ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea in Informatica

ORGANIZZAZIONE DELL'INTERFACCIA UTENTE SU ANDROID

Relatore:
Chiar.mo Prof.
VITTORIO GHINI

Presentata da:
VINCENZO COLUCCI

III Sessione di laurea
2010-2011

Indice generale

Ad Anna

Per avermi dato la forza di andare

avanti e di mantenere la promessa.

Glossario

Activity: una singola schermata che mostra un'interfaccia grafica. L'activity è una delle componenti basi delle applicazioni su Android. Un programma Android è di solito composto da più activity che sono avviabili anche singolarmente ma che una volta avviate fanno parte dello stesso processo.

App: è il termine con cui comunemente ci si riferisce alle applicazioni in contesto mobile.

Flower: struttura a fiore, da cui deriva il nome, che assolve alla funzione di avviatore di applicazioni.

Market: abbreviazione per *Android Market*, repository di software mantenuto da Google da cui è possibile scaricare apps gratuitamente o a pagamento.

Home: l'interfaccia principale del sistema operativo, richiamata tramite un apposito tasto.

Launcher: su Android le applicazioni pensate per avere il ruolo di Home vengono comunemente chiamate con questo nome.

Jailbreak: processo di *rooting* su iOS.

Rooting: la procedura di rooting consiste nell'ottenere tramite appositi hack i permessi di superutente sul dispositivo. Questa procedura di solito invalida la garanzia del produttore.

Sistema Operativo Desktop: si intendono i sistemi operativi orientati all'utilizzo su personal computer.

Introduzione

Figura 1

Figura 2

L’interfaccia è il mezzo con il quale l’uomo interagisce con un servizio, sia esso erogato da un oggetto, da una persona o da un software. Tuttavia ogni interfaccia ha la sua bontà e a seconda del servizio erogato questa può essere elementare (interruttore luce) o estremamente complicata (modulo di richiesta della borsa di studio).

Nel progettare un’interfaccia, gli obiettivi da tenere in considerazione sono tanti e questi determinano la bontà del risultato finale. Le figure 1 e 2 mostrano rispettivamente un esempio di cattiva e buona interfaccia. Qual è il problema del miscelatore in figura 1? Esso ha due valvole per la regolazione del flusso di acqua calda e fredda ma all’utente non interessa cambiare “il flusso di acqua calda”, un utente generalmente è interessato solo alla temperatura dell’acqua e al flusso d’acqua totale, parametri che sono invece direttamente gestiti dal miscelatore in figura 2. Quali sono i vantaggi che una buona interfaccia ci garantisce? Efficienza, intuitività, ergonomia, design. Nel caso di un azione banale come quella appena citata, questi vantaggi non sono poi cosi evidenti: usiamo un

rubinetto 5 o 6 volte al giorno e anche dovendo aprire due valvole al posto di una, il guadagno in termini di tempo è del tutto ininfluente ma quando un interfaccia viene usata per diverse ore al giorno, come quella di un programma di elaborazione testi o di un browser, la disposizione e la scelta degli elementi dell'interfaccia possono aumentare notevolmente la qualità d'uso del prodotto.

Android

Android è un sistema operativo opensource* acquisito nel 2005 da Google Inc., costituito da uno stack software su kernel Linux. La sua diffusione è cresciuta fino a portarlo a diventare il sistema operativo mobile più diffuso al mondo insieme ad iOS e si stima che attualmente ogni giorno siano attivati circa 850.000 dispositivi Android [1]. Il grande assortimento di software, la sua disponibilità gratuita e la possibilità di operare personalizzazioni sul sistema, hanno fatto sì che tutti i principali produttori di smartphone ad oggi abbiano in produzione qualche modello con il sistema operativo Google.

Perché ho scelto Android

Android è il sistema operativo di larga diffusione più libero attualmente sul mercato, sia in senso di disponibilità del codice sorgente, sia per quanto riguarda le possibilità di azione sul sistema e l'indipendenza dal produttore del OS. Il supporto ad una gran varietà di hardware e la possibilità di ridurre i costi di sviluppo sostenuti dai produttori, hanno reso possibile la sua diffusione su tutte le fasce di prezzo, rendendolo di fatto il sistema operativo mobile economicamente più accessibile.

* Android è rilasciato sotto licenza Apache 2.0, tuttavia alcune parti del sistema, come le applicazioni che interagiscono con i servizi Google, non sono rilasciate sotto licenza opensource.

