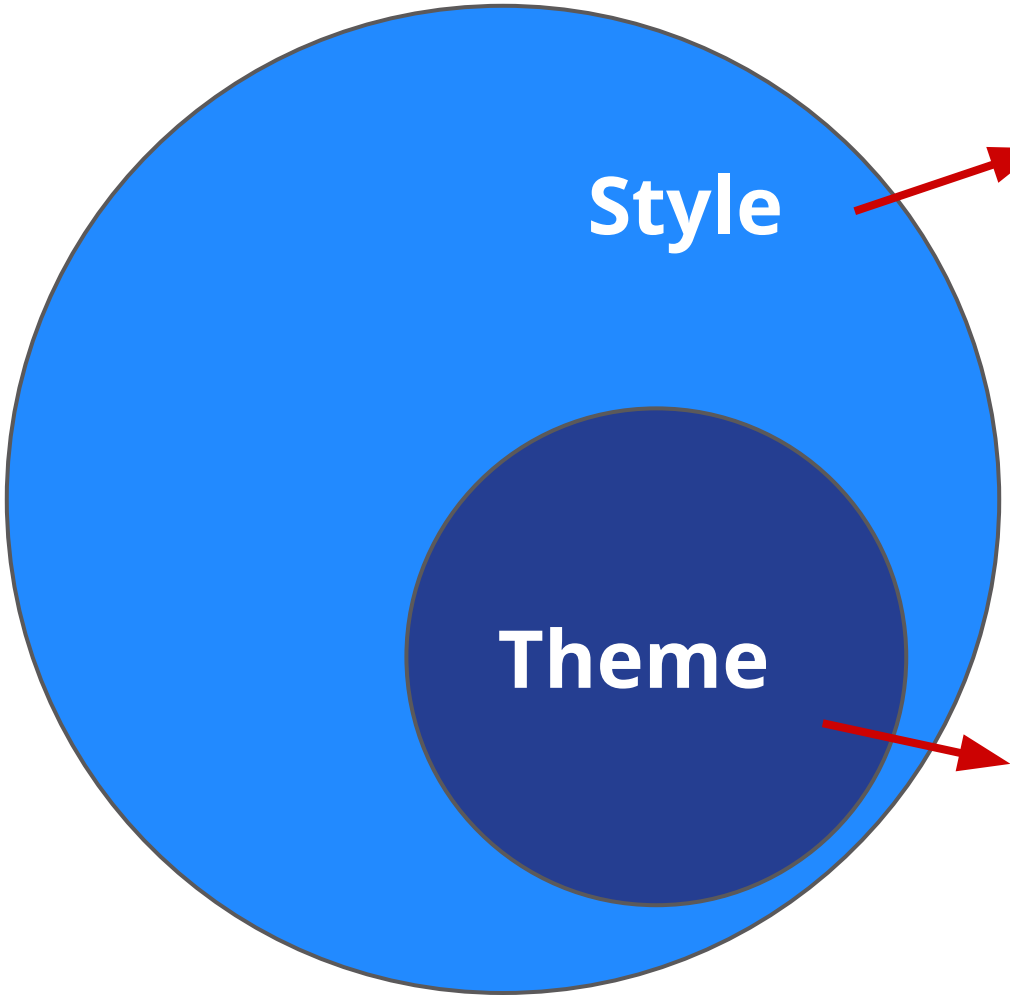
6.3.1 Style/Theme



Applicabile ad ogni elemento

Applicabile ad Activity e/o Applicazioni

6.3.1 Style/Theme



Applicabile ad ogni elemento

Applicabile ad Activity e/o Applicazioni

6.3.1 Style usage

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CustomText" parent="@style/Text">
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">#008</item>
    </style>
</resources>
```

6.3.1 Style usage

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CustomText" >
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">#008</item>
    </style>
</resources>
```

6.3.1 Style usage

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CustomText" >
        <item name="android:textSize">40sp</item>
        <item name="android:textColor">#008</item>
    </style>
    <style name="CustomText.Big" >
        <item name="android:textSize">100sp</item>
    </style>
</resources>
```

6.3.1 Style usage

```
...
<EditText
    style="@style/CustomText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello, World!" />
...
```

6.3.1 Theme usage

