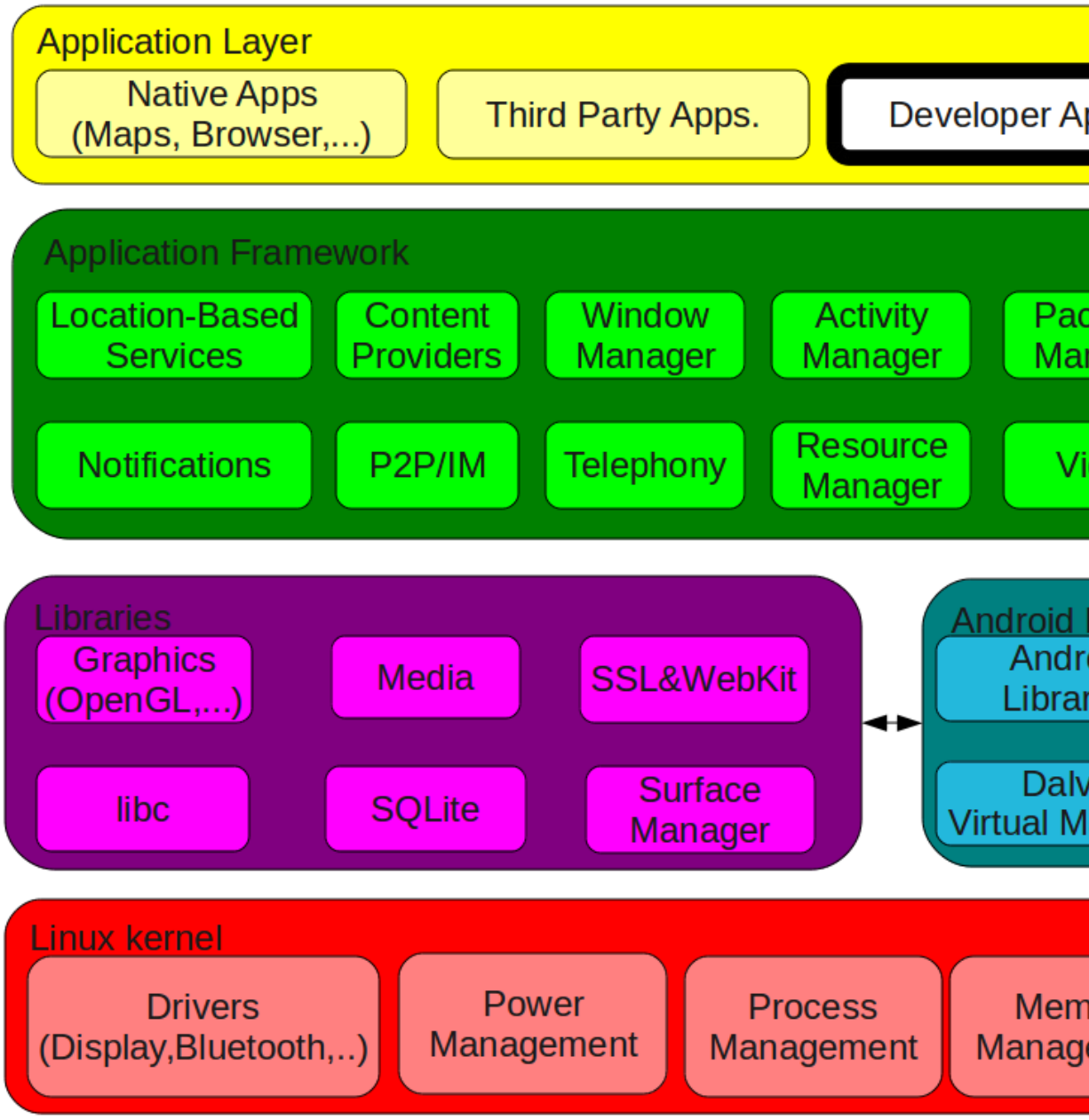
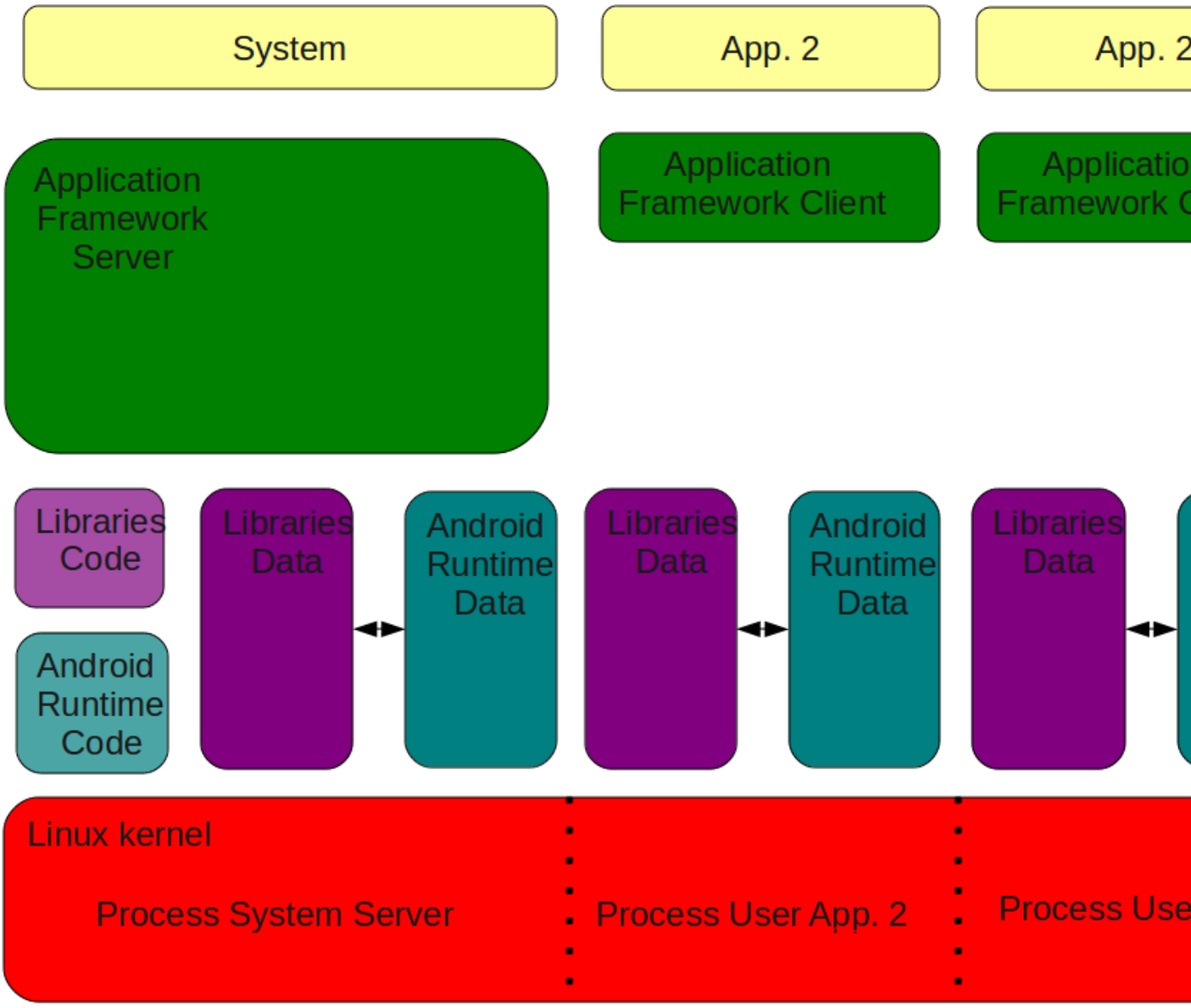
Lezione 19 – Cenni Sviluppo applicazioni su Android

