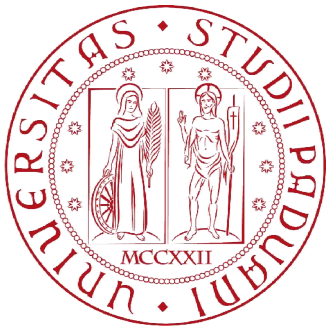
UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

Corso di laurea in Ingegneria Informatica



Una interfaccia utente perfezionata per lettori di ebook su dispositivi mobili

Relatore:
Carlo Fantozzi

Correlatore:
Alberto Pettarin

Laureando:
Valentino Giacometti

ANNO ACCADEMICO 2013-2014

Sommario

In questa tesi sarà presentata in prima istanza l'applicazione per la lettura di libri elettronici EPUB3Reader sviluppata durante il corso di programmazione di sistemi embedded da me e da altri due studenti di ingegneria informatica; dopodiché verranno illustrate le modifiche effettuate durante il periodo di tesi per rendere l'interfaccia, e quindi l'intera applicazione, più fruibile dall'utente medio a cui questa è indirizzata.

Indice

Introduzione

Il termine ebook deriva dalla contrazione di electronic book, cioè libro elettronico, e indica un qualsiasi documento di testo digitale. Un lettore di ebook è un dispositivo o un software che permette la lettura dei libri digitali.

Il progetto EPUB3reader nasce dalla constatazione che gli attuali software di lettura non offrono alcun vantaggio rispetto ad un libro tradizionale; essi implementano perlopiù funzioni scelte in base ai profitti che ne possono derivare e non ai vantaggi che ne può trarre l'utente.

EPUB3reader è un applicazione per Android che permette di leggere gli ebook, nel formato EPUB3, e che fornisce funzioni per la fruizione innovative; il fulcro dell'attuale applicazione consiste nel disporre di due aree visuali distinte in cui si possono visualizzare diversi passaggi dello stesso libro o ebook diversi o risorse remote come una pagina Web. Partendo da questa possibilità si sono sviluppate diverse opzioni avanzate: “Parallel Texts” permette di visualizzare nei due pannelli lingue diverse dello stesso libro e di sincronizzarne la consultazione in modo che quando viene girata la pagina di una lingua, automaticamente si girerà anche quella dell'altra. Un'altra funzione molto utile riguarda le note: se si tocca una nota questa verrà aperta nello stesso pannello dove è visualizzato il libro ma se il tocco sarà prolungato questa verrà aperta in un'altra area permettendo la consultazione della nota e del testo contemporaneamente.

Lo scopo della tesi è migliorare l'interfaccia utente di questa applicazione implementando nuove funzioni o perfezionando quelle esistenti. Tra le nuove funzioni si annoverano: la possibilità di cambiare le dimensioni dei due pannelli, quando questi sono entrambi visualizzati, in modo da dare maggiore rilevanza al primo o al secondo (nella versione precedente i pannelli occupavano lo schermo in modo paritario); la possibilità di visualizzare l'indice del libro, con collegamenti ipertestuali per accedere ai vari capitoli; la possibilità da parte dell'utente di cambiare: il font, la dimensione dei caratteri, l'interlinea, l'allineamento del testo, i margini e il colore sia dello sfondo che del font.

Per accedere a tutte queste nuove funzioni è stato ampliato e perfezionato il

menù che in questa versione è stato reso dinamico, correggendo un piccolo problema della versione di base che implementava un menù statico.
Nel prossimo capitolo si parlerà di che cos'è il formato EPUB3, dei dispositivi mobili, delle caratteristiche della piattaforma Android e infine dell'applicazione EPUB3reader, versione base. Il secondo capitolo sarà invece dedicato all'analisi del lavoro svolto in questa tesi, quindi verrà dedicata una sezione ad ogni nuova funzione implementata. Nel terzo e ultimo capitolo si trarranno le conclusioni sull'intero progetto.

Capitolo 1: EPUB, Android, EPUB3Reader

1.1. Da EPUB a EPUB3

Il mercato dei libri elettronici si divide tra ebook con formati proprietari come iBooks, KF8, ecc... ed EPUB formato aperto ma nato tecnologicamente obsoleto.

EPUB è uno standard per le pubblicazioni digitali e per i documenti basato sul modello Web. Venne adottato come modello ufficiale dall'IDPF (International Digital Publishing Forum) nel settembre del 2007, sostituendo il formato OEB, Open ebook, aggiornandolo e migliorandolo.

Il formato preso in esame è normato a sua volta da tre specifiche:

- l'Open Publication Structure (OPS) che norma la formattazione dei contenuti;
- l'Open Packaging Format (OPF) che descrive in xml la struttura del file .epub;
- l'OEBPS Container Format (OCF) che definisce il processo per la creazione dei file e la compressione dell'EPUB nel formato .zip

Come già detto EPUB si basa sullo standard web e ne eredita i linguaggi: si avvale del codice XHTML per strutturare le pagine di testo, dei fogli di stile CSS per layout e formattazione delle pagine e del linguaggio XML per manifest, indice dei contenuti e metadati, il tutto compresso in un archivio zip come da specifica.

Il formato EPUB, oltre al fatto già citato di essere libero e aperto, offre la possibilità di usare grafica vettoriale, che utilizza primitive geometriche come: punti, linee, curve e poligoni (basate su espressioni matematiche), per rappresentare le immagini permettendo così un'ottima resa qualitativa. Il testo viene adattato in base al dispositivo in cui è visualizzato (reflowing), variando la dimensione dei caratteri consentendo così una comoda lettura sia su schermi di

grandi dimensioni sia su dispositivi mobili, con schermi più piccoli. Inoltre l'intero stile del libro (dimensione dei caratteri, font, allineamento del testo e interlinea) è modificabile accedendo al file .css contenuto nel OEBPS.

I “metadati” e l'indice dei contenuti sono legati in modo indissolubile al libro, arricchendolo di informazioni aggiuntive, inoltre nel file EPUB possono essere presenti varie versioni dello stesso libro permettendo così agli editori di inserire il libro in varie lingue, in modo che lo stesso file possa essere distribuito in varie zone geografiche.

Contenuti audio e video sono integrabili nelle pagine di testo e viene dato pieno supporto al “Digital Rights Management” (DRM) una tecnologia usata dagli editori e dai detentori di copyright per controllare l'uso dei contenuti digitali dopo l'acquisto.

Nel ottobre del 2011 IDPF ha aggiornato le specifiche che definiscono il formato EPUB passando alla versione EPUB3 che si basa sulle tecnologie:

- XHTML5 grazie al quale è ora possibile inserire tag semantici (utilizzati ad esempio per le note a piè pagina).
- CSS3 che fornisce strumenti avanzati per la modifica dello stile dei contenuti.
- Javascript che permette funzionalità interattive avanzate come ad esempio la possibilità di inserire formule matematiche o grafici che prima venivano caricate sotto forma di immagini.

Con EPUB3 si è migliorata, inoltre, la gestione dei metadati e il supporto per dizionari e glossari; vengono gestiti modalità di scrittura diversi e i caratteri non romani(come quelli utilizzati nella lingua araba o cinese), ampliando ulteriormente il mercato di questo formato.

Queste innovazioni sono gestite da quattro nuove specifiche:

-EPUB Pubblications, che definisce la semantica, i metadati ed elenca tutte le risorse utilizzate per il funzionamento dell'EPUB, sostituendo l'OPF;

-EPUB Content Documents 3.0, che norma l'uso di HTML5, SVG, fogli di stile in CSS3 e sostituendo l'OPS;

-EPUB Media Overlays 3.0, è una nuova specifica che definisce gli oggetti multimediali del libro;

-EPUB Open Container Format 3.0 che non risulta modificato rispetto alla versione precedente.

All'interno di un file .epub, nella versione EPUB3, si trova:

- un file denominato "mimetype", un documento di testo ASCII, che permette a un' applicazione di riconoscere il tipo di file tramite un meccanismo a “numero magico”.
- Un file "container.xml", collocato in una cartella denominata "META-INF", che punta al file "content.opf".
- la cartella OEBPS che contiene, al limite in sottocartelle: tutti i file XHTML, i fogli di stile CSS, il file di navigazione dei contenuti nav.xhtml, il file content.opf ed eventuali file di tipo multimediale.

Se di un libro sono presenti più lingue le pagine di lingue diverse solitamente non si trovano in cartelle diverse ma sono identificabili tramite una sigla che le caratterizza in modo univoco per esempio "it" per le pagine in italiano o "en" per quelle in inglese; questo metodo se pur molto diffuso non è imposto dal modello.

Allo stato attuale lo standard presenta ancora delle lacune come l'inadeguatezza nella descrizione di grafica avanzata che ne esclude l'utilizzo per realizzare libri ad immagini, fumetti o libri ricchi di grafici e tabelle in generale.