```
<style name="CustomTheme" parent="android:Theme.Light">
    <item name="android:windowBackground">
        @color/custom_theme_color
    </item>
    <item name="android:colorBackground">
        @color/custom_theme_color
    </item>
</style>
```

6.3.1 Theme usage

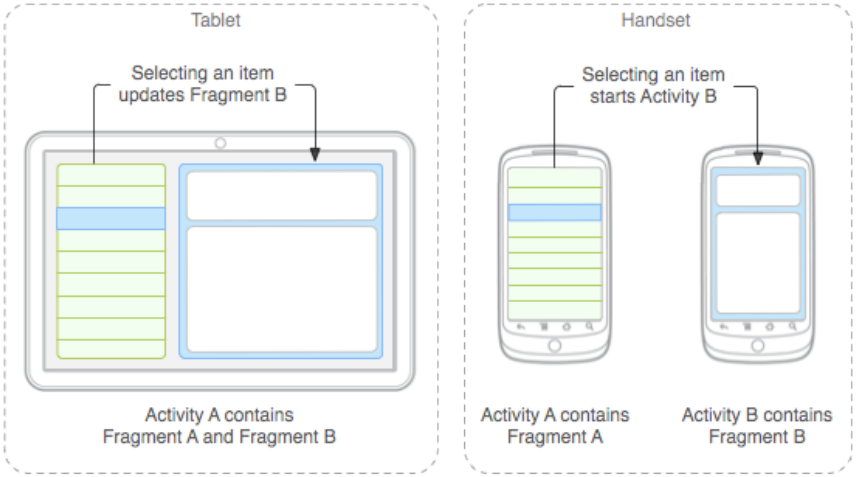
```
...
<activity android:theme="@style/CustomTheme">
...
<activity android:theme="@android:style/Theme.Translucent">
...
<activity android:theme="@android:Theme.Holo.DarkActionBar"
>
...
```

6.5

Fragments

6.5 Fragments

Rappresentano una parte di interfaccia grafica con annessa logica di gestione per semplificare lo sviluppo, aumentare il riciclo di codice e unificare la gestione smartphone/tablet.



6.5 Fragments

Introdotti da Android 3.0 e migliorati con le successive versioni, sono utilizzabili anche nelle versioni precedenti di Android con una libreria di supporto che affronteremo nel modulo di programmazione avanzata.

6.5 Fragments - comunicazione

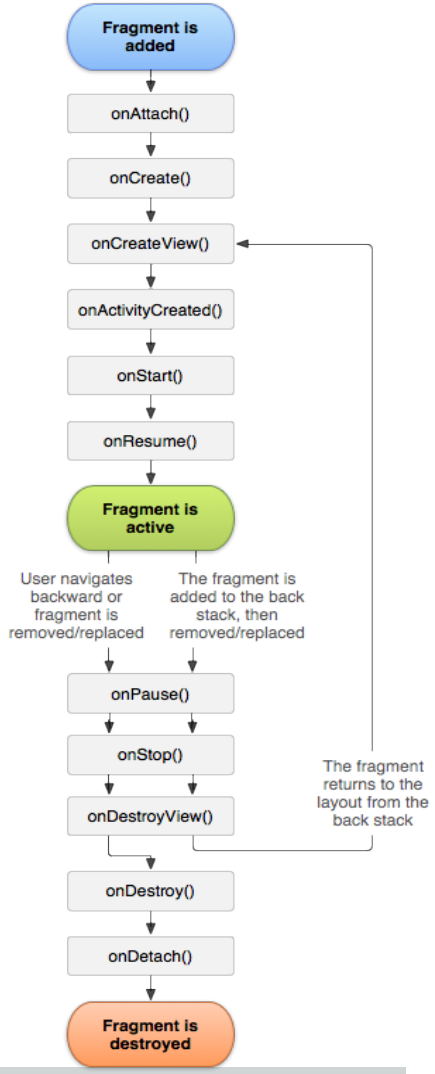
Visto che i fragment aggiungono un livello di indirezione per la gestione della UI è necessario un meccanismo di comunicazione tra Fragments e Activity che li gestisce.

6.5 Fragments - comunicazione

Soluzione: ogni fragment definisce un'interfaccia con specifiche callback e l'activity ospitante implementerà questa interfaccia. Tramite le callback che stiamo per vedere il fragment potrà usare questa interfaccia per comunicare con l'activity.

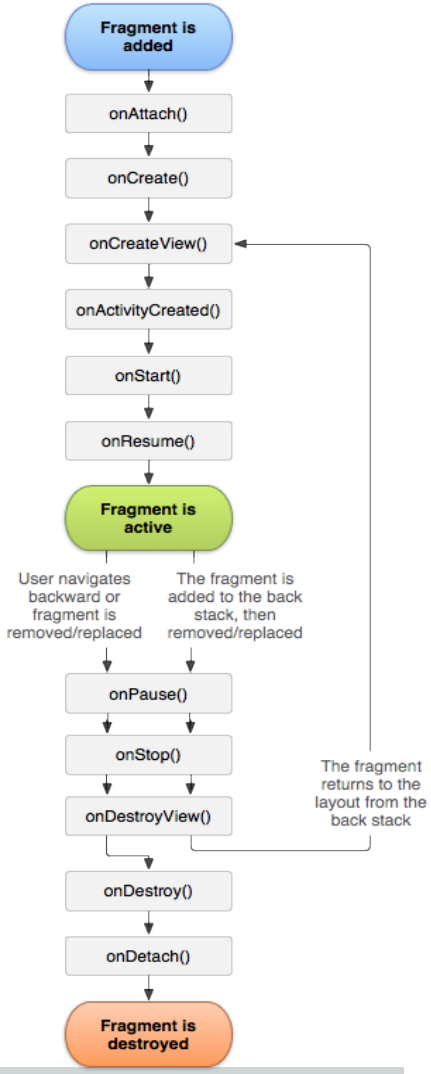
6.5 Fragments - ciclo di vita

Il ciclo di vita di un Fragment è fortemente correlato a quello dell'activity che lo contiene.