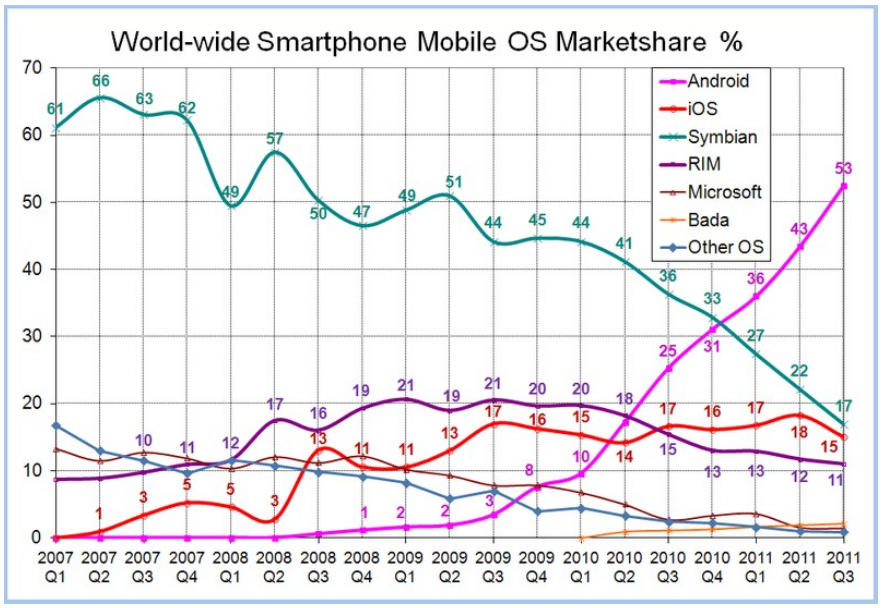

Figura 3 – Dati raccolti da Garner Inc [2]

Questi sono i tre punti su cui ho basato la mia scelta di utente e di sviluppatore: la libertà d'azione in quanto esigenza personale, l'apertura del codice come garanzia sul futuro di un sistema software e l'accessibilità economica poiché credo che ogni buona tecnologia debba essere accessibile a tutti.

La schermata Home

Ognuno dei principali sistemi operativi mobili ha un'interfaccia software principale chiamata generalmente *Home*. Questa schermata viene richiamata spesso durante l'utilizzo del dispositivo e ha il compito di permettere l'avvio di applicazioni e mostrare informazioni di riepilogo. L'importanza di questa interfaccia è tale che capita di sovente che molti utenti la confondano con il sistema operativo stesso. Una buona realizzazione è dunque fondamentale per il successo di tutto il sistema.

Il mio progetto si pone come obiettivo la creazione di un applicazione home per il sistema operativo Android costruita sui principi esposti precedentemente.

La tesi è suddivisa in 6 capitoli. Il primo presenta una breve panoramica sulle soluzioni adottate dai vari sistemi operativi mobile. Il secondo descrive gli obiettivi che mi sono posto e i principi che ho seguito nel progetto. Nel terzo capitolo sono descritti gli strumenti da me utilizzati durante lo sviluppo. Il quarto capitolo espone delle nozioni sull'architettura di Android utili per la comprensione dei capitoli successivi. Il quinto e il sesto capitolo infine mostrano le scelte progettuali e come queste sono state implementate.

1 Home nei principali OS mobili

1.1 iOS (iPhone, iPod, iPad)

Figura 1.1 - Home di iOS 4 su iPod Touch 3GS

L'interfaccia dei dispositivi Apple è caratterizzata dalla **fusione** dell'idea di Home con quella del menu principale. Tutte le icone sono distribuite sullo schermo e quando una schermata viene riempita ne viene dinamicamente aggiunta un'altra.

In fondo allo schermo un pannello, detto *dock*, permette di selezionare un numero limitato di app per l'avvio rapido.

Le restanti app non sono catalogate, ma a partire dalla versione 4 del sistema operativo, è stata aggiunta la possibilità di creare cartelle. Sta all'utente raggrupparle tramite un meccanismo comunque molto intuitivo.

Scorrendo lo schermo verso sinistra invece appare un campo di inserimento testo e una tastiera che permettono la ricerca rapida di contenuti all'interno del sistema.

La personalizzazione è limitata alla scelta dello sfondo e la versatilità è pressoché nulla (a meno di usare determinate app che richiedono l'utilizzo del Jailbreak).

Questa soluzione limita le funzionalità della Home al solo lancio di applicazioni ma ne rende l'utilizzo estremamente **semplice** e **intuitivo**.

1.2 Windows Phone 7 (Nokia Lumia)

Figura 1.2 - Demo web dell'interfaccia di WP7

La home di Windows Phone 7 mostra l'interfaccia più **originale** del panorama attuale ed è composta da due sole schermate. La principale, detta *start*, è dedicata alle operazioni più comuni ed ammette un numero illimitato di elementi. Quella laterale mostra una semplice lista dei programmi installati ordinata in ordine alfabetico, la catalogazione non è possibile se non per determinate voci come i giochi.

Gli elementi caratteristici dell'interfaccia di WP7 sono i **Live Tiles**, riquadri che oltre a fungere da avviatori, mostrano alcune informazioni utili come il numero di messaggi non letti o brevi messaggi testuali.

Lo stile grafico è semplice e minimale, predilige l'uso del nero come colore predominante; questa scelta oltre a creare un certo effetto stilistico, permette di risparmiare energia sui display OLED dove le zone scure non sono illuminate.

La personalizzazione si limita alla scelta dei colori degli elementi e alla selezione delle applicazioni da mostrare nella schermata start.

Questa soluzione è estremamente **efficiente** per quanto riguarda l'avvio delle applicazioni preferite e i live tiles contribuiscono a dare versatilità e possibilità di espansione seppur in maniera limitata.