Introduzione

Sebbene la programmazione sul sistema Android avvenga in Java, non viene utilizzata la J2ME per ragioni economiche e di performance. Al posto della V.M. per Java viene usata la Dalvik V.M.
Tale virtual machine ha le seguenti caratteristiche:
1) non accetta file .class ma un unico pacchetto .apk con un .dex che contiene tutte le classi. In questo modo di ottimizzano le dimensioni del file (no ripetizioni costanti, no ripetizione header etc.). Non è di contro possibile avere il caricamento dinamico delle classi, le classi caricabili sono solo quelle presenti nel .dex.
2) il bytecode è orientato ai registri (e non allo stack come la Java V.M.) per una maggiore velocità di esecuzione e compattezza del codice (a scapito di una maggiore difficoltà di compilazione e più basso riuso);
In questo modo si ottiene mediamente una riduzione del 50% della dimensione del bytecode prodotto e una diminuzione del 30% del numero di istruzioni da eseguire.
Inizialmente si pensava che bastasse implementare un gran numero di funzionalità in modo nativo e che potessero essere invocate staticamente per ottenere buone prestazioni. Per aumentare le performance dalla versione 2.2 di Android è previsto anche la compilazione Just In Time a granularità di Trace che ha queste caratteristiche rispetto a un JIT a granularità di metodo: 1) utilizzo minimo di memoria aggiuntiva 2) tempo veloce di avvio a scapito di una minore ottimizzazione (e possibilità di condivisione tra processi).
Per aumentare la velocità di startup delle DVM su cui girano i vari processi, il caricamento della DVM stessa deve essere molto veloce. Per questo si è usato una tecnica detta Zygote che permette la condivisione e il precaricamento di tutte le librerie core.



Per aumentare la sicurezza ogni applicazione gira su una propria istanza della VM che ha bisogno del proprio processo (analogamente alla VM standard di Sun). Per aumentare la sicurezza ogni processo diverso viene eseguito sotto le credenziali di un utente diverso e il kernel linux controlla i vincoli di accesso tra i vari processi.



In conclusione Android sfrutta il Kernel linux ma non è un sistema linux. Usa java come linguaggio ma le API che usa non sono quelle standard.

SDK

Si procede innanzitutto all'installazione della SDK:

