RELAZIONE PROGETTO SVILUPPO DI APPLICAZIONI MOBILI

Gabriele Masciotti 578318 a.a. 2020/2021

In questo elaborato riporto una breve relazione sul progetto di fine esame.
L'applicazione che ho sviluppato consente di gestire la creazione, memorizzazione, modifica e cancellazione di note (appunti) che l'utente può personalizzare a suo piacimento aggiungendo immagini e disegni fatti a mano. Nel seguito illustro in dettaglio il funzionamento dell'app, le varie fasi di esecuzione del programma e i suoi componenti.

1. OVERVIEW

Partendo dall'analisi del Manifest di progetto si evince che l'applicazione si compone sostanzialmente di **tre activities**. La principale (***MainActivity***) è quella in cui è mostrata all'utente la lista delle note salvate e che consente di crearne di nuove; la seconda (***note_details_activity***), figlia della principale, mostra all'utente la nota nella sua interezza, con tanto di immagini e disegni; la terza (***DrawingActivity***), figlia di note_details_activity, permette di creare disegni con le dita da inserire nella nota.
L'ultimo componente del manifesto della app è il file provider, il quale viene utilizzato per condividere le note che contengono immagini in altre app (come vedremo in dettaglio nelle sezioni successive di questo scritto).

2. MAIN ACTIVITY E NOTES ADAPTER

Il semplice layout dell'activity principale dell'applicazione si compone di una **RecyclerView** ed un **FloatingActionButton** (utilizzato per creare una nuova nota).
Vale senz'altro la pena parlare subito dell'Adapter utilizzato dalla prima, che praticamente contiene tutta la logica interessante.
La main activity crea un oggetto di tipo ***NotesAdapter***, passando come parametri al costruttore un riferimento al proprio contesto e uno al database delle note.
La classe NotesAdapter implementa l'adapter utilizzato dalla RecyclerView per mostrare all'utente la lista delle note che egli ha preventivamente creato e salvato. Per farlo si interfaccia con il database delle note chiedendo un nuovo "cursor", scorrendo il quale è poi in grado di costruire la lista di view, ciascuna delle quali presenta le caratteristiche di una nota (titolo e inizio del testo).
Come si nota, la classe implementa le due interfacce "*OnClickListener*" e "*OnLongClickListener*". Quando la view di una nota viene tenuta premuta dall'utente (long click), il programma mostra una ***action bar contestuale*** al disopra della *toolbar* della home page, in cui è visibile il numero delle note attualmente selezionate e viene offerta la possibilità di eliminarle. Dopo aver ricevuto la conferma di voler procedere con la cancellazione (espressa dall'utente tramite l'apposito *Dialog* mostrato), il programma elimina le note selezionate, e le immagini in esse contenute, dal database. Al tap sulla view invece viene aperta la pagina che mostra i dettagli della nota.

3. ACTIVITY CON DETTAGLI DELLE NOTE

Anche il layout dell'activity note_details_activity è piuttosto semplice; si compone di una toolbar con i pulsanti di azione e una scroll view che presenta due edit text, per il titolo e il testo della nota.
Un po' meno semplice è il suo funzionamento dinamico. Innanzitutto il programma distingue due modalità di visualizzazione dell'activity, ***modalità typing*** e non-typing. Questa distinzione viene effettuata al fine di cambiare dinamicamente il tipo di azioni che l'utente può compiere utilizzando i pulsanti del menu della toolbar sopra citata e per rendere le due edit text modificabili o meno a seconda della modalità. All'apertura della finestra dell'activity dopo il tap sul floating button dell'activity main, ad esempio, la modalità sarà impostata su typing, al contrario di quanto avviene quando l'utente vuole aprire una nota già creata e salvata toccando la sua view. In ogni momento

l'utente può ovviamente cliccare il pulsante di modifica per abilitare la modalità typing e poter quindi modificare la nota.
Un ulteriore controllo aggiuntivo che viene eseguito al momento della creazione dell'activity è se si sta creando una nuova nota oppure si stanno visualizzando i dettagli di una già presente nel database, al fine di poter scegliere se creare un nuovo record o modificarne uno già esistente, rispettivamente.
A questo punto avviene il ***caricamento della nota***. Se in essa sono presenti delle immagini viene creato un nuovo **task asincrono** che se ne occupa (la gestione delle immagini verrà discussa qui di seguito).