1.2. La piattaforma Android

Una piattaforma è un insieme ben ordinato di pacchetti software, ottimizzato per le funzioni e per l'hardware a cui è destinato, nel caso di Android i dispositivi mobili. Questa include: il sistema operativo, un set di applicazioni di base ad esempio il browser web e un ricco set di librerie per lo sviluppo di applicazioni. A differenza delle altre piattaforme mobili Android è l'unica a licenza open source, questo uno dei motivi per cui è stata scelta per lo sviluppo dell'applicazione qui presentata.

L'architettura di Android e strutturata per livelli e si fonda sul Kernel linux, personalizzato, che contiene i driver software necessari per gestire le varie periferiche (ad esempio: display, videocamera, alimentazione...). Al livello

superiore si trovano le librerie che permettono l'implementazione di funzionalità a più alto livello, tra queste WebKit fornisce gli strumenti per il rendering e la navigazione web. Grazie a questa è possibile visualizzare, tramite la classe WebView, pagine XHTML, di cui gli EPUB sono composti. WebView fornisce un pannello per la visualizzazione delle pagine web e implementa le funzioni necessarie per: caricare un URL, navigare all'interno della pagina, selezionare pozioni di testo per la ricerca nel documento o in rete.

Altre librerie presenti in questo livello sono: "SQLite" che permette di utilizzare database relazionali con interfaccia SQL, "SGL" per realizzare grafica bidimensionale e "OpernGL|ES" per la grafica tridimensionale.

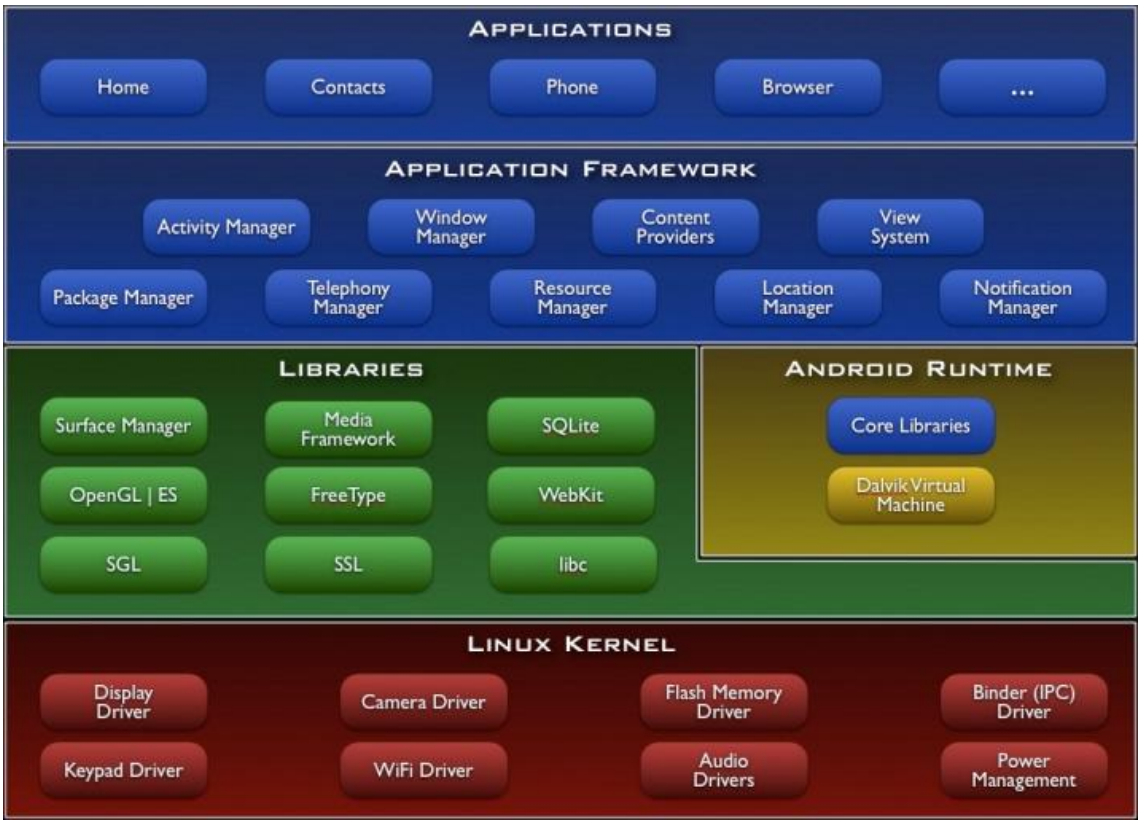


Figura 1: Architettura Android

Sopra alle librerie si trova la Java virtual machine, proprietaria, denominata “Dalvik”: le applicazioni Android infatti sono sviluppate in Java, con l'aggiunta di alcune librerie personalizzate, il codice prodotto con questo linguaggio viene poi compilato nel bytecode Java e successivamente in un formato proprietario (DEX); i file DEX, infine, vengono eseguiti dalla macchina virtuale Dalvik.

Il livello successivo contiene i framework di più alto livello, tra cui:

- “View System”: fornisce i mattoni per costruire i componenti dell'interfaccia utente.
- “Window Manager”: crea le finestre e invia all'applicazione gli eventi che riguardano l'interfaccia utente.
- “Activity Manager”: gestisce il ciclo di vita delle applicazioni.

L'ultimo livello le applicazioni base fornite dal sistema come il browser e l'agenda e quelle scaricate successivamente dall'utente.

Ogni applicazione è isolata dalle altre: viene eseguita in un proprio processo Linux con un proprio “user ID” e non può accedere ai componenti del sistema operativo. Questo inscatolamento fa sì che un malfunzionamento non si propaghi nel sistema o tra varie applicazioni; ciononostante le comunicazioni sono garantite tramite i “Content Providers”, contenitori di dati condivisi.

I processi possono segnalare se stessi al sistema come Content Provider di un insieme particolare di dati: quando queste informazioni sono richieste vengono richiamate da Android grazie ad un insieme specifico di API che permettono di agire sul contenuto secondo le specifiche definite.

Un'applicazione può avere componenti di quattro tipi diversi:

– Activity: fornisce una schermata con cui l'utente può interagire per compiere un'azione, come scattare una foto, mandare una mail o visualizzare una mappa. Per ogni activity viene fornita una finestra su cui disegnarne l'interfaccia; la finestra solitamente si adatta allo schermo, ma questo non è d'obbligo. Un'applicazione solitamente consiste in più activities collegate tra loro; tra queste viene definita una “main” activity, la prima ad essere visualizzata quando l'applicazione viene lanciata. Ogni activity può lanciarne un' altra per eseguire azioni diverse; ogni volta che una nuova activity inizia, e una si ferma, il sistema le preserva in uno stack (back stack) che implementa la politica “last in, first out”, così quando un'activity viene chiusa, premendo il tasto “back”, quella memorizzata precedentemente viene resa di nuovo visibile. Quando lo stato di una activity cambia questo viene notificato attraverso i metodi, callback, del ciclo di vita che forniscono l'opportunità di eseguire specifiche azioni in conseguenza al cambiamento di stato.

– Service: esegue le operazioni “long running” in background e non

fornisce un'interfaccia utente. Più servizi possono essere operativi in background e non terminano anche se l'utente cambia applicazione. In più, una componente può essere legata a un “service” per interagire con esso e eseguire comunicazioni tra processi (Interprocess Communication, IPC). Per esempio, un service può gestire la riproduzione di musica, eseguire le operazioni di I/O o interagire con un “content provider”.

– Content Provider: gestisce l'accesso a set di dati strutturati. Per far ciò, esso, incapsula i dati e ne fornisce le regole d'accesso. I content provider sono l'interfaccia standard per scambiare dati tra processi.

– Broadcast Receiver: permette di ricevere avvisi propagati a tutto il sistema tramite degli “intent”. Ad esempio quando il livello di carica della batteria scende sotto una certa soglia Android provvede ad inviare un avviso globale che può essere raccolto da una qualunque applicazione tramite i broadcast receiver.

Le Activities, i services e i broadcast receiver sono attivati dagli “intent”, i quali sono una descrizione astratta di una operazione da eseguire.

Le Activities attraversano diversi stati:

- Active (Running): l'activity è visibile su schermo e può ricevere input dall'utente.
- Paused: l'activity rimane parzialmente visibile ma non può ricevere input dall'utente.
- Stopped: l'activity non è visibile.