1.3 Android

Figura 1.3 - Interfaccia di un tipica home per Android

Android è un sistema completamente **modulare** ed è l'unico a permettere la sostituzione dell'applicazione home. Ogni suo componente è sostituibile da un'applicazione che svolge la stessa funzione. Le applicazioni che fungono da home su Android sono chiamate *launcher*, termine che non è del tutto adatto poiché non si limitano solo al lancio di applicazioni. Tratterò per il momento del launcher di stock di Android.

Il concetto alla base del launcher di default di Android è quello direttamente mutuato dai personal computer di *desktop*. Le schermate non hanno nessuna funzione specifica ma possono ospitare quello che l'utente preferisce: icone, contatti della rubrica o widget.

Il menu delle applicazioni è una semplice griglia che mostra le applicazioni in ordine alfabetico. Non c'è alcuna forma di catalogazione e non è data la possibilità di creare cartelle nel menu. L'unico modo per raggruppare applicazioni è creare una cartella sul desktop e poi aggiungervi singolarmente una scorciatoia per ognuna di esse, un meccanismo così scomodo e inefficiente che nella pratica è raramente utilizzato.

L'approccio proposto da Android è senza dubbio il più **versatile** dal momento che permette all'utente di costruire il proprio ambiente di lavoro secondo le proprie necessità. Il rovescio della medaglia è che proprio come avviene nelle scrivanie reali, spesso l'utente non ha idea di come costruire questo spazio o non se ne vuole occupare e il desktop perde subito la sua funzione, diventando solo deposito di oggetti che restano inutilizzati.

Android non pone alcuna condizione vincolante affinché un'applicazione venga utilizzata come home. Ciò potrebbe far immaginare che Android pulluli di soluzioni alternative, in realtà la situazione è ben diversa e tutte

le principali proposte (ADW Launcher, Launcher Pro, Go Ex Launcher...) non fanno altro che aggiungere alla home originale qualche caratteristica come effetti di transizioni, supporto per gestures, possibilità di personalizzazione o miglioramenti in termini di performances.

In base a quanto appena detto, trovo opportuno sorvolare sulle differenze tra i vari launcher e continuare la trattazione riferendomi a questi ultimi come **modello home Android**.

1.4 Considerazioni

Come già detto, il **modello Android** sostanzialmente si basa sul concetto di desktop che si è rivelato vincente sui PC, ma perde gran parte del suo senso su piccoli dispositivi. Le app su dispositivi mobile non sono disposte in finestre, ma vanno ad occupare tutto lo schermo. È come avere fogli grandi quanto la scrivania, tutto quello che ci mettiamo sopra viene inesorabilmente coperto quando apriamo un app.

L'obiettivo del mio progetto è quello di creare un launcher originale, unendo i punti di forza di quanto visto finora: l'intuitività dell'interfaccia di iOS, la rapidità di quella di WP7 e la versatilità di Android. Il target è l'utente che vuole un sistema ben organizzato, subito pronto e facile da utilizzare.

2 Obiettivi e principi del progetto

La Home è il mezzo con cui uno smartphone mette a disposizione i propri servizi per l'utente.

Una buona interfaccia dovrebbe **non complicare** le funzioni semplici, come effettuare una chiamata o scattare una foto e dovrebbe guidare l'accesso a quelle più complesse. IOS e Android mettono a disposizione su un pannello in fondo allo schermo le azioni più comuni, WP7 fa di meglio e dedica un'intera schermata a questo scopo.

Nessuno dei 3 però fornisce una sistema nativo di **catalogazione** e di facile accesso alle applicazioni. L'accesso alle applicazioni dovrebbe essere fornito in base alla funzione e non all'ordine alfabetico; un utente potrebbe, infatti, dimenticare il nome di un programma o voler avviare un'attività senza avere in mente un'applicazione precisa. Un buon esempio di soluzione che risponde a questo problema è il menu del desktop environment Gnome (Figura 2.1).

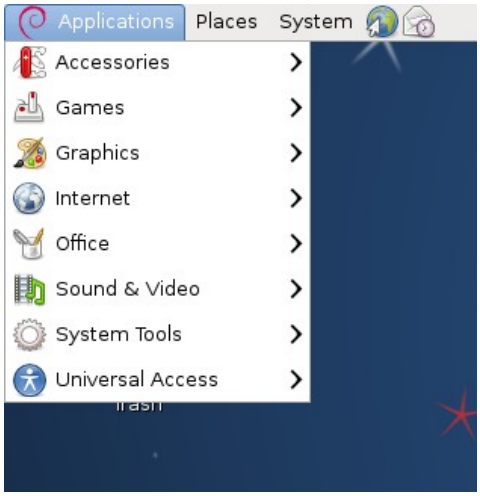

Figura 2.1 – Menu organizzato per categorie.