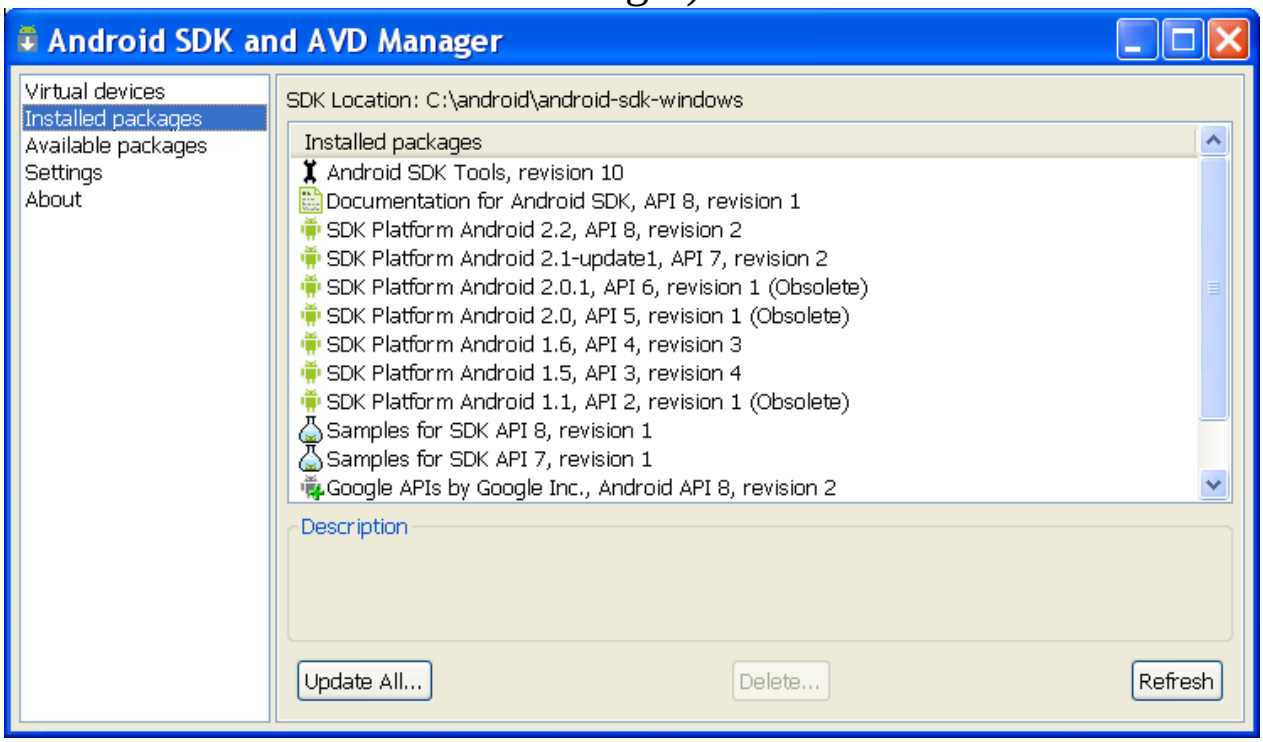
1. La SDK per lo sviluppo su Android richiede la preventiva installazione della SDK di Java.
2. Una volta installato Java, possiamo procedere all'installazione della SDK Started Package per Android
3. Opzionalmente ma vivamente consigliato e assunto nei seguito, si

può procedere all'installazione del plugin ADT per Eclipse

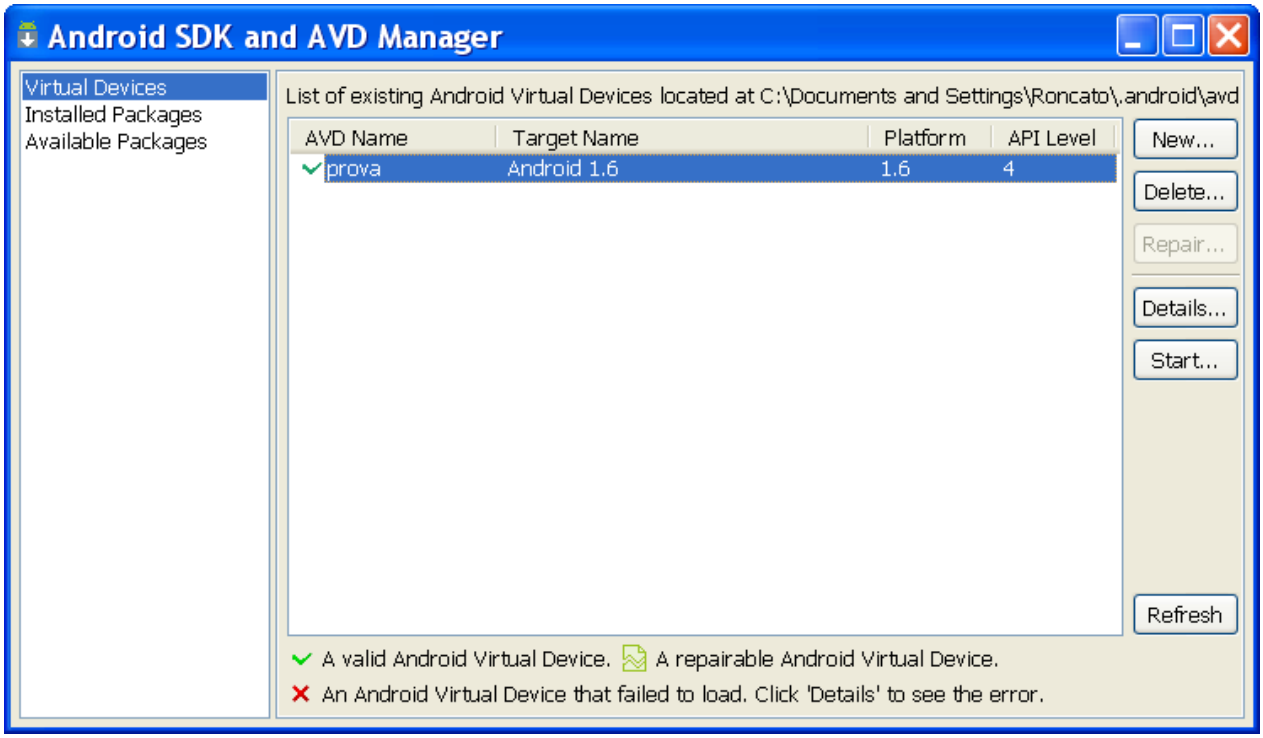
4. Successivamente si possono aggiungere altre piattaforme e componenti alla SDK di Android tramite l'Android SDK and AVD Manager. A esempio, possono essere installate varie versioni della SDK, pacchetti di documentazione o varie librerie. Consiglio di utilizzare le versioni più vecchie delle SDK in quanto più leggere.