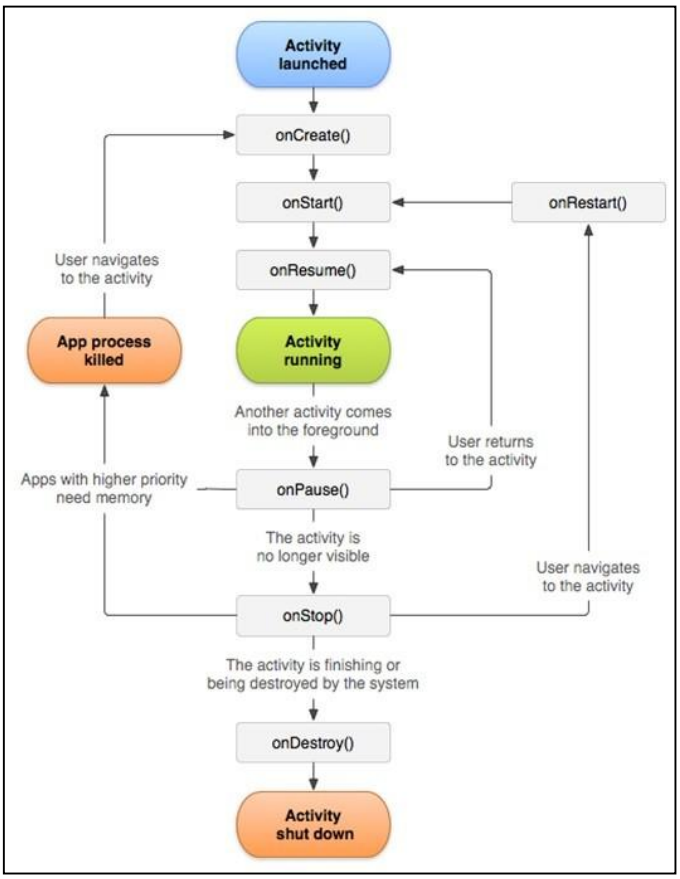


Figura 2: Ciclo di Vita

- Destroyed: l'activity è stata rimossa dalla memoria da Android.

Questi quattro stadi vengono attraversati da tutte le activities e ne costituiscono il ciclo di vita (lifecycle). Nel passare da uno stato all'altro vengono chiamati dei metodi(Callback methods) che permettono di eseguire alcune azioni fondamentali.

Durante il Callback method "onCreate" si inizializza l'activity: si carica la risorsa che definisce il layout dell'interfaccia utente tramite il metodo “setContentView” e utilizzando il metodo “findViewById" si crea un collegamento con i “widgets” che compongono l'interfaccia e con cui è necessario interagire in modo programmatico.

Durante il metodo “onPause()”, chiamato ogniqualvolta l'utente sta lasciando l'applicazione, è bene salvare lo stato dell'activity. Questa operazione è fondamentale per realizzare una buona applicazione in quanto in Android, come nelle altre piattaforme mobili, lo stato deve essere persistente, cioè anche se una activity viene terminata il suo stato deve essere salvato in modo che, quando una nuova istanza viene creata, questa riprenda dallo stesso punto in cui l'utente l'aveva abbandonata come se non si fosse mai chiusa. Ad avvalorare questa politica si aggiunge il fatto che l'activity viene distrutta anche quando cambia l'orientamento dello schermo, questo per poter riadattare il layout della stessa in base alla nuova orientazione.

Per quanto riguarda il layout di una activity e la creazione dei menù, Android come ogni altra piattaforma mobile, fornisce due metodi per realizzarli: programmatico e dichiarativo.

L'approccio programmatico consiste nel creare gli elementi dell'interfaccia utente nel codice dell'applicazione: grazie a questo si ha molta flessibilità e l'interfaccia utente può essere costruita a tempo d'esecuzione, ciononostante questa opzione è spesso scartata poiché porta alcune complicazioni tra cui: la necessità di ricompilare ad ogni modifica effettuata, difficoltà nel supportare vari linguaggi di programmazione e una promiscuità tra elementi di interfaccia e codice che può causare confusione ed errori.

Con l'approccio dichiarativo, invece, gli elementi dell'interfaccia utente sono elencati in una struttura dati esterna al codice, e collegati al codice in un

secondo momento, i benefici maggiori di questo metodo sono un miglior design, codice ed elementi di interfaccia ben distinti, e semplicità nell'effettuare modifiche, che non richiedono una ricompilazione.
Per applicare quest'ultimo approccio viene fornito un editor grafico che rende molto semplice la creazione di oggetti per l'interfaccia utente, questi vengono salvati in un file XML che viene compilato dentro “resource” che viene a sua volta caricato nel codice java.
Attraverso l'approccio dichiarativo e i file XML è possibile definire il layout non solo delle activities ma anche delle eventuali finestre di dialogo, inoltre consente di aggiungere o togliere opzioni al menù.
Ad ogni acrivity è legato un menù di azioni od opzioni. "Menu" è l'intefaccia utilizzata per gestire gli oggetti in un menù. Attraverso questa interfaccia è possibile aggiungere o togliere opzioni e gestire gli eventi causati dalla scelta di un opzione. Quando un utente preme su un oggetto che compone il menù il sistema chiama il metodo, presente nell'activity, onOptionsItemSelected(MenuItem) il quale restituisce un identificativo a cui si può associare una determinata azione.
Un altro strumento molto utile per la realizzazione di un'applicazione sono le finestre di dialogo. Queste vengono realizzate tramite la classe DialogFragment la quale fornisce gli strumenti necessari a mostrare su schermo la finestra e ne controlla in modo autonomo il ciclo di vita. Attraverso DialogFragment si ha il pieno controllo sulla finestra si può deciderne il contenuto, quando mostrala, nasconderla o cancellarla. Per implementare questa classe è necessario effettuarne l'Override e implementarne il metodo onCreateView(LayoutInflater, ViewGroup, Bundle) nel quale viene inizializzato il contenuto della finestra di dialogo. DialogFragment richiede come requisito minimo di funzionamento Android 3.0.
Per realizzare il layout delle finestre di dialogo e delle activities ci si avvale spesso di oggetti utili ad interagire con l'utente come: pulsanti, barre di avanzamento di stato, liste, caselle di testo e molti altri.
Questi widget sono definiti nel package android.widget attraverso delle classi che ne forniscono un'implementazione di base:

- Button è la classe utilizzata per definire un pulsante che può essere premuto dall'utente per scatenare un azione.
- ListView definisce una lista verticale di elementi selezionabili e ne implementa la navigazione.
- Toast fornisce una piccola finestra di dialogo con l'utente che viene mostrata sopra all'activity corrente per pochi istanti. Questo widget viene usato per mostrare messaggi di testo; ad esempio messaggi di errore.
- SeekBar implementa la barra di progresso. L'utente può interagire con la barra modificandone la progressione.
- Spinner definisce un menù a discesa (o spinner). Lo spinner contiene una lista di valori e quando viene premuto mostra l'intera lista e consente all'utente di sceglierne un elemento.

La dimensione di ogni oggetto che compone il layout è definita tramite tre parametri: larghezza, altezza e peso. La larghezza e l'altezza indicano il numero di pixel, in larghezza e in altezza, che l'oggetto occupa sullo schermo.
Il peso è un valore numerico che specifica l'ammontare di spazio rimanente che l'oggetto può occupare; per esempio se un layout è composto da due view una di peso 7 e una di peso 3 la prima occuperà 7/10 e la seconda 3/10 dello schermo.
Le componenti di un' applicazione sono dichiarate nel “manifest” (AndroidManifest.xml), nello stesso sono dichiarati anche i requisiti hardware e software dell'applicazione e i permessi richiesti dall'applicazione, ad esempio il permesso “CAMERA” permette all'applicazione di avere accesso alla fotocamera.

1.3. EPUB3reader

EPUB3reader è un'applicazione sviluppata da me e altri due studenti di ingegneria informatica dell'università di Padova durante il corso di “Programmazione di Sistemi Embedded” del professor Carlo Fantozzi.
Basandosi sull'idea dell'ingegner Alberto Pettarin, amministratore delegato della società ReadBeyond, si è sviluppata un'applicazione che non solo permette di visualizzare libri elettronici in formato EPUB3, e per retro-compatibilità EPUB2,

ma offre anche delle funzionalità avanzate e innovative.

Per far ciò si è data la possibilità all'utente di suddividere lo schermo in due pannelli permettendo così la lettura simultanea di due libri oppure di un libro e le sue note o ancora di un libro in due lingue diverse.

Per gestire e manipolare i file EPUB ci si affida alla libreria esterna EPUBLIB, sviluppata da Paul Siegmann sotto licenza GNU; questa permette di incapsulare l'ebook in un oggetto di tipo “Book” su cui compiere tutte le operazioni per visualizzare o gestire le varie parti del libro elettronico.

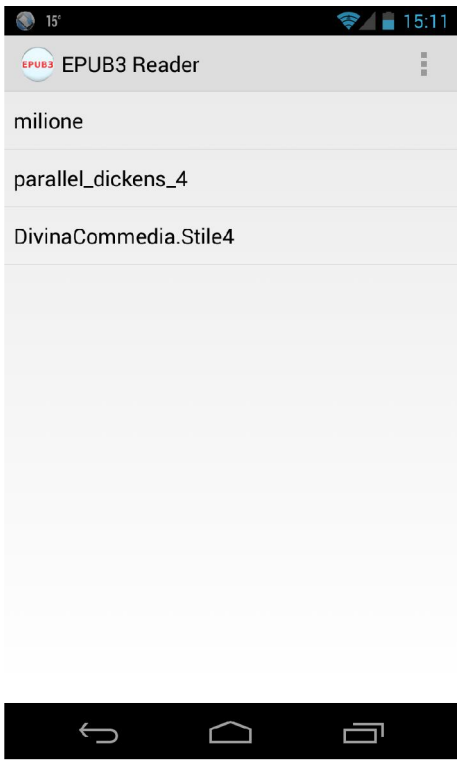


Figura 3:EPUB3Reader con due libri aperti

Il metodo "getContents" restituisce una lista contenente tutte le risorse del libro utilizzando "getHref" si ottiene l'indirizzo di ogni risorsa e infine attraverso quest'ultimo, insieme all'indirizzo assoluto dell'EPUB, si ottiene l'indirizzo completo da dare in ingresso alla webView che visualizzerà la pagina.