3.1 CARICAMENTO E GESTIONE DELLE IMMAGINI

Per consentire all'utente di inserire un qualsiasi numero di immagini all'interno del testo della nota il programma utilizza uno **SpannableStringBuilder** per disegnare delle bitmap ovunque egli voglia.
In particolare, quando l'utente desidera inserire una nuova immagine nella nota, il programma inserisce all'interno del testo un *image placeholder* che poi verrà sostituito visivamente con l'immagine desiderata.
Alla pressione del pulsante della toolbar per inserire una nuova immagine, se l'utente ha fornito il permesso di leggere la memoria, gli viene data la possibilità di scegliere la foto dalla galleria personale attraverso il picker di sistema.
A scelta effettuata, il task asincrono di caricamento dell'immagine effettua dei **controlli sulla risoluzione e qualità**. Per assicurarsi di risparmiare memoria di archiviazione e di poter caricare le immagini nella nota utilizzando il cursor del database in ogni dispositivo, il task effettua dei tentativi di interrogazione del database. Se la dimensione massima della cursor window viene superata allora si procede con una riduzione di qualità della **compressione della bitmap**. Dopo aver selezionato la qualità giusta, la bitmap viene *compressa in uno stream in formato Jpeg* per poi essere convertita in un byte array e memorizzata in nuovo record della tabella delle immagini del database (maggiori dettagli sul database forniti in seguito).
La cancellazione dell'immagine da parte dell'utente comporta la rimozione del placeholder e la conseguente eliminazione della bitmap dal database. Il task *cleanDatabaseFromDeletedImages* si occupa infatti di ripulire la tabella delle immagini del database da quelle eliminate dall'utente oppure da quelle inserite in note non salvate prima di uscire.
Quando l'utente aprirà la nota con immagini in un secondo momento, il task *loadNoteDetail* si occuperà di individuare tutti i placeholder delle immagini e di reperire le bitmap da mostrare, utilizzando gli id delle foto contenuti nei placeholder stessi, dal database.
Questo approccio alla gestione delle immagini consente di supportare il caricamento di quasi qualsiasi immagine in ogni dispositivo, risparmiando memoria di archiviazione e velocizzando l'esecuzione del programma.

3.2 INSERIMENTO DEI DISEGNI

Quando l'utente vuole **inserire un disegno** fatto a mano nella nota il programma lancia l'activity apposita, di cui ci occuperemo tra poco, dalla quale riceve, al termine, l'id del disegno memorizzato all'interno della tabella delle immagini del database. La memorizzazione, la cancellazione e la visualizzazione dei disegni infatti vengono gestite esattamente come per le immagini essendo le bitmap disegnate dall'utente compresse in formato Jpeg e convertite in byte arrays.

3.3 CONDIVISIONE DELLE NOTE

Un'ulteriore funzionalità messa a disposizione dall'activity in questione è quella di condividere le note. Il programma lancia un intent di tipo ACTION_SEND, o ACTION_SEND_MULTIPLE se la nota contiene immagini, in modo tale che all'utente sia chiesto dal sistema con quale app vuole procedere alla condivisione. Nel caso di condivisione di una nota che contiene immagini (e/o disegni), dopo aver preparato il testo della nota (ripulito dai placeholder di cui abbiamo parlato) il programma prepara le immagini da rendere disponibili alle altre applicazioni selezionabili per la condivisione. Per fare questo sfrutta il **file provider** definito nel manifest alla scopo di consentire

alle altre applicazioni l’accesso alle uri dei nuovi file creati per contenere le bitmap da condividere e e memorizzati nelle memoria cache della app.

4. CREAZIONE DEI DISEGNI

La ***DrawingActivity*** è l’activity che viene lanciata quando l’utente desidera creare un disegno da inserire nella nota. Il suo layout contiene la custom view che ho creato e che costituisce una semplice area di disegno. Quest’ultima non fa altro che disegnare sullo schermo i vari paths che l’utente descrive sulla view con le proprie dita. La DrawingActivity consente di modificare la dimensione e il colore del tratto da disegnare, utilizzando un range slider e importando un semplice color picker da libreria esterna.

5. DATABASE

L’applicazione utilizza un database implementato dalla classe java ***NotesDataBase***, la quale estende SQLiteOpenHelper. Il database contiene due tabelle, una per le note e una per le immagini (e i disegni che sono salvati come immagini). Sia le note che le immagini sono identificate da un id intero non nullo che viene utilizzato per recuperarle ed eventualmente modificarle o eliminarle.
Nella classe sono presenti tutti i metodi che il programma sfrutta per manipolare il database.