Maggiori dettagli sono disponibili a link:
http://developer.android.com/sdk/installing.html

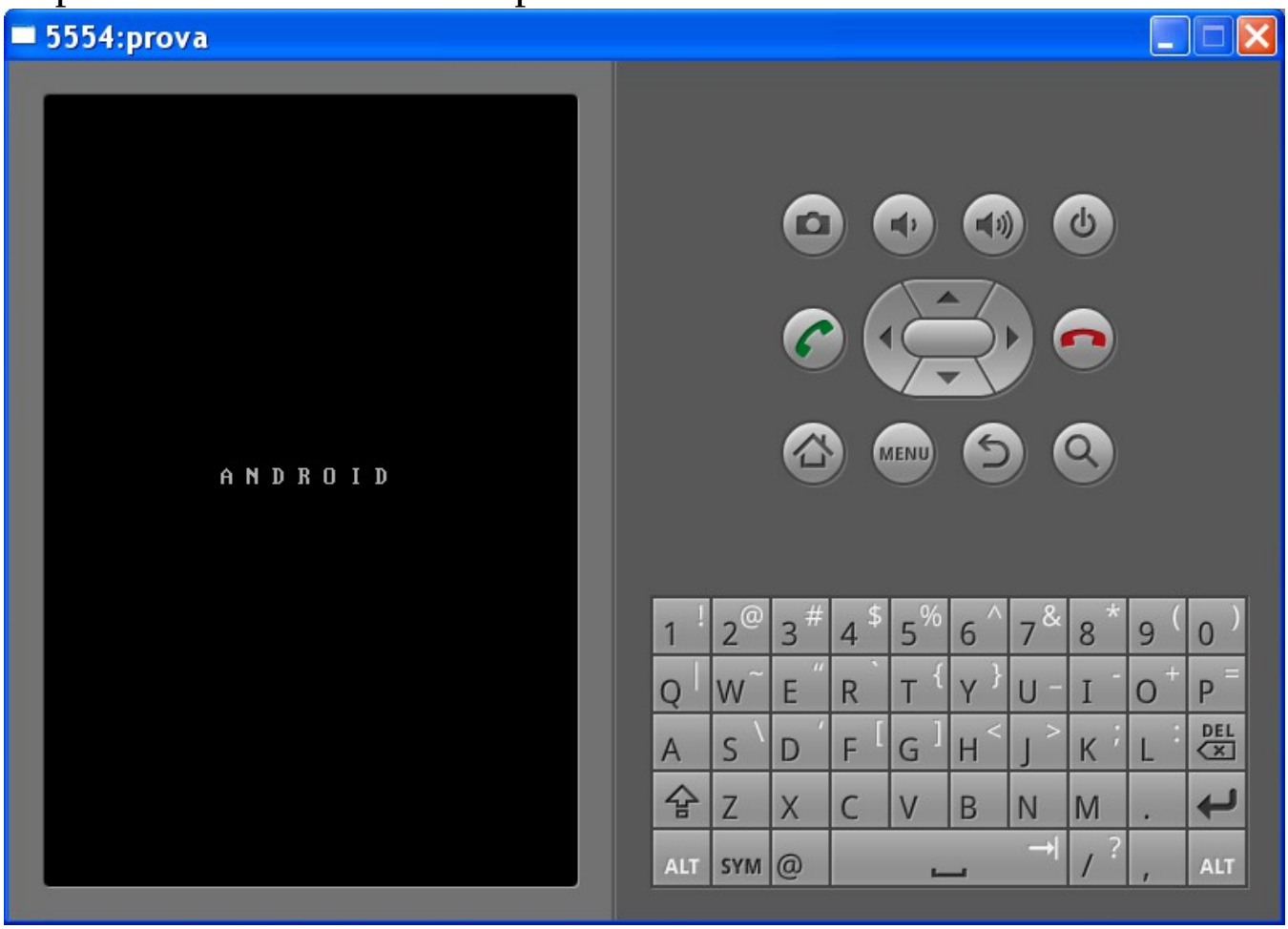
Aspetto SDK Manager dopo l'aggiunta delle piattaforme (Window ->Android SDK and AVD Manager):