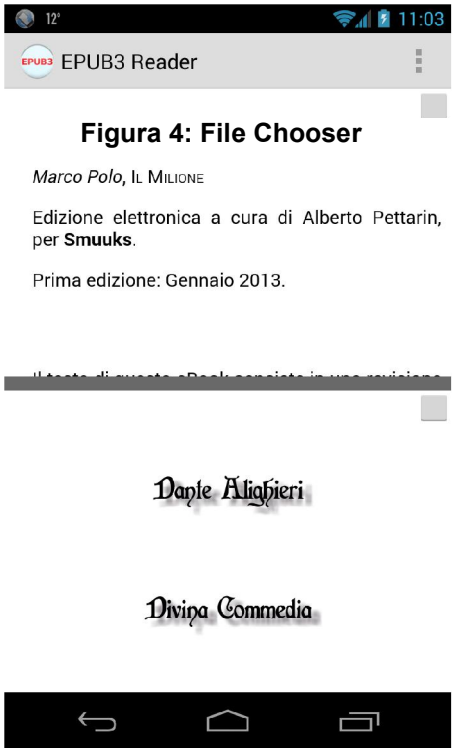


Figura 4: File Chooser

Le due activities su cui si basa l'applicazione sono: FileChooser e EpubReaderMain, il cui layout è implementato utilizzando il metodo dichiarativo.

Alla prima apertura dell'applicazione, e ogniqualvolta si debba scegliere un libro, viene inizializzata FileChooser che mostra sullo schermo del dispositivo la lista dei libri disponibili, aggiornabile tramite un'opzione del menu.

Figura 5:Menù dell'applicazione

Quando un libro viene selezionato,

semplicemente cliccando sul titolo dello stesso, l'activity si chiude e richiama l'altra: EpubReaderMain. La prima operazione compiuta da questa è la decompressione dell'archivio .ZIP dove il libro è contenuto. Una volta estratto il libro e i file ad esso associati questi vengono copiati in una cartella temporanea denominata “epubtemp”, nella quale, per evitare sovraccarichi, possono essere contenuti al massimo due libri; il numero massimo di libri è stato scelto in linea con il limite di EPUB visualizzabili contemporaneamente dall'applicazione.

Se un nuovo libro viene aperto mentre lo spazio in “epubtemp” è completamente occupato, questo prenderà il posto di quello non più in uso.

Una volta compiuta l'operazione di “unzip” il libro diviene utilizzabile e quindi visualizzato nello schermo del dispositivo tramite un pannello della classe WebView, la quale fornisce pieno supporto per la visualizzazione e la navigazione di file XHTML, formato con cui sono realizzate le pagine degli EPUB.

Per muoversi all'interno della pagina basta utilizzare lo “swipe verticale”, una gesture molto intuitiva e completamente supportata da WebView che consiste nel tenere il dito premuto sullo schermo e muoverlo dall'alto verso il basso per scorrere in giù la pagina o dal basso verso l'alto per scorrerla in su.

Per cambiare capitolo, invece, è stato realizzato il metodo “swipePage” che consente di voltare pagina utilizzando la gesture: “swipe orizzontale” definita, nel programma come lo scorrimento orizzontale del dito lungo lo schermo per almeno un quarto della larghezza della webView. Questo limite è stato scelto per evitare che “swipePage” venga richiamato erroneamente.

Associato ad ogni pannello, in alto a destra, si trova un piccolo pulsante quadrato e semi-trasparente adibito alla

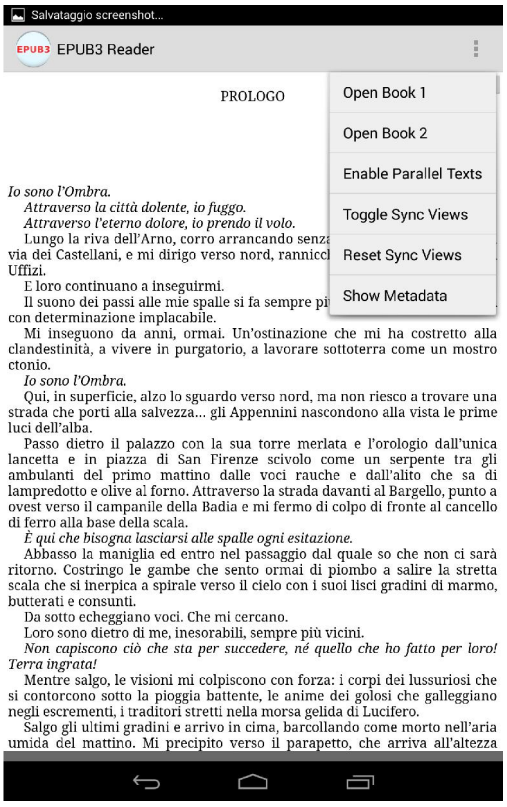


Figura 6:Menu

chiusura: esso oltre a chiudere il pannello elimina il libro dalla cartella temporanea “epubtemp” consentendo di avere all'interno di essa sempre il minor numero possibili di EPUB, in modo da occupare la minor quantità possibile la memoria del dispositivo.

La schermata principale oltre ai pannelli per la visualizzazione dei libri aperti è fornita di un ampio menù che offre diverse opzioni.

- Apertura di un primo libro: il libro visualizzato nel primo pannello verrà sostituito da un nuovo libro scelto attraverso “fileChooser”.
- Apertura di un secondo libro: se il secondo pannello non è visibile, verrà visualizzato con il contenuto richiesto, se la seconda vista era presente il suo contenuto verrà sostituito con il nuovo libro scelto.
- Abilitare la modalità Parallel Text: scegliendo questa opzione verrà visualizzato nello schermo un “DialogFragment”, cioè una finestra di dialogo con l'utente, nel quale sarà possibile scegliere di quali linguaggi fruire, al massimo due, tra tutti quelli presenti nell'EPUB. Compiuta questa operazione si ritornerà alla schermata principale con entrambi i pannelli aperti e su ognuno di questi una delle lingue scelte.

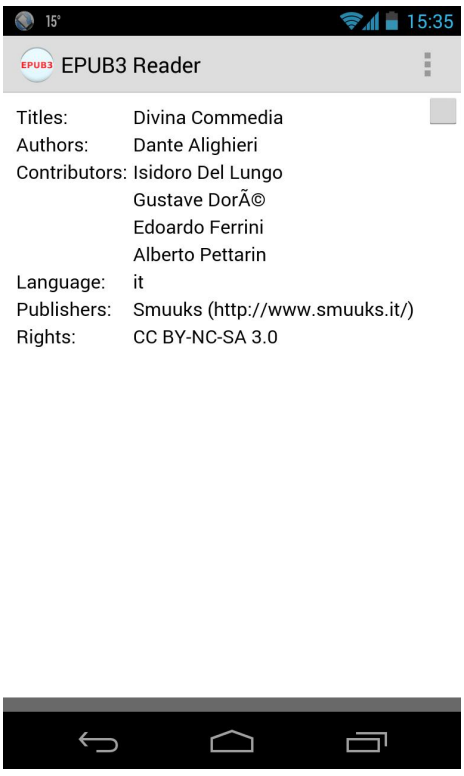


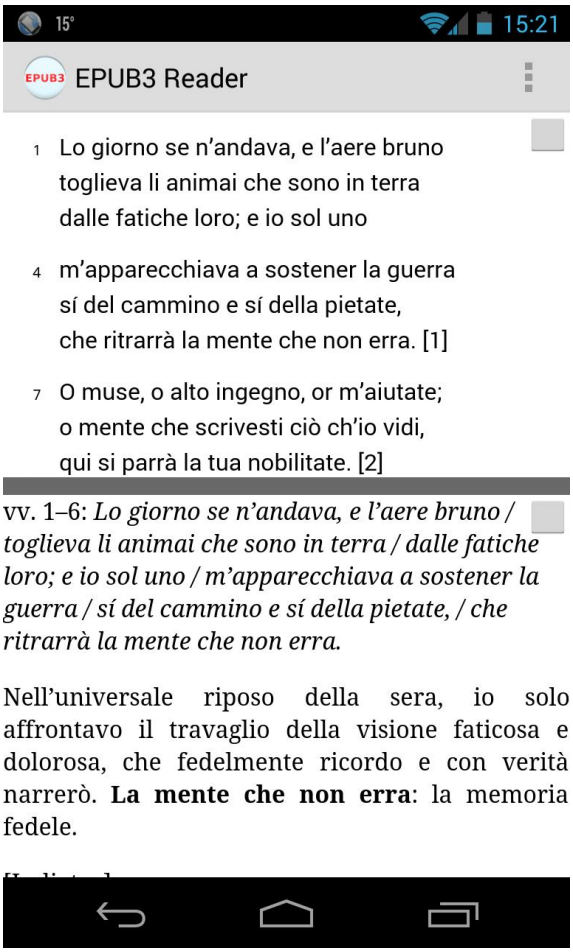
Figura 7: Visualizzazione dei metadati: titolo, autori, contributi, lingue disponibili, editori e diritti

- Abilitare la sincronizzazione delle pagine tra due libri: quando questa opzione è abilitata girare la pagina sul primo o sul secondo pannello sarà indifferente, perché entrambi i libri gireranno pagina. Questa opzione risulta particolarmente utile nel fruire della modalità Parallel Text.
- Allineare la pagina del primo libro con quella del secondo e viceversa:

premendo su questa opzione la pagina del primo libro si allineerà con quella del secondo o viceversa.