6.5 Fragments - ciclo di vita

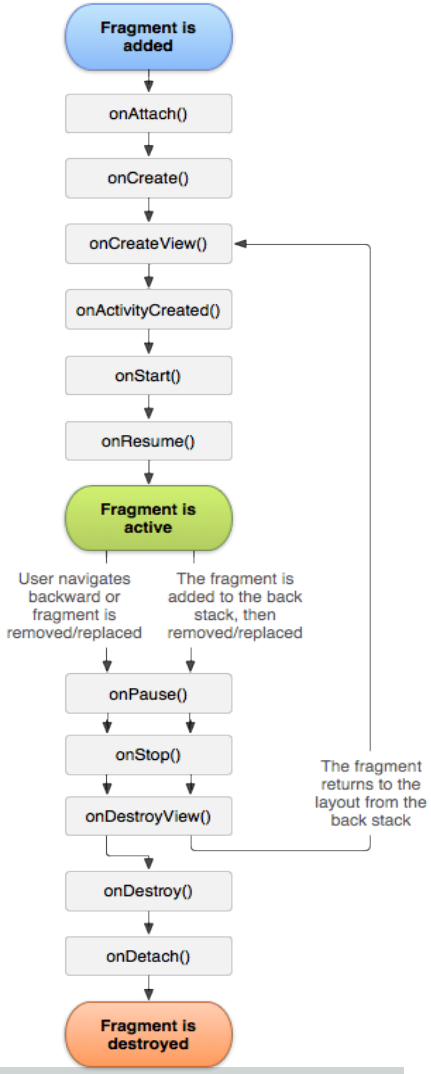
Spesso, aggiornare il codice per supportare i fragment può essere fatto semplicemente spostando il codice dalle callback nell'activity alle relative callback del Fragment.



6.5 Fragments - ciclo di vita

Principali callback.

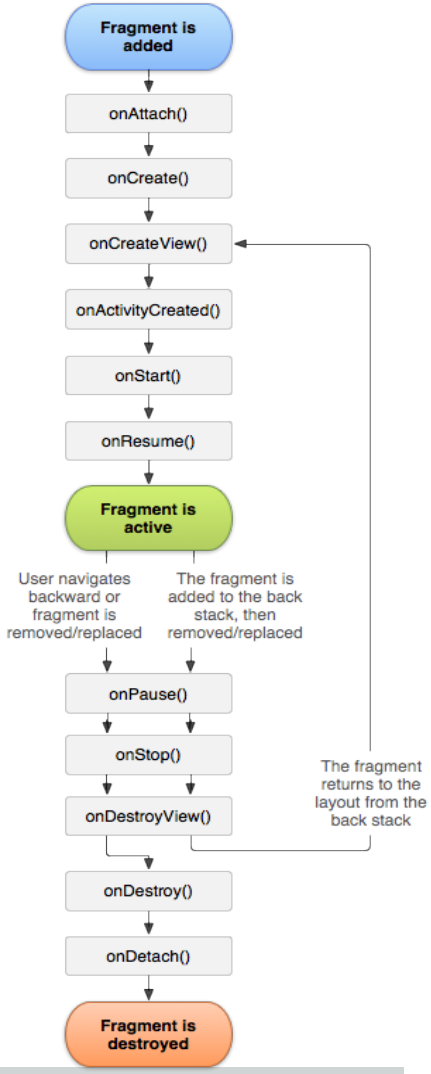
onCreate(): chiamata quando il fragment viene creato. Utilizzato per inizializzare oggetti che devono essere mantenuti anche nel caso in cui il fragment venga messo in pausa o stoppato.



6.5 Fragments - ciclo di vita

Principali callback.

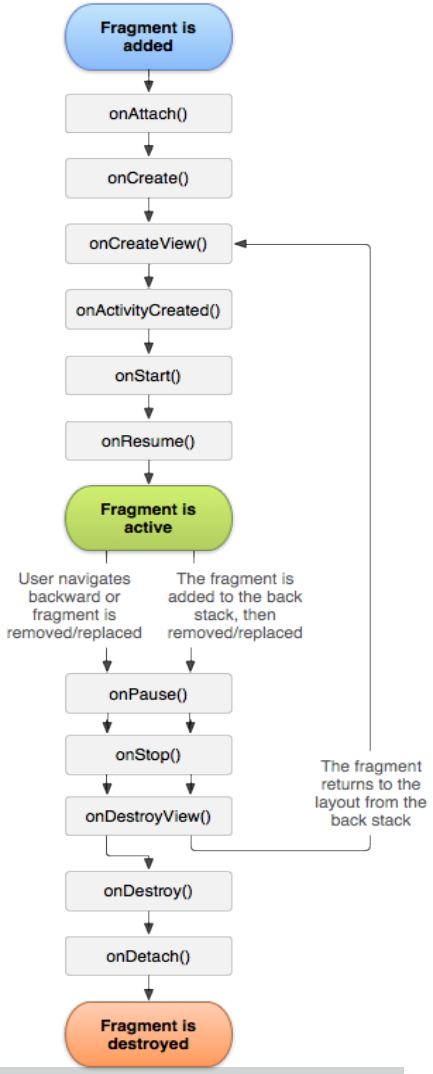
onCreateView(): chiamata quando il fragment deve creare l'interfaccia per la prima volta. Deve ritornare una *View* che sarà ciò che viene mostrato come elemento root in questo fragment (ritornare null se non vogliamo grafica).



6.5 Fragments - ciclo di vita

Principali callback.

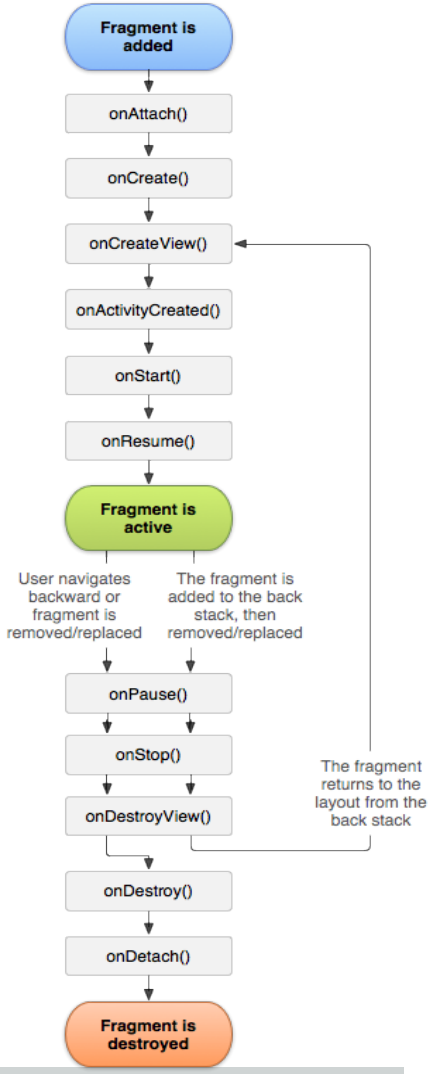
onPause(): chiamata quando l'utente abbandona il fragment: non necessariamente il fragment verrà distrutto.



6.5 Fragments - ciclo di vita

Principali callback.

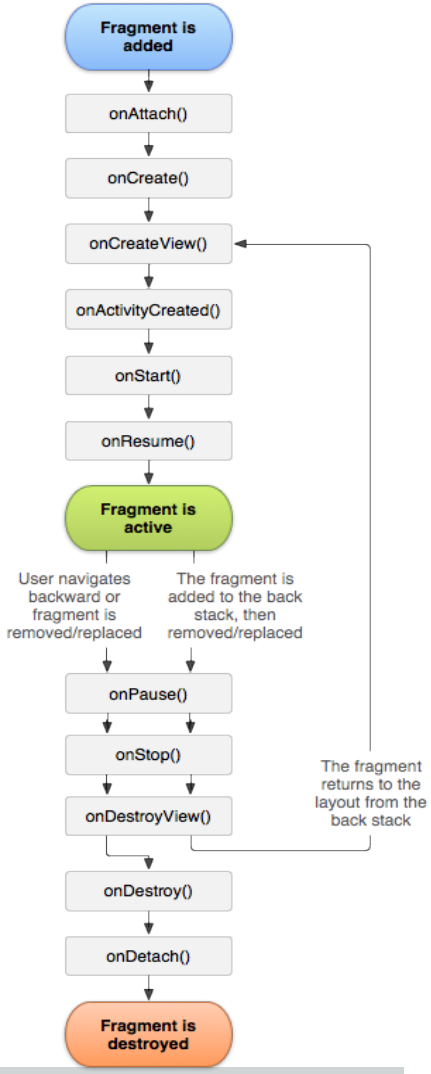
onAttach(): chiamata quando l'activity ospitante si aggancia al fragment. Tramite questa callback possiamo ottenere l'interfaccia per la comunicazione Fragment -> Activity.



6.5 Fragments - ciclo di vita

Principali callback.

onSaveInstanceState(): come nel caso delle activity questa callback deve essere utilizzata per immagazzinare lo stato attuale del fragment e poterlo ripristinare all'onCreate.



6.5 Fragments - creazione

Per impostare la UI del fragment fare l'override del metodo onCreateView:

```
...
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Creiamo la View a partire dal layout in XML.
        return inflater.inflate(R.layout.detail_fragment, container, false);
    }
...
```

6.5 Fragments - attach statico

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="it.neunet.sads.fragments.ListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="it.neunet.sads.fragments.DetailFragment"
            android:id="@+id/detail"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
```

Indica la classe da istanziare per gestire il layout e il comportamento.

6.5 Fragments - attach dinamico

L'attach dinamico consente di cambiare runtime i fragments nel layout -> **transaction**.

```
// Otteniamo il fragment manager per gestire il layout.
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
// Creiamo il nuovo fragment che vogliamo aggiungere al layout.
DetailFragment fragment = new DetailFragment();
// Lo aggiungiamo ed infine facciamo il commit della transazione.
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

6.5 Fragments - transazioni

L'attach dinamico porta come vantaggio l'utilizzo del TransactionManager che consente di "navigare" tra le modifiche.

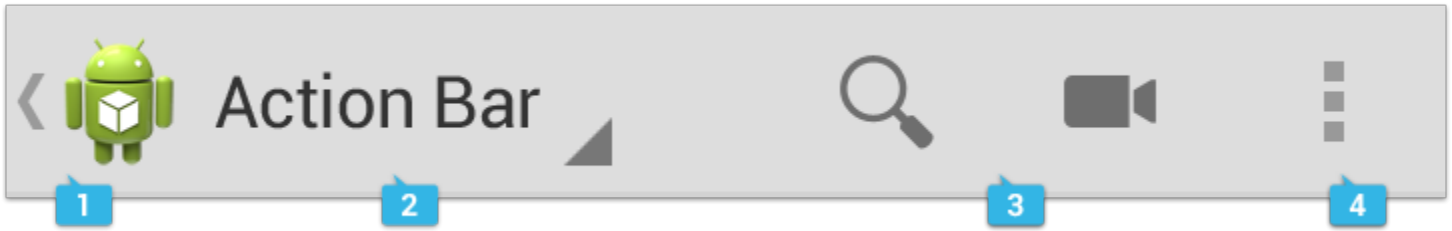
```
...
// Aggiungiamo il fragment alla transazione.
fragmentTransaction.add(R.id.fragment_container, fragment);
// Aggiungiamo la transazione al Back Stack.
fragmentTransaction.addToBackStack(null);
// Effettuiamo il commit: da questo momento se l'utente preme back torna alla config precedente.
fragmentTransaction.commit();
```

6.6

Action Bar e navigazione

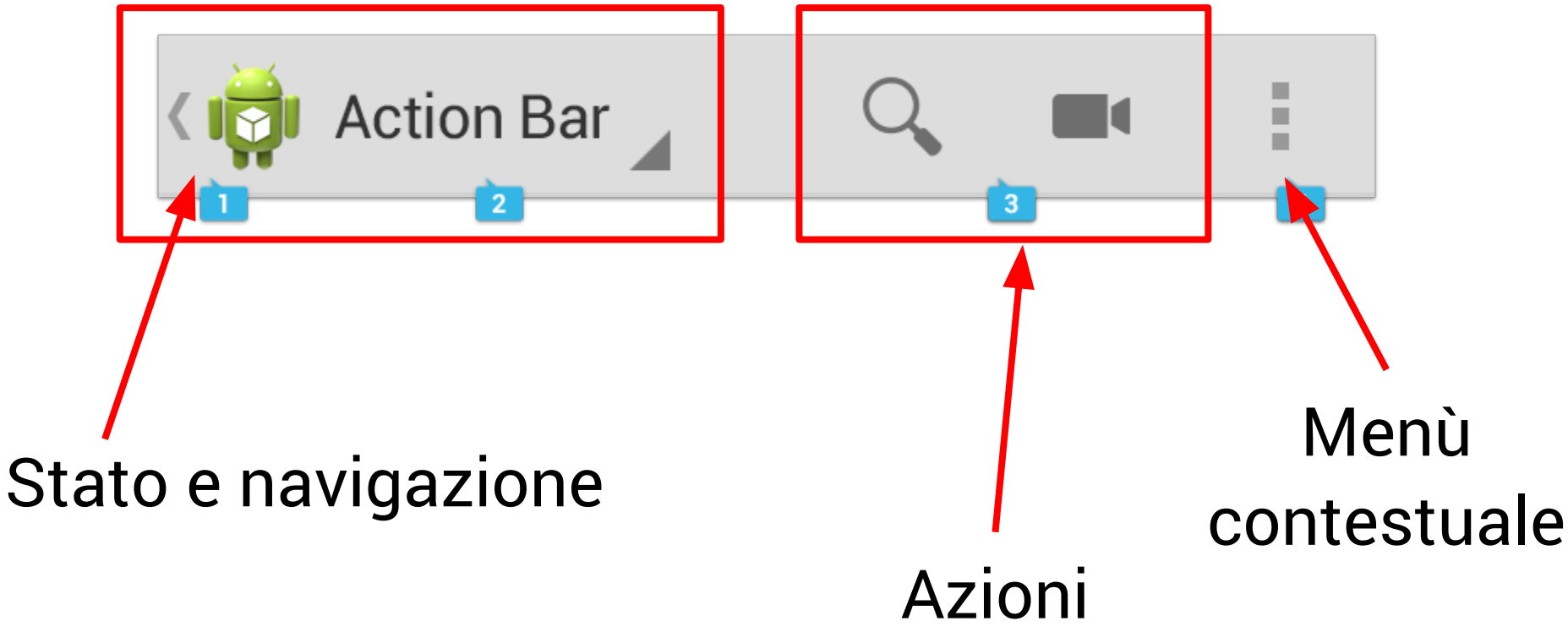
6.6 Action Bar

Novità di Android 3.0 *Honeycomb* (API lvl 11).



6.6 Action Bar

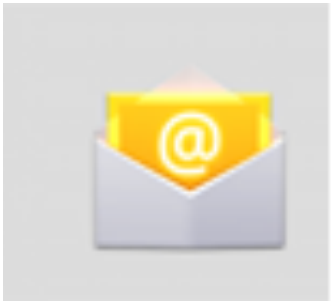
Novità di Android 3.0 *Honeycomb* (API lvl 11).



6.6 Action Bar per navigazione

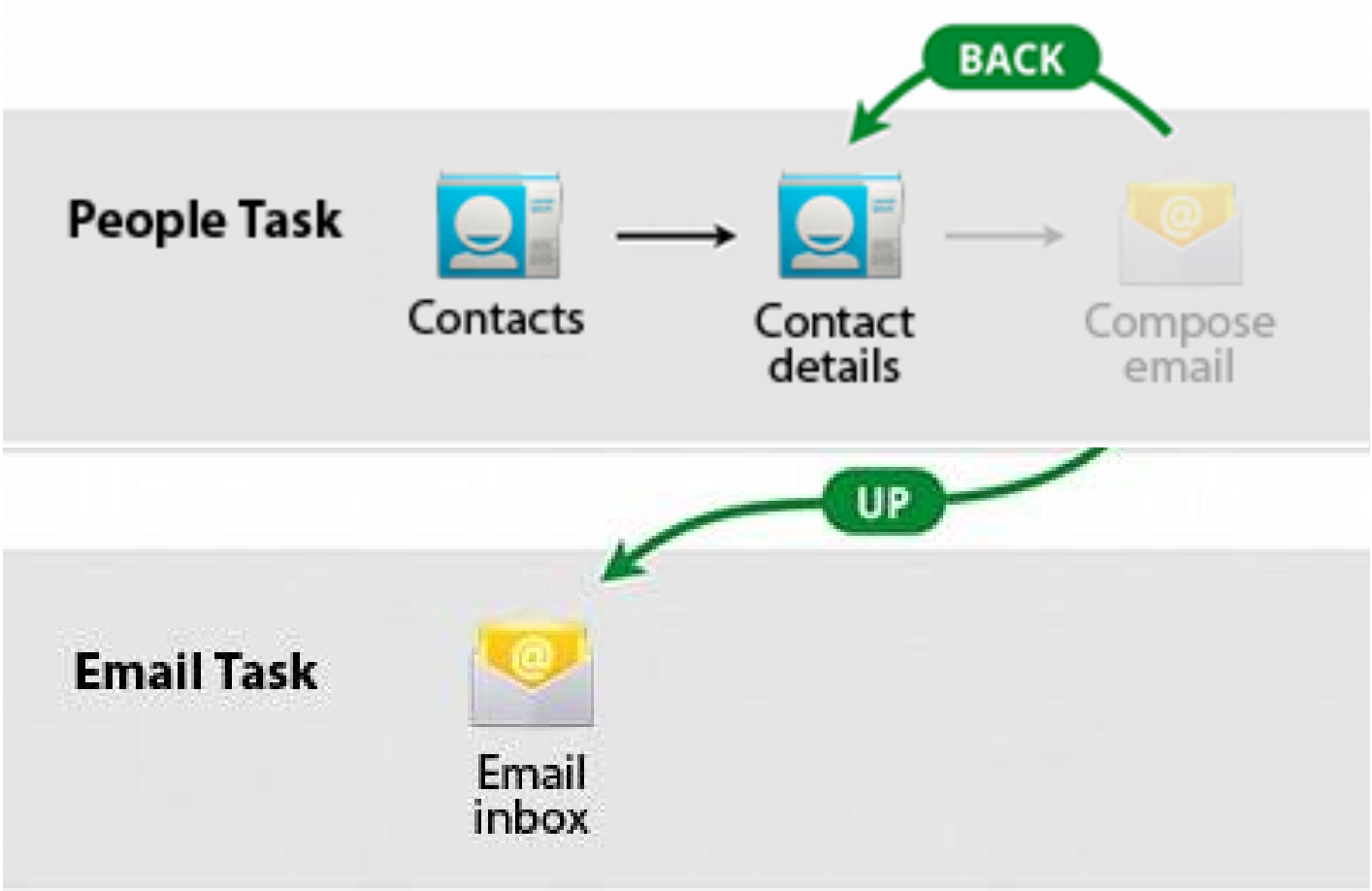
Nuovo modello di navigazione da *Honeycomb*:

Action Bar rappresenta graficamente anche lo stack di navigazione.



6.6 Nuovo modello di navigazione

6.6 Nuovo modello di navigazione



6.6 Task stack "sintetico"

Si utilizza un `TaskStackBuilder`.

(Nella support library.)

Genera uno stack "sintetico" su Android ≥ 3.0.

Non genera nessuno stack e funziona in maniera *legacy* su Android < 3.0.

6.6 Intent "up"

Per generare l'Intent di navigazione:

```
NavUtils.navigateUpFromSameTask()
```

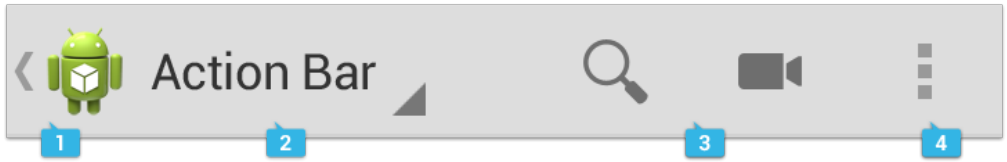
6.6 Pulsanti nella Action Bar

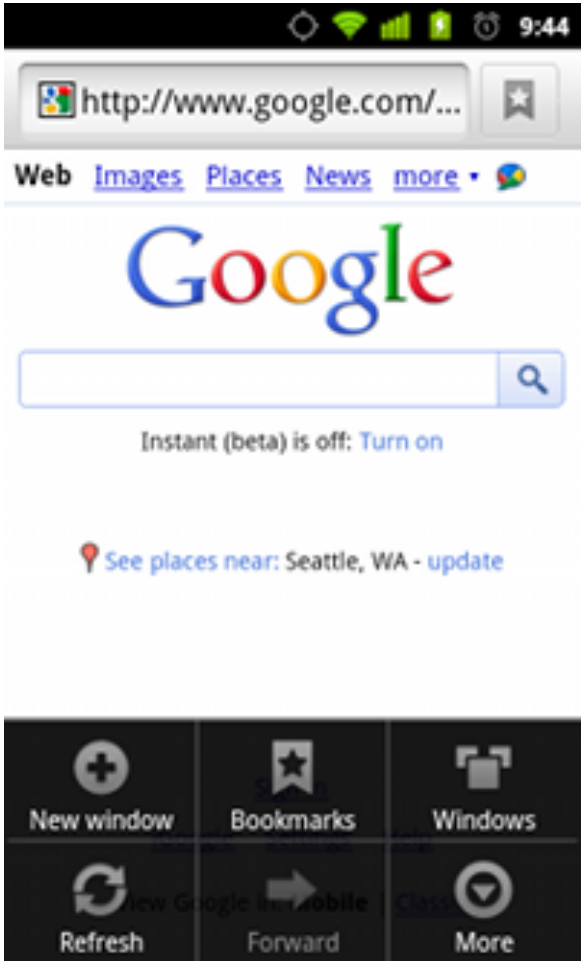
Conversione del "vecchio" menu contestuale di Android pre-*Honeycomb*.

Viene "gonfiato" come il layout principale:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater()
      .inflate(R.menu.main, menu);
   return true;
}
```