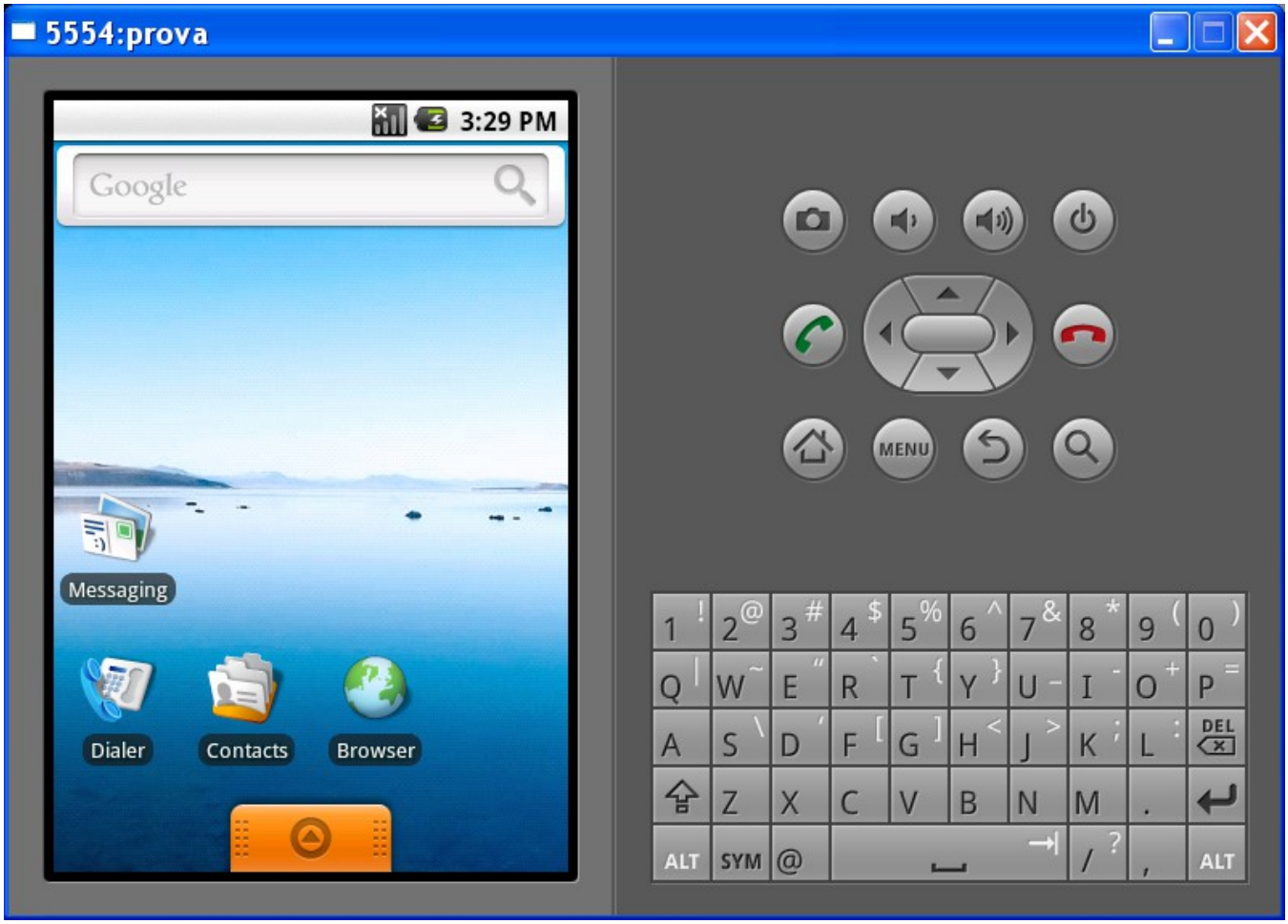
La SDK contiene anche un emulatore di terminare mobile con il quale è possibile selezionare il tipo di versione dell'API supportata dal terminale più la presenza o meno di vari dispositivi hardware (es GPS, accelerometri, quantità di memoria etc.)
Lista simulatori creati:



Aspetto del Simulatore alla partenza:



Aspetto simulatore in attesa esecuzione applicazione:



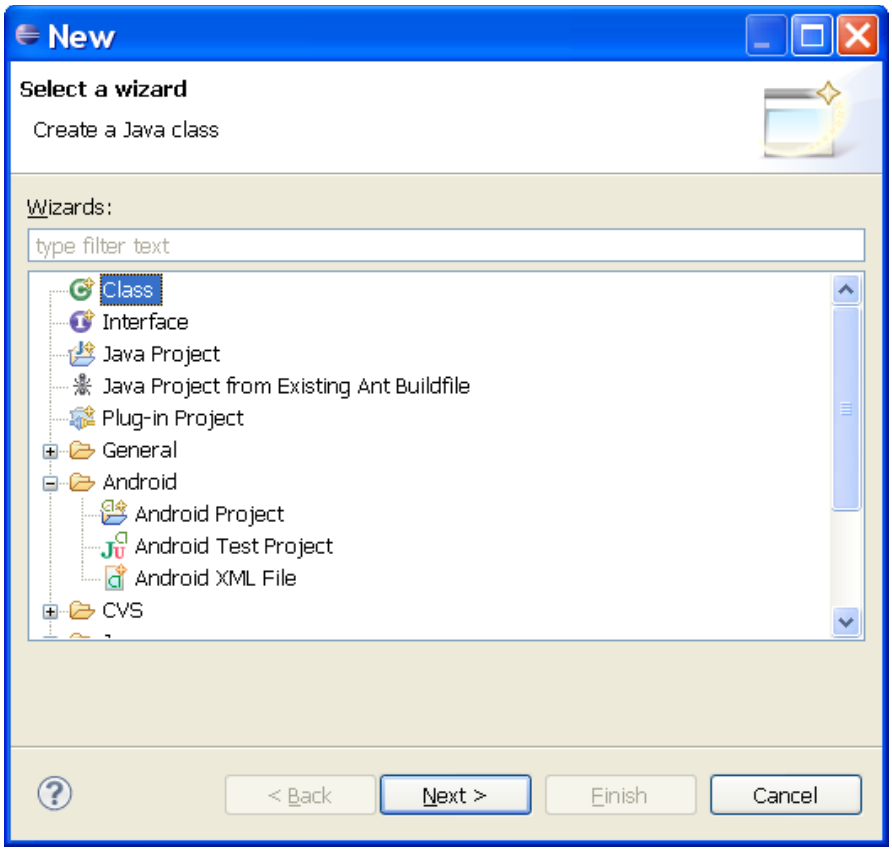
Architettura

Architettura orientata al riuso dei componenti ottenuto tramite file xml con la descrizione dell'aspetto dell'applicazione, file di risorse esterni al codice Java.
I componenti di un'applicazione possono essere:

- Activity: schermate visibili
- Services: background services
- Content provider: dati condivisi
- Broadcast receiver: ricevono e reagiscono a eventi in broadcast
- Intent: componenti che attivano altri componenti

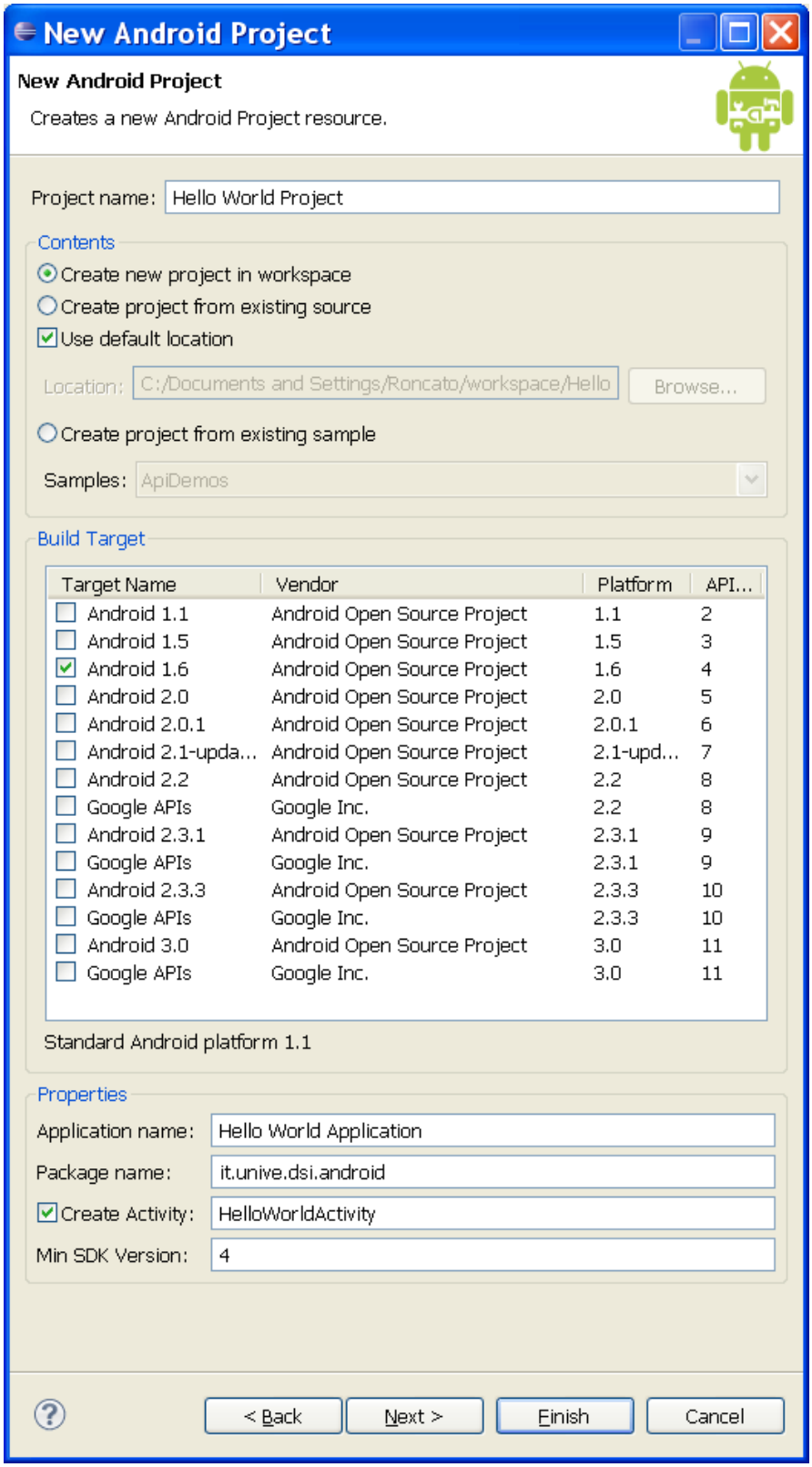
Esempio Applicazione

Da Eclipse selezionare New Other e quindi "Android Project".