- Visualizzazione dei metadati: scegliendo di visualizzare i metadati, ovvero i tag identificativi di un'opera come il titolo o l'autore, di un libro questi verranno mostrati nel relativo pannello.

In ogni momento sarà possibile abbandonare l'applicazione, o con il tasto “back” del dispositivo o chiudendo tutti i pannelli aperti: quando questo avviene lo stato viene salvato. Per far ciò si crea un oggetto di tipo SharedPreferences, dove lo stato è effettivamente memorizzato, e un oggetto di Editor che svolge le attività di comunicazione tra SharedPreferences e l'applicazione. L'editor viene utilizzato per modificare i valori all'interno di SharedPreferences in modo sicuro infatti tutte le modifiche effettuate attraverso l'editor non vengono copiate all'originale sharedPreferences finché non viene chiamato il metodo commit(). In questo modo si è certi dell'integrità dei valori salvati in sharedPreferences in ogni momento. Quando l'applicazione verrà riaperta lo stato verrà caricato, accedendo a SharedPreferences, e visualizzato dando così l'impressione che l'applicazione non sia mai stata realmente chiusa.



vv. 1–6: *Lo giorno se n’andava, e l’aere bruno / toglieva li animai che sono in terra / dalle fatiche loro; e io sol uno / m’apparecchiava a sostener la guerra / sí del cammino e sí della pietate, / che ritrarrà la mente che non erra.*

Nell’universale riposo della sera, io solo affrontavo il travaglio della visione faticosa e dolorosa, che fedelmente ricordo e con verità narrerò. **La mente che non erra**: la memoria fedele.

Figura 8:Nota aperta con una pressione prolungata

Una importante funzione fornita da questa applicazione è la possibilità di scegliere “come” aprire un collegamento ipertestuale: qualora questo venga semplicemente premuto il contenuto del link verrà visualizzato nello stesso

pannello in cui è stato cliccato; se invece si applicherà una pressione prolungata verrà aperto nell'altro. Grazie a ciò si può consultare il testo dove era presente il link e il contenuto dello stesso contemporaneamente, questo può essere molto utile quando si vuole consultare una nota (figura 8).

Per l'apertura del link nello stesso pannello non è stato necessario implementare nuovi metodi poiché questa, corrispondendo alla normale apertura di un collegamento ipertestuale, è già completamente supportata dalla classe WebView. Invece per l'apertura nella view opposta si è dovuto creare un “OnLongClickListenter”, per ogni pannello, il quale resta in attesa dell'evento che corrisponde alla pressione prolungata di un collegamento e in risposta esegue le operazioni necessarie a visualizzare il contenuto nella view opposta a quella dove l'evento è stato catturato. L'applicazione, seppur innovativa e ricca di spunti per future evoluzioni, non è priva di difetti:

1) l'unzip dell''EPUB è poco efficiente e per libri ricchi di illustrazioni e altri file multimediali potrebbe impiegare un tempo considerevole.
2) il menù presenta tutte le opzioni in ogni momento anche se alcune hanno senso solo se due libri sono visualizzati. A questo problema è stata data una soluzione temporanea inserendo dei messaggi di errore nel caso venga scelta un'opzione non utilizzabile.
3) la TOC non è visibile dall'utente anche se presente nel libro.
4) le impostazioni grafiche e il layout delle pagine non sono modificabili.
5) la dimensione dei due pannelli è fissa.
6) si consente di scegliere più di due lingue per la modalità “Parallel Text”, anche se poi solamente due verranno visualizzate(figura 9).

L'applicazione è attualmente disponibile pubblicamente su GitHub. Il codice sorgente è stato rilasciato sotto licenza MIT(open source e free software), in modo che chiunque possa visualizzare ed in futuro contribuire al progetto.

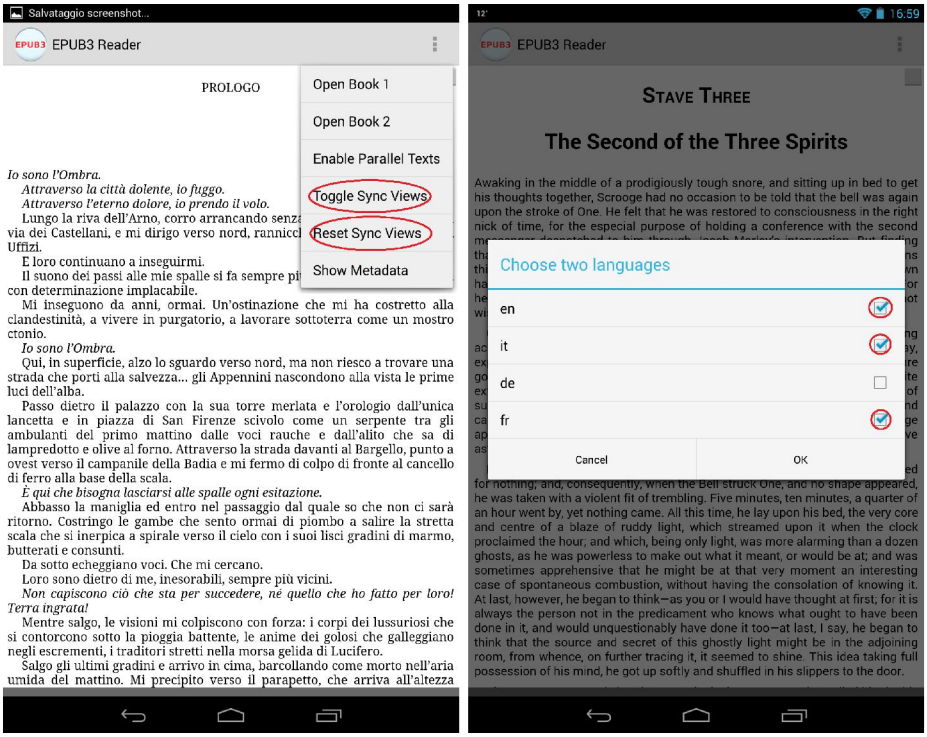


Figura 9: Errori: a sinistra menù statico, a destra tre lingue selezionate quando il limite reale è due

Capitolo 2: Novità nell'interfaccia utente per EPUB3reader

2.0 Introduzione

Per migliorare l'esperienza nell'utilizzo di EPUB3reader si è deciso di aumentare le potenzialità e di correggere alcuni dei problemi dell'applicazione. In particolare si vuole:

- dare maggiore possibilità di personalizzazione permettendo di modificare in modo completo lo stile di ogni pagina di qualsivoglia libro e di variare la dimensione dei pannelli in base alle proprie preferenze.
- aumentare le potenzialità della funzione “Parallel Text” implementando lo scorrimento verticale sincrono delle pagine.
- dare accesso a tutte le informazioni relative al libro premettendo all'utente di visualizzare l'indice dei contenuti.

2.1 Table of Contents

La Table of Contents, TOC, è un albero di riferimenti ai capitoli del libro e corrisponde all'indice analitico di un libro tradizionale con l'aggiunta che questa offre collegamento diretto ai capitoli.

Per accedere alle informazioni che costituiscono la TOC è stato sufficiente affidarsi ai metodi della già citata libreria EPUBLIB. In particolare con il metodo getTableOfContents() si ottiene un oggetto di tipo TableOfContents e con getTocReferences() si ottiene la lista di riferimenti al testo per i nodi di primo livello. Per accedere ai nodi di livello inferiore si richiama su ogni nodo della lista creata precedentemente il metodo getChildren() e si procede in modo ricorsivo all'attraversamento dell'intero albero.

Ottenuti in questo modo i riferimenti, si è proceduto a inserirli in una stringa di codice HTML utilizzango i tag per le liste ordinate <ul> e <li> in modo da esplicitare la struttura gerarchica caratteristica degli alberi.