L'**eliminazione di ogni ridondanza** è un principio sempre valido nella progettazione di interfacce. Ogni funzione dovrebbe essere raggiungibile in un solo modo ben studiato, così da non generare confusione nell'utente e non aggiungere complessità al programma. Le icone delle funzioni principali dovrebbero essere abbastanza chiare da non richiedere ulteriori

descrizioni, questo oltre a semplificare il layout e permettere soluzioni stilistiche più pulite, permette di ridurre il testo da tradurre in diverse lingue e rendere **accessibile** il programma anche nelle lingue non supportate.

La realizzazione grafica ha un ruolo importante nella progettazione di un interfaccia. È importante creare un ambiente sobrio, che metta in risalto i controlli senza rinunciare ad un look accattivante.

Infine le possibilità di personalizzazione non sono un aspetto da sottovalutare. Poter scegliere uno sfondo, un set di icone o lo stile dei pulsanti contribuisce a far sentire un utente “a casa” ed evitano che presto ci si annoi del programma.

3 Ambiente di sviluppo

3.1 Sistema operativo

Il sistema operativo da me scelto per lo sviluppo del progetto è *Ubuntu 11.10* che utilizzo anche quotidianamente. Ci sono diversi motivi alla base di questa scelta.

Trovo irrinunciabile l'uso del linguaggio **bash** e la disponibilità di software da usare da linea di comando per creare in pochi minuti script che automatizzino operazioni altrimenti lunghe e ripetitive, le stesse che su Windows richiederebbero programmi specifici. La potenza del terminale inoltre, mi consente di interagire comodamente con il telefono tramite *adb* (software incluso nell'SDK android) per avviare una shell ed eseguire operazioni sul telefono o ottenere informazioni utili in fase di debug.

3.2 IDE

Gli ambienti di sviluppo più utilizzati per Android sono *Eclipse* e *Intellj IDEA*, entrambi multi-piattaforma e scritti in Java.

Eclipse è direttamente supportato da Google come IDE principale: ha strumenti potenti come l'editor grafico di layout ed è aggiornato direttamente tramite repository su Ubuntu. Il processo di installazione e configurazione è illustrato passo per passo sul sito ufficiale Android dedicato agli sviluppatori developer.android.com.

Intellj IDEA riesce a gestire meglio le risorse di sistema rispetto ad Eclipse e adotta alcune scelte di interfaccia che rendono più agile la gestione e la scrittura del codice.

Eclipse è sicuramente la scelta consigliata per sviluppatori android ma Intellj IDEA è un scelta valida per chi vuole un ambiente agile e non ha bisogno di funzionalità di alto livello come l'editor grafico.

La mia scelta è ricaduta su Eclipse, con il quale ho più familiarità, dal momento che reputo i vantaggi offerti da Intellj IDEA non abbastanza importanti da giustificare la migrazione ad un sistema che mi è in gran parte sconosciuto. Inoltre un fattore decisivo nella mia scelta è stato il più ampio bacino di utenti di Eclipse, che generalmente significa ampio supporto e maggiori probabilità di trovare soluzioni a problemi che altri sviluppatori hanno incontrato prima di noi.

3.2 Hosting e Software di Controllo Versione

Durante il progetto mi sono servito del hosting gratuito fornito da Bitbucket *https://bitbucket.org*. Molti siti offrono free hosting per progetti opensource ma, non essendo sicuro della licenza con cui rilasciare il codice, ho preferito riservarmi la libertà di decidere in un secondo momento.

Il software di controllo versione usato da BitBucket è **Mercurial,** che tramite la possibilità di gestire il *versioning* in locale, permette maggior velocità e l'uso anche senza connessione internet.

4 Architettura Android

4.1 La piattaforma Android

La piattaforma è costituita da 3 strati: il kernel Linux, il middleware che fornisce servizi di comunicazione tra i processi e lo strato applicativo.

Le applicazioni, che vengono comunemente definite “apps”, sono distribuite tramite archivi con estensione APK e sono scritte in **Java non standard** (tramite l’NDK è possibile includervi parti compilate in codice macchina). Una volta installata, ad ogni applicazione viene assegnato un nuovo *userid* e un **insieme limitato di permessi**; all’esecuzione ogni applicazione viene avviata da una macchina virtuale differente. In questo modo ogni applicazione viene eseguita in una *sandbox* e viene applicato il principio dei *privilegi minimi*. [3]

Come già detto, le applicazioni sono scritte in Java ma la VM utilizzata da Android non è la Java VM. Prende il nome di **Dalvik Machine**, lavora con un bytecode diverso ed è ottimizzata per l'utilizzo su dispositivi mobili.

Il costo maggiore in termini di efficienza resta ed è rilevante, in un contesto con risorse limitate in quanto a potenza di calcolo, memoria e autonomia. D'altra parte con la scelta di Java, Google ha garantito agli sviluppatori un'ampia disponibilità di librerie e l'utilizzo di un linguaggio di programmazione già conosciuto.