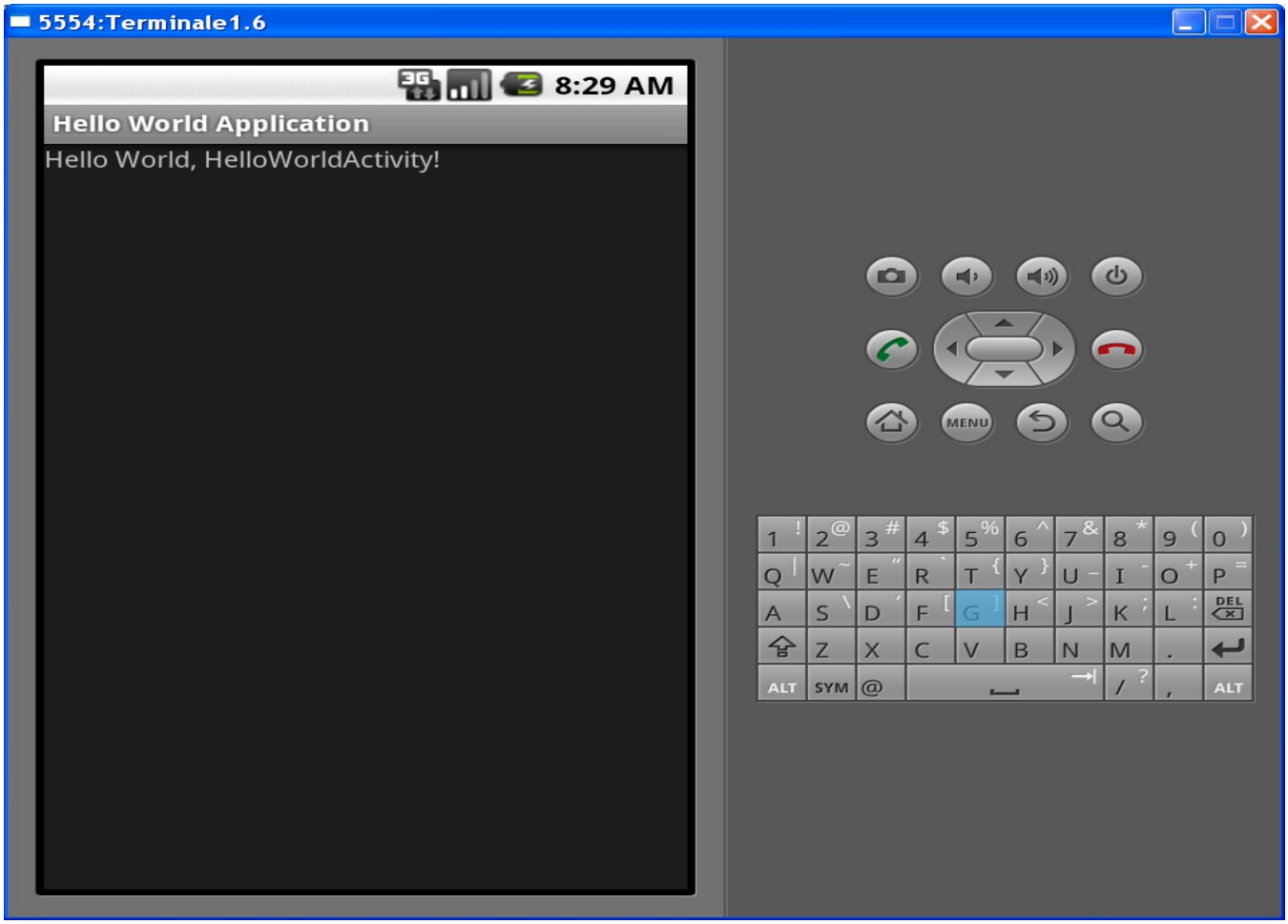


Successivamente compilare:

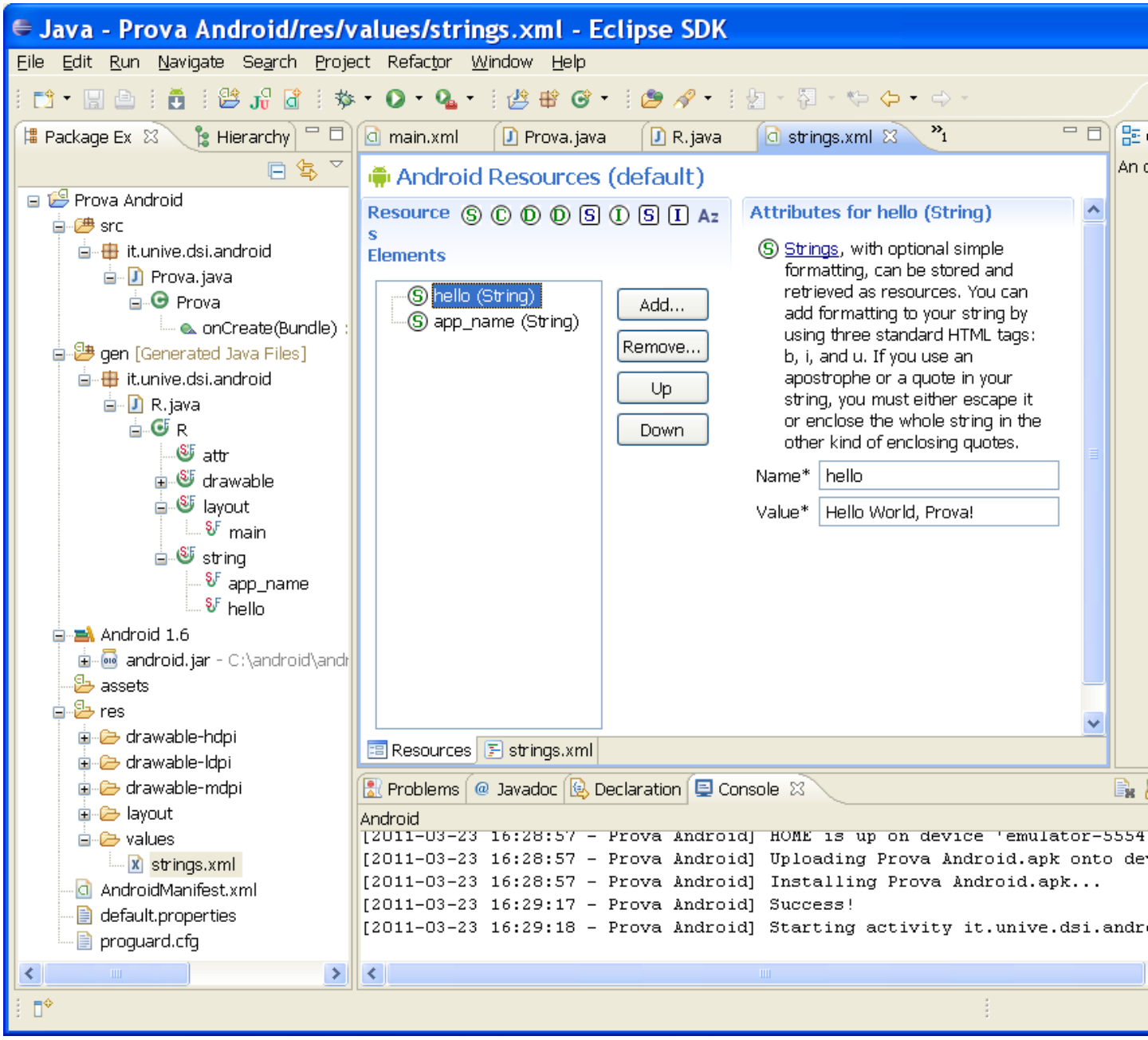
1) nome progetto
2) selezionare la/le piattaforma/e su cui far girare l'applicazione
3) il nome dell'applicazione
4) il package
5) il nome dell'attività
6) eventualmente la minima piattaforma su cui deve girare l'applicazione.