Una volta formata la stringa questa viene scritta in un file .html denominato

Toc.html. Terminata la scrittura il file viene salvato nella stessa cartella in cui è memorizzato l'epub; in questo modo le informazioni necessarie a visualizzare la Table of Contents sono sempre disponibili.

```
public void createTocFile() {
	List<TOCReference> tmp;
	TableOfContents toc = book.getTableOfContents();
	String html = "<html><body><ul>";
	tmp = toc.getTocReferences();
	if (tmp.size() > 0) {
		html += getS(R.string.tocReference);
		for (int i = 0; i < tmp.size(); i++) {
			String path = "file://" + location + decompressedFolder +
			"/"+ pathOPF + "/" + tmp.get(i).getCompleteHref();
			html += "<li>" + "<a href=\"" + path + "\">"
				+ tmp.get(i).getTitle() + "</a>" + "</li>";
			List<TOCReference> children = tmp.get(i).getChildren();
			for (int j = 0; j < children.size(); j++)
				html += r_createTocFile(children.get(j));}}
		html += getS(R.string.tablebodyhtmlClose);
		// si scrive il file html
		String filePath = location + decompressedFolder + "/Toc.html";
		try {File file = new File(filePath);
			FileWriter fw = new FileWriter(file);
			fw.write(html);
			fw.flush();
			fw.close();}
			catch (IOException e) {
			e.printStackTrace();}}
```

Per attraversare l'albero si è utilizzata la navigazione in pre-ordine in modo che i capitoli e i sottocapitoli siano visualizzati in modo corretto e ordinato.

```
public String r_createTocFile(TOCReference e) {
	String childrenPath = "file://" + location + decompressedFolder + "/"
				+ pathOPF + "/" + e.getCompleteHref();
	String html = "<ul><li>" + "<a href=\"" + childrenPath + "\">"
```

```
                + e.getTitle() + "</a>" + "</li></ul>";
        List<TOCReference> children = e.getChildren();
        for (int j = 0; j < children.size(); j++)
                html += r_createTocFile(children.get(j));
        return html;}
```

Per visualizzare la TOC su schermo si è aggiunta, al menù dell'applicazione, un'opzione, “Table of Contents”, tramite cui si può scegliere se vedere la TOC del primo o del secondo libro, nel caso questo esista.

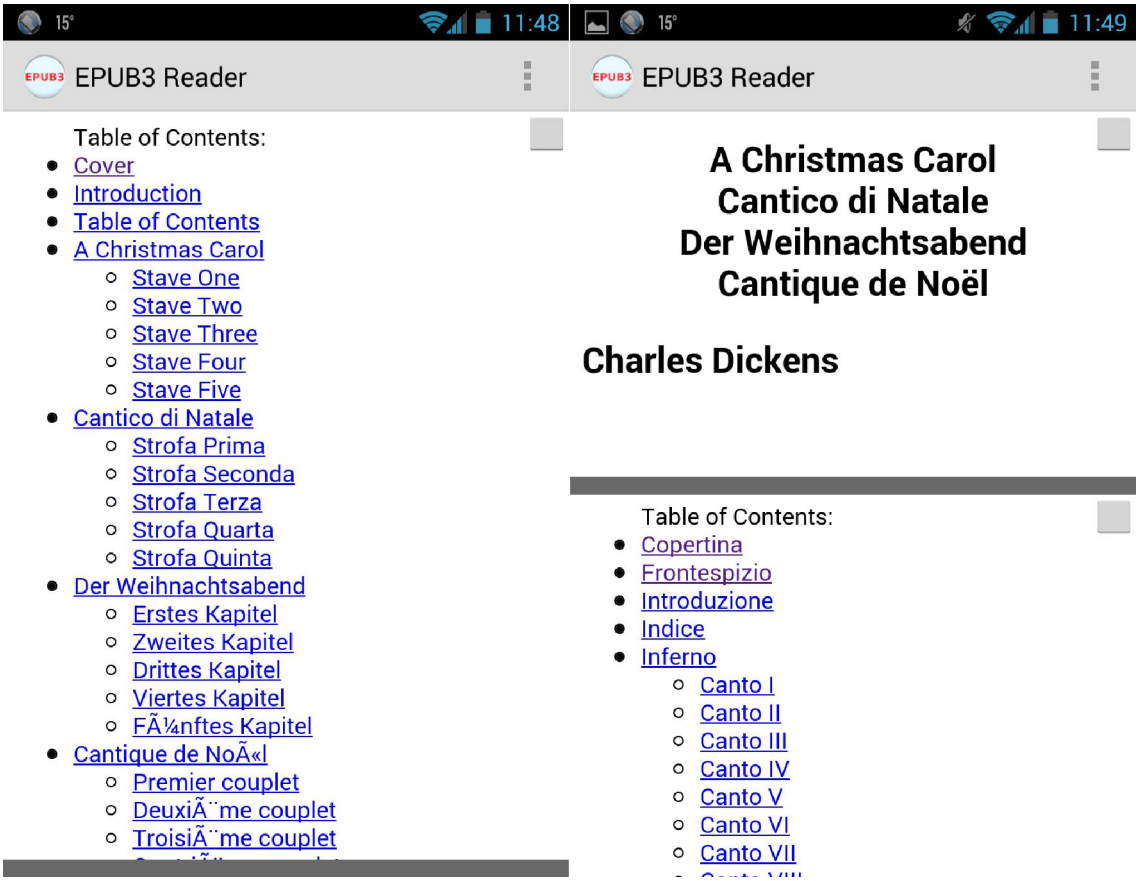


Figura 10:A sinistra la TOC del libro "A Christmas Carol" a sinistra la TOC della "Divina Commedia" aperta come secondo libro

2.2 Pannelli Regolabili

Nella sezione, 1.3, si sottolineava che nella prima versione dell'applicazione i pannelli erano di dimensione fissa e occupavano ciascuno il 50% dello spazio disponibile. Si vuole, ora, dare la possibilità all'utente di poter regolare i pannelli a propria discrezione. Per far ciò si è creata una nuova opzione nel menù denominata “Change Panel Size”: attivandola verrà visualizzata una finestra di dialogo, il cui layout è definito tramite un file XML, con una “seekBar” che fornisce un pratico modo per cambiare la dimensione del primo pannello, la dimensione del secondo si adatterà in base a quella del primo.

Per evitare valori estremi, che potrebbero generare effetti indesiderati, in particolare che il primo pannello occupi tutto lo schermo o che non ne occupi nessuna parte, si è imposto che la dimensione non possa andare oltre dei valori limiti impostati al 10% e al 90% dello schermo.

Per come sono stati definiti i pannelli nella versione originale di EPUB3reader per poterne cambiare la dimensione in modo dinamico si è dovuto accedere al peso di questi oggetti.

Purtroppo nella piattaforma Android non è possibile accedere direttamente al solo peso degli oggetti di layout e quindi si è dovuto utilizzare un metodo che modifica tutti e tre i valori caratteristici di tali oggetti: altezza, larghezza e peso.

Questo comporta il dover distinguere l'orientamento del dispositivo (portrait o landscape) poiché i parametri di altezza e larghezza cambiano in base a questo.

```
protected void changeViewsSize(float weight1) {
firstViewSize = weight1;
if (getResources().getConfiguration().orientation == Configuration.
ORIENTATION_LANDSCAPE)
      {weight1 = weight1 + (float) 0.01; //value fix
      LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(
                              0, LayoutParams.MATCH_PARENT, weight1);
      layoutView1.setLayoutParams(params1);
      LinearLayout.LayoutParams params2 = new LinearLayout.LayoutParams
                              0, LayoutParams.MATCH_PARENT, (1 - weight1));
      layoutView2.setLayoutParams(params2);
}
```

```
else {
      LinearLayout.LayoutParams params1 = new LinearLayout.LayoutParams(
                                    LayoutParams.MATCH_PARENT, 0, weight1);
      layoutView1.setLayoutParams(params1);

      LinearLayout.LayoutParams params2 = new LinearLayout.LayoutParams(
                                    LayoutParams.MATCH_PARENT, 0, (1 - weight1));
      layoutView2.setLayoutParams(params2);}}
```