6.6 Retrocompatibilità dei menu





6.6 Definizione dei menu

Risorsa XML in cartella `/res/menu`:

```
<menu>
   <item
      android:id="@+id/action_ciao"
      android:showAsAction="always|withText"
      android:title="Ciao mondo!"
      android:titleCondensed="Ciao"
      android:icon="@drawable/ciao"
      />
```

6.6 Definizione dei menu

Risorsa XML in cartella `/res/menu`:

```
<menu>
    <item
        android:id="@+id/action_ciao"
        android:showAsAction="always|withText"
        android:title="Ciao mondo!"
        android:
        android:
        />
```

- always
- never
- ifRoom
- withText

6.6 Eventi dei menu

Metodo virtuale della classe `Activity`:

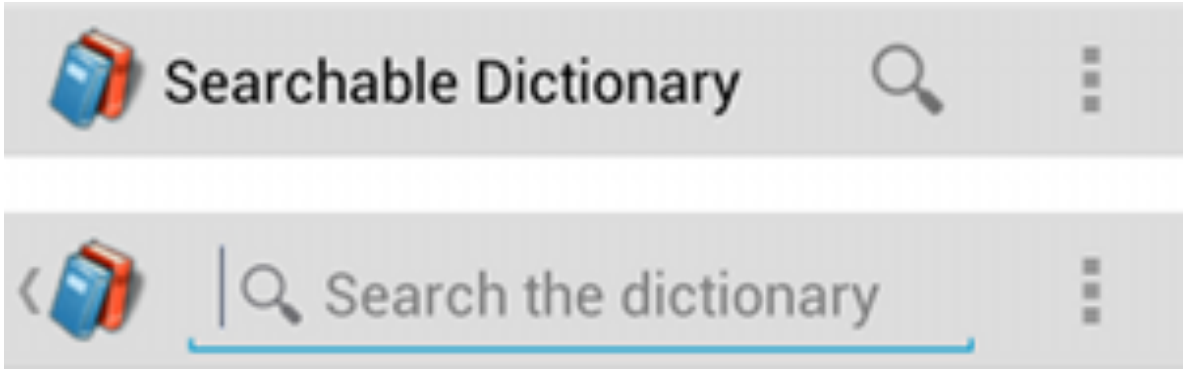
```
@Override
public boolean onOptionsItemSelected(MenuItem
item) {
   switch(item.getItemId()){
      case R.id.action_ciao:
         //...
         break;
```