4.2 Compatibilità tra dispositivi diversi

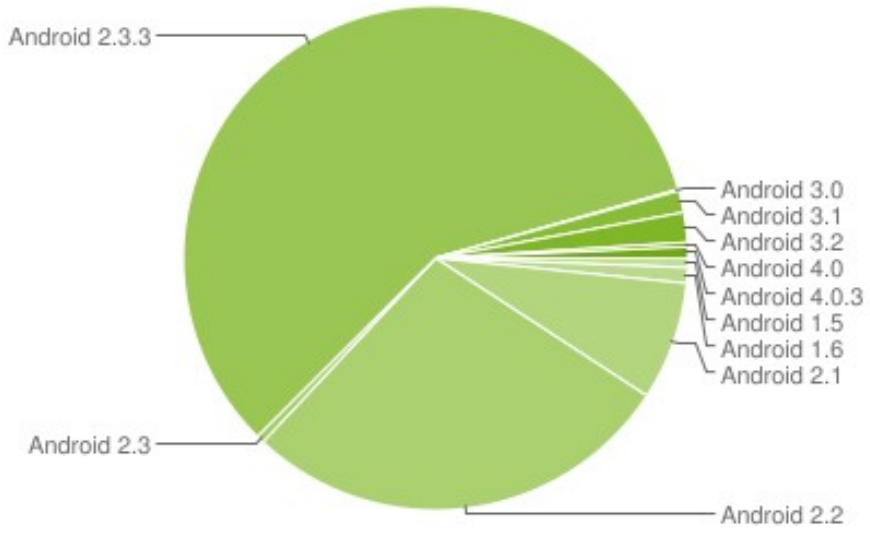

Figura 4.1 - Distribuzione delle diverse versioni di Android al 25 febbraio 2012 [4]

La compatibilità di un applicazione su più dispositivi Android incontra due problemi:

Il primo è di **natura software** e consiste nella *frammentazione*, cioè la distribuzione delle varie versioni del sistema operativo sui dispositivi attualmente attivi.

Il secondo è di **natura hardware**. A differenza di iOS, Android è installato su dispositivi anche con caratteristiche molto diverse tra loro. Le dimensioni dello schermo variano dai 3 pollici dei piccoli lettori multimediali ai 10 pollici dei tablet. Uno schermo AMOLED ha caratteristiche molto diverse da un LCD e un touchscreen resistivo ha più limitazioni di uno capacitivo. La tastiera non è presente su tutti i terminali, alcuni terminali non hanno nemmeno i tasti fisici.

Tutte queste differenze causano la difficoltà per gli sviluppatori di rilasciare applicazioni compatibili con i vari dispositivi.

4.3 Le activity e le View

Un'activity in Android è una delle componenti principali di un programma, la si può pensare come una singola schermata dotata di un'interfaccia grafica. Un programma Android è di solito composto da più activity che sono avviabili anche singolarmente, ma che una volta avviate fanno parte dello stesso processo.

La piattaforma di Android mantiene in primo piano (*foreground*) un solo screen alla volta, portando in secondo piano (*background*) gli altri.

Le varie activities sono "impilate" come un mazzo di carte: il tasto **back** permette di passare da uno screen all'altro, portando in background lo screen attuale e assegnando il foreground al precedente.

Un'activity può disegnare un'interfaccia usando un file XML che ne descrive il layout. La struttura di un layout è composta da View e ViewGroup, le prime descrivono elementi grafici come pulsanti o campi di testo, i secondi hanno il ruolo di contenitori.

Per ogni View devono essere necessariamente specificate altezza e larghezza. Le unità di misura disponibil sono i pixel e i *dip, density-indipendent-pixels,* che come il nome suggerisce, permettono di indicare una dimensione indipendentemente dalla risoluzione dello schermo.

5 Progettazione

Figura 5.1 – La struttura ideata per il programma.

Il primo passo è stato dividere il programma in due parti: la home (visibile nelle schermate 1, 2 e 3 della figura 5.1) e il menu delle applicazioni (schermata 4 in figura 5.1).

5.1 La Home

La home nella mia visione ha 3 funzioni principali: mostrare informazioni, permettere l'avvio rapido di applicazioni, mettere a disposizione una gestione delle attività. Queste tre funzioni sono svolte

dai tre pannelli di cui si compone: Widget Panel, Standby Panel, TaskManager Panel.

5.1.1 Standby Panel

Lo StandbyPanel, indicato dal numero 2 nello schema in fig. 5, è la vera schermata principale del sistema, mostra un orologio, i pulsanti per l'apertura del menu applicazioni e del menu contatti e una struttura che ho chiamato Flower.

L'orologio non è parte del programma (che si limita a consigliare una scelta di default) ma è liberamente selezionabile tra quelli disponibili come widget per Android, in modo da consentire un'ampia scelta all'utente.

Il **Flower** è una struttura che mostra un numero variabile di elementi disposti ad anello, che permettono l'avvio rapido di determinate *azioni*.

Figura 5.2 – Il Flower

Un'azione è un'astrazione sulla reale applicazione da avviare. Aggiungendo all'anello ad esempio un'azione di tipo *browser*, il programma mostra una lista dei browser web installati e permette di impostare una scelta predefinita. In questo modo l'utente non deve preoccuparsi dell'applicazione da avviare, ma sceglie semplicemente cosa fare. L'utente può aggiungere al Flower anche applicazioni non associate ad una specifica azione, ma il numero massimo di applicazioni del Flower è

limitato a 9 in modo da evitare che l'utente cada nell'abuso che vanificherebbe l'idea per cui è stato pensato.