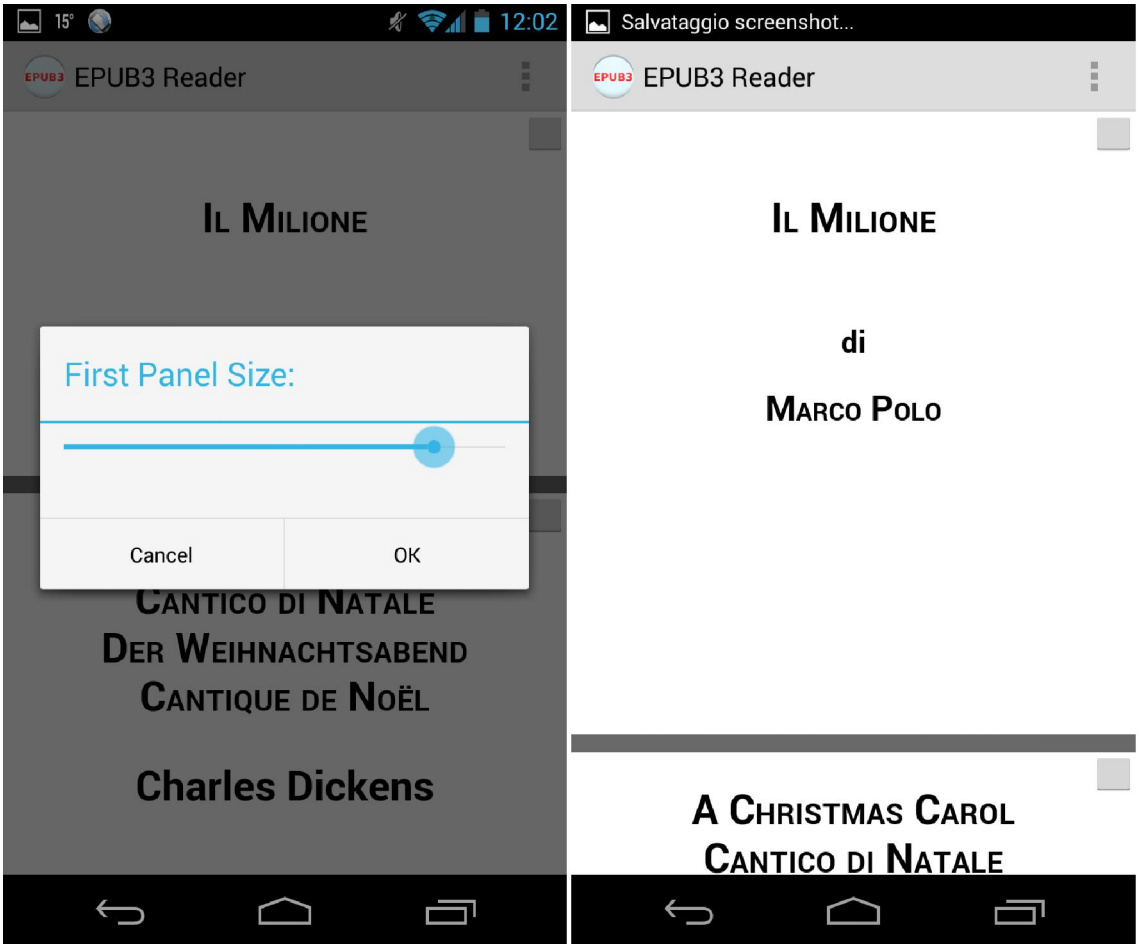


Figura 11: Pannelli regolabili. A sinistra la finestra di dialogo con l'utente, a destra i pannelli una volta regolati

2.3 Impostazioni grafiche regolabili

Le impostazioni grafiche di ogni pagina in un EPUB sono definite tramite un file CSS. Per poter consentire all'utente di modificare l'aspetto delle pagine si è deciso di iniettare dinamicamente uno <style> nel sorgente XHTML della pagina prima di passarlo alla WebView: in questo modo lo stile originario della pagina non viene corrotto e si ottiene comunque il risultato desiderato.

Per impostare le proprie preferenze l'utente, dopo aver scelto su quale libro applicare le modifiche, ha a disposizione una finestra di dialogo con varie opzioni.

- Font Color: consente di scegliere quale colore deve avere il testo;
- Background Color: imposta il colore dello sfondo;
- Font Family: imposta la famiglia del font con cui verrà visualizzato il testo le opzioni sono: Sans Serif, Serif e Monospace.
- Font Size: permette di cambiare la dimensione delle lettere che compongono il testo offrendo un range che va dal 90% della dimensione prevista dal CSS al 300%.
- Line Height: consente di ampliare o ristringere lo spazio di interlinea.

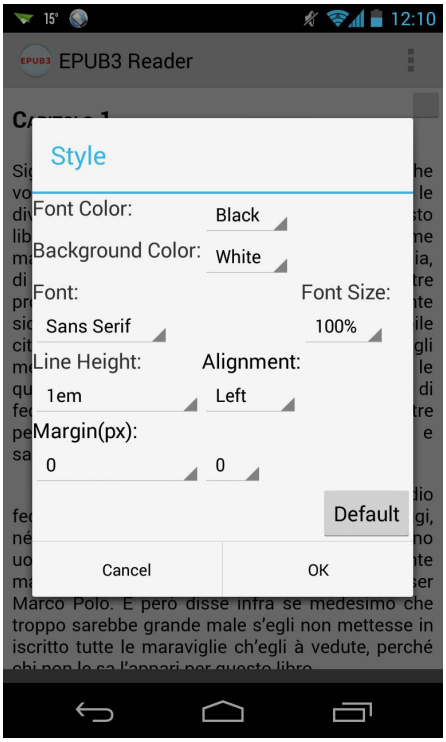


Figura 12: finestra di dialogo per la scelta dello stile

- Alignment: permette di scegliere l'allineamento del testo. Le opzioni sono: Left, Center, Right e Justify.
- Margin: imposta la dimensione dei margini destro e sinistro del testo il range di opzioni spazia dallo 0% al 25% della dimensione del pannello.

Oltre a queste opzioni viene offerta all'utente la possibilità di azzerare le modifiche effettuate e ritornare allo stile originario del libro tramite un pulsante di reset.

Una volta impostate le proprie preferenze verrà invocato il metodo della classe EpubNavigator changeCSS al quale si passa su quale libro si è scelto di operare.
Questo metodo compie due operazioni: la prima consiste nel richiamare un secondo metodo denominato addCSS e la seconda consiste nel ricaricare la pagina attuale per applicare le modifiche operate tramite addCSS.

```
	public void changeCSS(BookEnum which) {
		if (which == BookEnum.first) {
			book1.addCSS(EpubReaderMain.getSettings());
			loadPageIntoView1(pageOnView1);
		}
		if (!exactlyOneBookOpen && which == BookEnum.second) {
			book2.addCSS(EpubReaderMain.getSettings());
			loadPageIntoView2(pageOnView2);
		}
	}
```

Il metodo addCSS crea una stringa in linguaggio CSS contenente le preferenze del utente, passate tramite un array di stringhe, e la inietta nel codice di tutte le pagine del libro.

```
public void addCSS(String[] settings) {
	// CSS
		String css = "<style type=\"text/css\">\n";
	//settings[0] contiene il colore del font
		if (!settings[0].isEmpty()) {
			css = css + "body{color:" + settings[0] + ";}";
			css = css + "a:link{color:" + settings[0] + ";}";
		}
. . . //tutte le altre preferenze vengono inserite in modo simile
		css = css + "</style>";
	//iniezione del css nelle pagine
	for (int i = 0; i < spineElementPaths.length; i++) {
		String path = spineElementPaths[i].replace("file:///", "");
		String source = readPage(path);
		source = source.replace(actualCSS + "</head>", css + "</head>");
	//il nuovo css sostituisce il precendente css </head> è necessario nel
```

```
//caso in cui actualCSS sia una stringa vuota
        writePage(path, source);
        }
        actualCSS = css;
```

Se l'utente preme il pulsante di reset presente nella finestra di dialogo, al metodo addCSS viene passato un array di stringhe vuote e la stringa "css" vuota sostituisce quella iniettata precedentemente, eliminando così ogni variazione di stile effettuata.

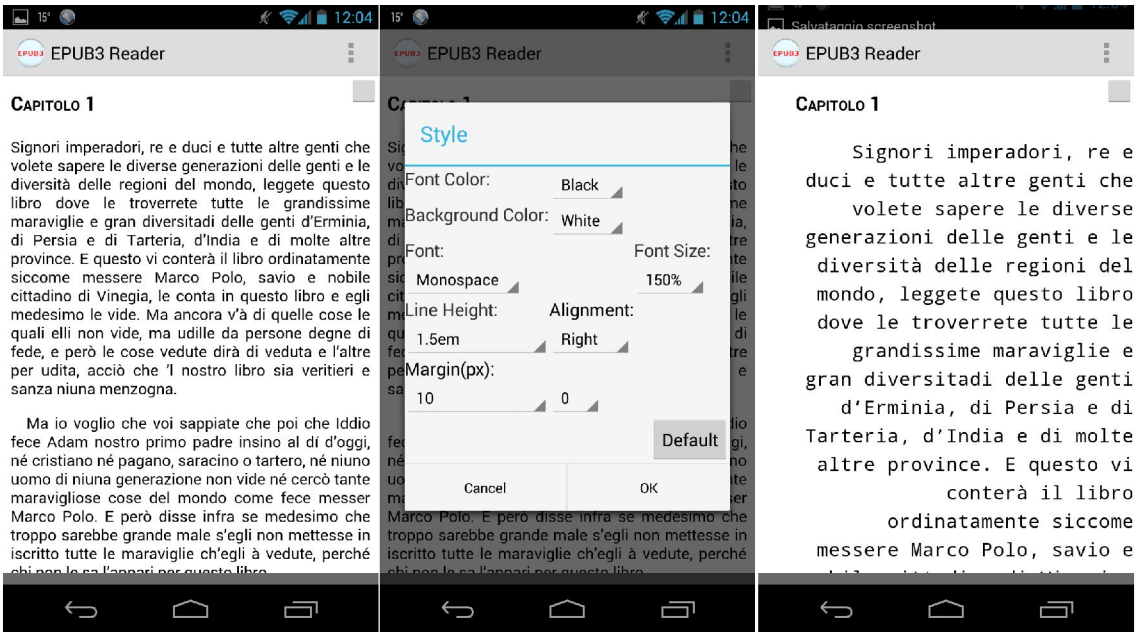


Figura 13:A sinistra la pagina originale, al centro la finestra di dialogo con le impostazioni da inserire, a sinistra la stessa pagina dopo il cambio di stile

2.4 Scorrimento univoco

Per ampliare le potenzialità della modalità “Parallel Text” si è voluto concedere all'utente l'opportunità non solo di cambiare sezioni con un sol gesto ma anche di poter sincronizzare lo scorrimento verticale.