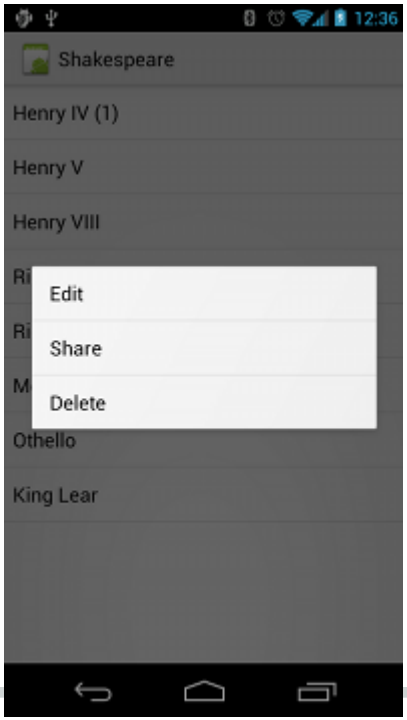
6.6 Demo

Action Bar, Notifiche e TaskStackBuilder.

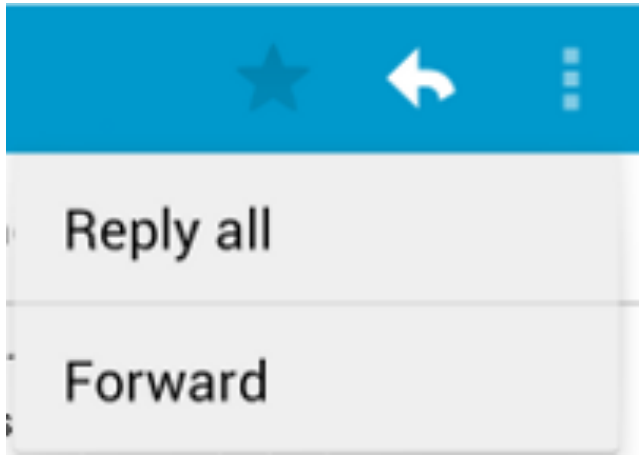
6.6 Altre cose dei menu...

actionViewClass



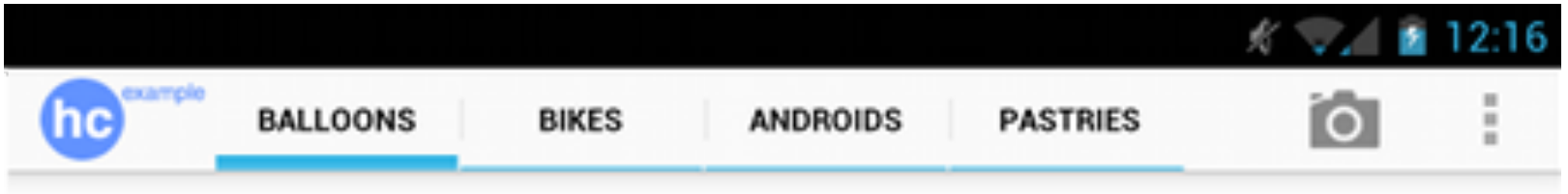


Context menu



Popup menu

6.6 Navigazione tab based

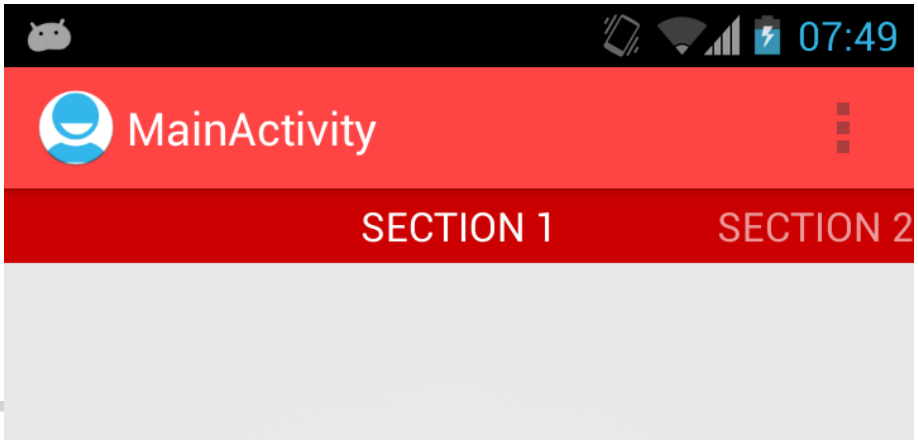


Tab nell'Action Bar: sfrutta Fragment nativi.

(API level 11.)

Soluzione compatibile:

- PagerTitleStrip
- PagerTabStrip



6.6 Demo

ViewPager, PagerAdapter e PagerTitleStrip.

6.7

Notifiche avanzate

6.7 Demo

Notifica "speciale".

6.7 Demo

Stili estesi delle notifiche:

`BigText`

`BigPicture`

`Inbox`

6.7 Demo

Notifiche di progresso.