I due pulsanti sul fondo della schermata, invece, rappresentano i due menu principali, quello delle applicazioni e quello dei contatti. L'idea è che un'attività può partire dalla volontà di interagire con un applicazione installata sul dispositivo o da quella di comunicare con un contatto.

5.1.2 TaskManager Panel

Questa è senza dubbio la parte più controversa del progetto. Il pannello TaskManager, indicato dal numero 3 nella figura 5.1, fornisce uno strumento di gestione dei processi in esecuzione, permettendo di passare da un processo all'altro o di forzare la chiusura di un'applicazione.

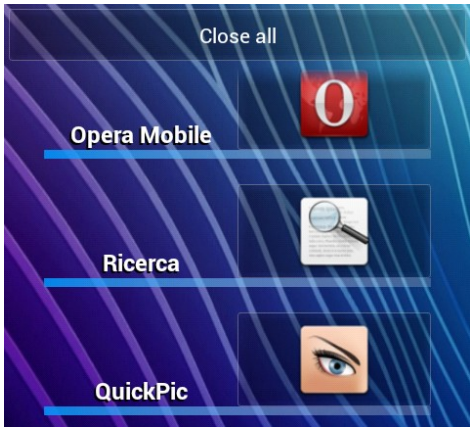

Figura 5.3 – TaskManager

Questa funzione è pensata per sopperire ad una mancanza del sistema operativo. Infatti Android adotta una gestione dei processi "innovativa" che è completamente lasciata al sistema. L'utente non ha meccanismi per chiudere le applicazioni in esecuzione né tanto meno per visualizzarle.

L'introduzione di un taskmanager andrebbe contro le intenzioni dei progettisti Android, inoltre l'unico modo di chiudere un programma è quello di inviargli un segnale di terminazione. Questo tipo di chiusura forzata non si preoccupa dello stato del programma e in alcuni casi, potrebbe interrompere operazioni importanti come la scrittura di dati sul

disco che potrebbe lasciare il sistema in uno stato di incoerenza, con conseguenti perdite di dati o insorgere di malfunzionamenti.

Ho deciso di insistere sull'introduzione di questa feature, dal momento che reputo una grave mancanza l'assenza di un sistema di gestione dei processi in esecuzione, con la riserva di decidere sul suo sviluppo in base ai feedback ricevuti dagli utenti.

5.1.3 Widget Panel

I widget sono al momento una caratteristica esclusiva di Android, anche se sono disponibili già da anni sui sistemi operativi desktop. La loro funzione è quella di espandere le funzionalità del desktop presentando semplici informazioni o fornendo scorciatoie per determinate features.

Spesso, dopo un iniziale utilizzo diffuso di tali programmi, si tende a limitarne l'uso. I widget sono costantemente attivi e sfruttano, seppur in maniera ridotta, risorse di sistema; inoltre sono difficilmente integrabili tra loro e con il resto del sistema. La grande varietà di widget e le numerose funzioni a cui assolvono rendono però impensabile la loro esclusione, in quanto ridurrebbe drasticamente le possibilità di espansione e quindi la versatilità del sistema. La mia scelta è stata dunque quella di limitare l'utilizzo dei widget ad una sola schermata, indicata dal numero 1 nella figura 5.1, in modo da indirizzare il sistema verso un ordine. Avendo uno spazio definito perde senso l'idea della disposizione spaziale

dei widget che quindi una volta scelti, vengono semplicemente aggiunti in fondo alla lista.

5.2 Il menu delle applicazioni

Il menu delle applicazioni, in Android comunemente chiamato *drawer*, ha la caratteristica di essere organizzato per categorie. È composto da una lista delle categorie e dalla griglia delle icone dei programmi.

Cliccando sull'icona di una categoria vengono visualizzati i programmi appartenenti ad essa, mentre premendola per qualche secondo appare un menu che da la possibilità di aggiungere o rimuovere determinate categorie.

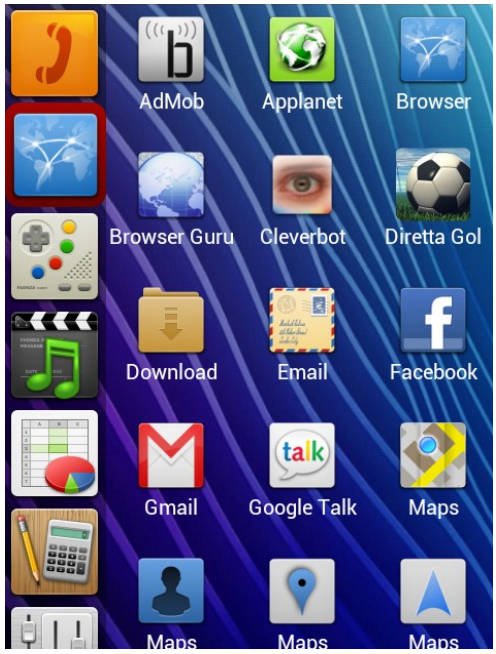

Figura 5.4 – Il menu principale

Le icone dei programmi su singolo tocco lanciano l'applicazione associata e su pressione prolungata permettono di entrare in modalità editing, dove è possibile cambiarne la categoria o sceglierne la disinstallazione (in modo simile a come succede su iOS).