A questo punto viene creato in automatico tutto il necessario per la tipica applicazione Hello World. Se la facciamo girare nel simulatore fatto partire precedentemente otteniamo:



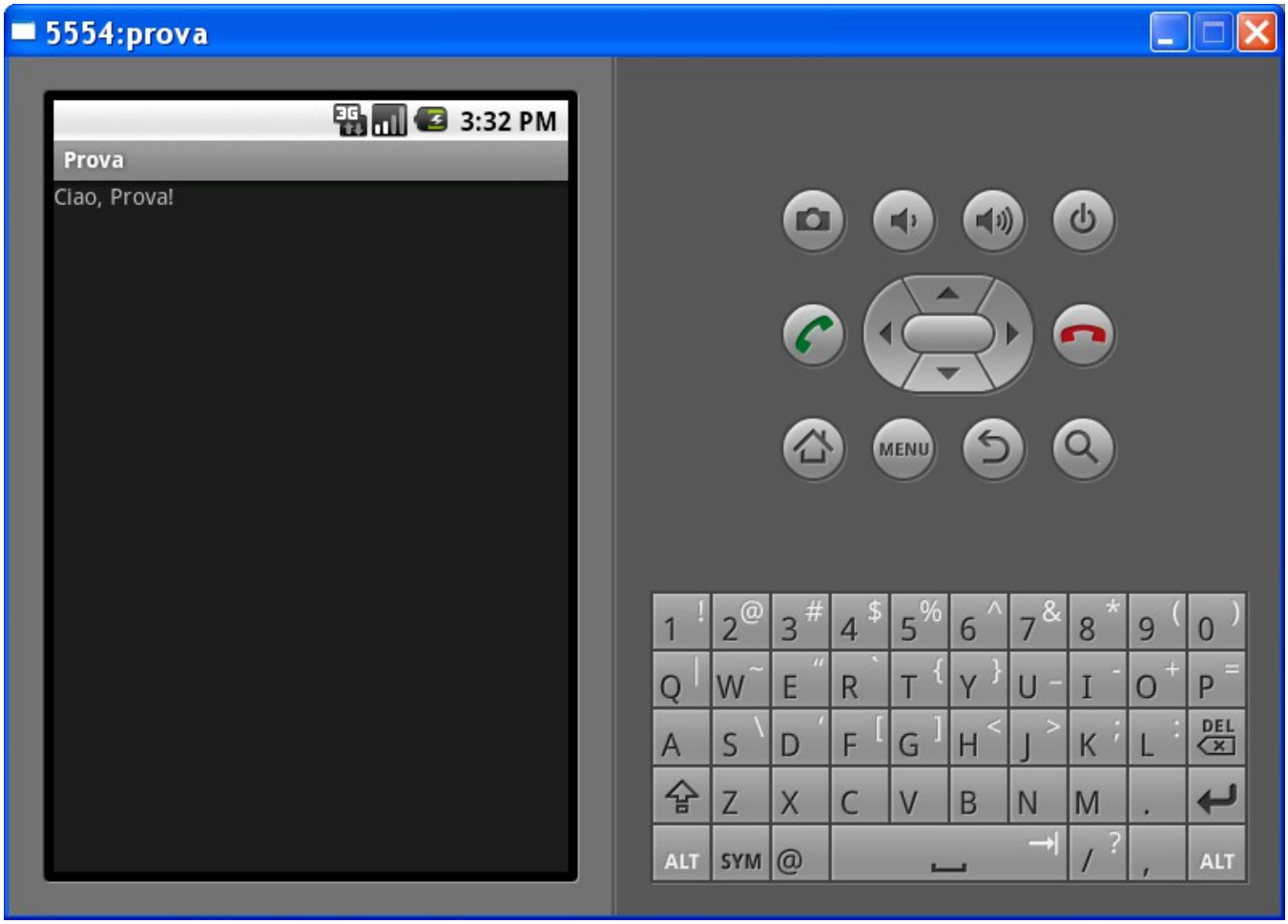
Senza bisogno di modificare il codice Java ma solo i file di risorse del progetto che contengono le stringhe possiamo modificare il progetto:



Che corrisponde al file xml strings.xml con il seguente contenuto:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="hello">Hello World,
HelloWorldActivity!</string>
   <string name="app_name">Hello World
Application</string>
</resources>
```

E cambiando la stringa in "Ciao, Prova!" otteniamo il seguente risultato:



Codice Java:

```
package it.unive.dsi.android;
import android.app.Activity;
import android.os.Bundle;
public class HelloWorldActivity extends
Activity {
  /** Called when the activity is first
created. */
  @Override
  public void onCreate(Bundle
savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```