Per far ciò si deve catturare l'evento "scroll sulla prima view" e passare l'intervallo di valori raccolto, punto iniziale e finale in pixel, al metodo "ScrollTo" che applicherà al secondo pannello la variazione di pixel calcolata.

Questa soluzione porta con sé due problematiche.

1. I due testi essendo in lingue diverse, pur avendo lo stesso contenuto,

sono di lunghezza diversa.

2. Risulta impossibile catturare l'accelerazione che viene impressa durante lo scroll.

Per ovviare al problema delle lunghezze diverse è stato sufficiente applicare un fattore di scala, calcolato come lunghezza del testo del secondo pannello diviso lunghezza del testo dove è stato effettuato lo scroll, al valore passato al metodo "ScrollTo".

All'impossibilità di catturare l'accelerazione si è posto rimedio catturando la variazione di posizione nella prima view a intervalli di tempo regolari. Per far ciò si è creato un thread nel quale in un ciclo prima si mette a dormire il thread stesso e poi, al suo risveglio, si applica al secondo pannello il nuovo range di valori registrato.

```
protected void syncScroll(View v1, View v2, MotionEvent event) {
	int action = MotionEventCompat.getActionMasked(event);
	final WebView wv1 = (WebView) v1;
	final WebView wv2 = (WebView) v2;
	final float DiffContent = ((float)wv2.getContentHeight() / (float)wv1
							.getContentHeight());
	final int wv1Height = wv1.getContentHeight();
	final int wv2Height = wv2.getContentHeight();

	switch (action) {
	case (MotionEvent.ACTION_DOWN):
		swipeOriginY = event.getY();
		swipeOriginX = event.getX();
		Thread t = new Thread();
		new Thread(new Runnable() {
		public void run() {
			try {
			while (wv1.isPressed()) {
				Thread.sleep(18);
				//passo alla seconda webView l'intervallo di scroll
				//scalato del rapporto tra le altezze del contenuto
				//delle due WebView
				wv2.scrollTo(0, (int) ((float) wv1.getScrollY()
```

```
                        / (float) wv1Height * (float) wv2Height));
                for (int i = 0; i < 2500 / 30; i++) {
                    Thread.sleep(30);
                    wv2.scrollTo(0, (int) ((float) wv1.getScrollY()
                        / (float) wv1Height * (float) wv2Height));
                            }
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                }}}).start();
            break;
```

La soluzione al secondo problema seppur tecnicamente corretta presenta ancora dei limiti: lo scroll sulla seconda vista procede a scatti e risulta fastidioso alla vista. Questi difetti sono sotto analisi e verranno sicuramente rimossi in una futura versione dell'applicazione. Nel frattempo si è deciso di non consentire all'utente l'attivazione di questa modalità.

2.5 Nuovo menù e aggiustamenti

A fronte dell'aggiunta di nuove opzioni si è dovuto ampliare il menù inserendo le voci:

- Table of Contents da accesso alla TOC del libro.
- Change Panel Size permette di variare le dimensioni dei due pannelli.
- Change Style, tramite cui si accede alla schermata per il cambio di stile del libro.

Oltre ad aver aggiunto queste nuove opzioni si è voluto correggere il difetto che il menù presentava nella versione precedente rendendolo dinamico, ovvero variando le opzioni disponibili in base allo stato dell'applicazione e in particolar modo al numero di libri visualizzati.

Per far ciò si è sovrascritto il metodo "onPrepareOptionMenu", richiamato ogni qual volta si prema sul menù, per far si che la visibilità delle voci vari.

```
@Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        if (navigator.isParallelTextOn() == false
                && navigator.isExactlyOneBookOpen() == false) {
            menu.findItem(R.id.meta1).setVisible(true);
```

```
...

if (navigator.isExactlyOneBookOpen() == true
                        || navigator.isParallelTextOn() == true) {

                menu.findItem(R.id.meta1).setVisible(false);
```

Si è effettuata una ulteriore correzione eliminando, come incoraggiato nella piattaforma Android, le stringhe inserite nel codice sorgente spostandole nell'apposito archivio R.String e richiamandole quando necessario tramite il metodo "getString".

```
errorMessage("Cannot Load Page!")                    //Prima
errorMessage(getString(R.string.error_LoadPage));    //Dopo
```

Questo modo di operare porta con sé un grosso vantaggio in robustezza poiché se una stringa deve essere usata più volte basta richiamarla e non riscriverla, fattore che può generare errori.

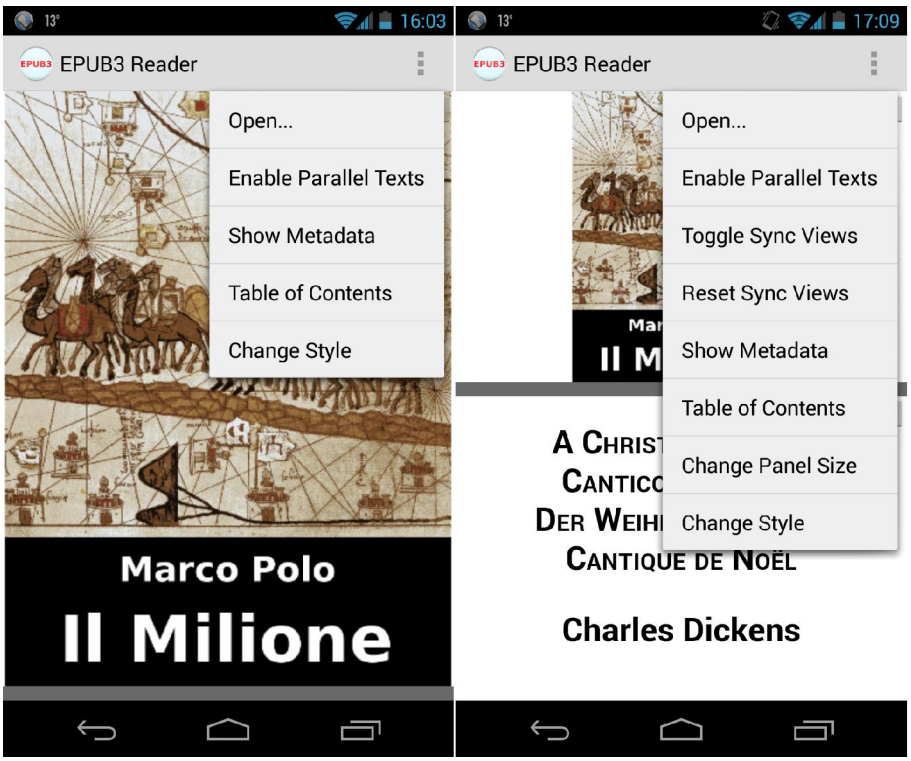


Figura 14: Menù dinamico

Capitolo 3: Conclusioni

L'applicazione allo stato attuale rimane un prototipo che si pone come esempio di ciò che gli EPUB offrono in termini di possibilità di visualizzazione ma che attualmente non viene sfruttato.

La strada per renderla commercializzabile è ancora lunga e molti dei problemi che aveva in origine non sono ancora stati risolti, ciononostante un buon passo avanti è stato effettuato rendendo l'interfaccia utente più completa e pulita.

Si spera in questo modo che il progetto continui a suscitare interesse (già nella versione di base si era guadagnato qualche articolo in blog specializzati) e che venga presto valorizzata dalla comunità in modo da stimolare nuovi sviluppi.

Proprio per renderla disponibile alla comunità, EPUB3Reader con le modifiche illustrate in questo documento ha sostituito la precedente versione presente sul sito di code-sharing GitHub.

Sitografia

- http://idpf.org/epub sito ufficiale dell'international Digital Publisching Forum nel quale è definito lo standard EPUB in tutte le sue versioni.
- http://www.ebookreaderitalia.com/epub3reader-unapp-android-per-leggere-ebook-epub-3-0-complessi/ articolo del forum eBookReaderItalia.com riguardante l'applicazione EPUB3Reader.
- http://developer.android.com/develop/index.html home page del sito offerto come supporto agli sviluppatori da Android.
- http://www.smuuks.it/index.php/en/ sito della società di cui Alberto Pettarin è co-fondatore e da cui sono stati tratti i libri per testare l'applicazione.
- https://github.com/pettarin/epub3reader pagina di GitHub su cui è stato pubblicato EPUB3Reader.
- https://readbeyond.it/ sito web della s.r.l. di cui Alberto Pettarin è amministratore delegato