5.3 Realizzazione Grafica e Personalizzazione

Per la realizzazione grafica mi sono basato sullo stile futuristico adottato da Android 4.0, in modo da presentare all'utente un ambiente familiare e integrato con il resto del sistema.

Figura 5.5 – L'interfaccia Holo di Android 4.0

Ogni elemento dell'interfaccia è personalizzabile. L'applicazione è compatibile con i set di icone creati precedentemente per Launcher Pro e ADW Launcher. In questo modo, oltre a garantire da subito un grande assortimento di temi grafici, gli utenti potranno riutilizzare i prodotti scaricati dall'Android Market sul nuovo launcher.

6 Dettagli Implementativi

L’idea di partenza era quella di usare come base il codice sorgente del launcher predefinito di Android, in modo da dover riscrivere meno codice possibile; purtroppo ciò non si è rivelato facilmente praticabile. Il launcher di default è stato pensato come parte integrante del sistema e usa delle librerie non disponibili nel SDK, l’unica via per portarne a termine la compilazione è quella di disporre del codice di tutto il sistema.

Questo avrebbe significato aggiungere altro codice a quello già complesso del launcher e incorrere in probabili problemi di compatibilità con versioni diverse del sistema. Ho deciso allora di cominciare il progetto da zero.

In alcune fasi dello sviluppo mi sono servito della consultazione dei sorgenti di due progetti opensource:

android-launcher-for-sdk - Una versione modificata per essere compilabile con l’SDK pubblico del launcher di default della versione 1.6 di Android.
http://code.google.com/p/android-launcher-for-sdk

adw-launcher-android - Versione opensource di ADW Launcher Ex uno dei launcher più evoluti per Android. http://code.google.com/p/adw-launcher-android

Entrambi rilasciati sotto licenza Apache 2.0.

6.1 Le Activity

Le activity che compongono il progetto sono 3: workspace, drawer e preferences che corrispondono rispettivamente a home, menu delle applicazioni e preferenze. Da notare che home e menu delle applicazioni sono due activity separate; questa soluzione dà la possibilità, inedita tra i launcher attualmente disponibili, di poter avviare il menu principale (drawer) indipendentemente dal resto dell'applicazione (workspace), in modo da poter essere usato anche con altri launcher. Inoltre dà la possibilità al sistema di recuperare memoria chiudendo l'attività drawer quando necessario.

6.2 Le classi del progetto

Da precedenti esperienze con la progettazione di interfacce ho appreso che spesso capita di notare certe problematiche solo con l'uso del programma. Le soluzioni in questi casi possono richiedere di tornare a rivedere scelte progettuali che possono comportare importanti modifiche a livello implementativo; improntare il codice ad una forte modularità è stata da subito una chiara necessità.

La classe **Workspace** è l'activity principale, carica l'interfaccia della Home dal file workspace.xml, si occupa delle azioni associate ai tasti fisici ed effettua dei controlli all'avvio.

La classe **PanelsManager** si occupa della gestione dei pannelli che costituiscono la home, non fa alcuna assunzione né sulla loro tipologia né sul loro numero, si limita ad impostare il wallpaper e a fornire un metodo per passare da un pannello all'altro.

La classe **Panel** implementa un generico pannello, in particolare si occupa dell'inizializzazione di alcune variabili e del rilevamento dello swype, l'azione di trascinare un punto dello schermo verso destra o verso sinistra.

Le classi **WidgetPanel**, **StandbyPanel** e **TaskPanel** estendono la classe Panel e si occupano di disegnare le relative interfacce.

Il Flower è realizzato tramite la classe **Flower** che contiene per lo più funzioni di carattere geometrico per la determinazione della posizione degli oggetti che compongono l'anello e la classe **Bubble** che disegna i singoli elementi e ne associa le azioni.

La lista dei processi aperti è mostrata tramite un oggetto della classe **TaskList.**

La classe **Drawer** descrive l'activity che mostra il menu delle applicazioni, usa un elemento di classe **CategoryList** in cui vengono mostrate le categorie disponibili e un elemento di classe **IconGrid** che dispone in una griglia le icone delle applicazioni della categoria selezionata. Il meccanismo di drag&drop è fornito da un'apposita classe esterna.

6.2 Gli Adapter

In android non è possibile aggiungere in modo diretto View a elementi del tipo ListView o GridView, ma bisogna affidare a specifici oggetti questa mansione. Questi oggetti sono gli adapter. La loro necessità nasce per dare modo a ListView e GridView di caricare gli elementi dinamicamente, recuperando di volta in volta solo gli oggetti di cui ne viene richiesta la visualizzazione.

Per associare un adapter ad una View si usa il metodo *View*.**setAdapter()**. Nel mio progetto ho usato 3 adapter: **IconAdapter**, **CatAdapter** e **TaskListAdapter**.

La definizione di un adapter consiste nella creazione di una nuova classe che estenda la classe astratta *Adapter* e quindi nell'implementazione di alcuni metodi fondamentali. Oltre al costruttore, questi sono:

getCount(), restituisce il numero di elementi totali contenuti dalla view.
getView(), recupera le informazioni necessarie e crea le view che vanno a popolare la lista. Viene richiamato ogni volta che è richiesta la visualizzazione di un nuovo elemento.
getItem(), permette di risalire ad un oggetto quando questo è richiesto per compiere determinate azioni.

6.3 I database

L'applicazione fa uso di due database: uno relativo alle applicazioni installate e uno per gli elementi del Flower; la gestione è affidata rispettivamente alle classi **AppDatabase** e **FlowerDatabase**.

Il database delle applicazioni è necessario per memorizzare l'associazione con le categorie; ha anche una funzione di cache evitando di dover interrogare il gestore delle applicazioni ad ogni richiesta. Il database è aggiornato ad ogni nuova installazione o rimozione. Tale compito è svolto da un task asincrono descritto nella classe *AsyncDatabaseRefresh*() in modo da non generare attese.

Il database del Flower contiene le informazioni riguardo agli elementi selezionati dall'utente come azione associata, posizione ed icona.

6.4 Difficoltà incontrate

Ho incontrato diversi problemi durante lo sviluppo dell'applicazione. Innanzitutto si sono presentati problemi di performance nelle griglia delle applicazioni. Come spiegato precedentemente, Le *GridView* in Android sono ottimizzate in modo da caricare di volta in volta solo gli elementi visibili. Questo comportamento è di certo vantaggioso in caso di liste di elementi il cui recupero è impegnativo e per cui si prevede uno scorrimento lento. Nel mio caso invece, le icone sono immagini composte da poche centinaia di pixel e per trovare una determinata app è necessario spesso scorrere velocemente l'intera lista.

Ho risolto il problema creando una *hash map* che viene popolata all'avvio dell'activity in modo da ridurre le operazioni da effettuare nel metodo getView() e velocizzare il recupero delle view al momento del caricamento. Con questi accorgimenti l'avvio viene ritardato solo alla prima apertura del drawer creando non più di 1 secondo di attesa.

Un altro problema che ho dovuto affrontare è stato quello relativo alla gestione del touch. Quando un evento generato da un tocco coinvolge due o più elementi sovrapposti su più piani, ognuno di essi intercetta l'azione e spesso interferisce con gli altri.

Il problema si è verificato ad esempio con il drag-and-drop delle icone, che impediva lo scroll della griglia o con gli elementi sul desktop che causavano mal funzionamenti nello swype dei pannelli. Ogni caso ha richiesto un'analisi diversa del problema per definire i possibili casi di conflitto e stabilire per ognuno la precedenza tra le azioni.

6.5 La catalogazione automatica

Android non prevede la presenza di alcun indicatore della tipologia di un applicazione, all'interno della stessa. Per questo motivo è impossibile fornire un meccanismo di catalogazione automatica, che si basi solo sulle informazioni conosciute al sistema.

La soluzione da me proposta consiste nel lasciare inizialmente all'utente questo compito e nel raccogliere, con il consenso dello stesso, informazioni sulla catalogazione effettuata. Quando il database sarà abbastanza completo, sarà possibile ottenere da esso una catalogazione consigliata in base dati statistici.

Questo meccanismo è implementabile con la configurazione di un semplice server, che raccolga i dati, li inserisca in un database e su richiesta ne fornisca una rappresentazione facilmente trattabile dal dispositivo.

Conclusioni

L'idea di questo progetto è nata da una mia esigenza personale. Utilizzo ormai da qualche settimana una versione alpha dell'applicazione e ne sono sufficientemente soddisfatto.

Il programma è abbastanza completo da fornire l’insieme minimo di features richiesto per l’utilizzo quotidiano, ma mancano ancora alcune funzioni, fondamentali nell’ottica degli obiettivi per cui è stato costruito, come il pannello dei widget e la catalogazione automatica delle applicazioni. Dopodiché seguirà il lavoro di ottimizzazione e il miglioramento della compatibilità su dispositivi con caratteristiche particolari.

La modularità del codice ha permesso finora un’agile programmazione e ciò mi fa pensare che probabilmente si potrà avere una versione completa in poche settimane di lavoro.

L’app sarà resa disponibile sul Market in due versioni diverse, una gratuita contenente qualche limitazione ed una completa a pagamento.

Sarà il feedback degli utenti infine, a decretare il successo o il fallimento del progetto e a determinarne il suo sviluppo.

Riferimenti

[1] “Android@Mobile World Congress”, Andy Rubin
http://googlemobile.blogspot.com/2012/02/androidmobile-world-congress-its-all.html

[2] "Gartner Smart Phone Marketshare 2011 Q4". Gartner, Inc.
http://www.gartner.com/it/page.jsp?id=1924314, al 15/02/2012.

[3] “Application Fundamentals", Google,
http://developer.android.com/guide/topics/fundamentals.html

[4] "Platform Versions", Google
http://developer.android.com/resources/dashboard/platform-versions.html