Pianificatore

Relazione del progetto del corso di
Laboratorio di applicazioni mobili (a.a. 2018/2019)

Scopo dell'applicazione

L'applicazione *Pianificatore* ha come scopo la pianificazione e gestione delle attività personali giornaliere.
Non sono promemoria, ma compiti da svolgere; le attività si concludono all'interno della giornata stessa, non sono prolungate nel tempo.

Funzionalità previste

Le funzionalità previste sono: la creazione, la modifica e la rimozione di un'attività da svolgere; visualizzare i dettagli di un'attività; visualizzare con filtri le attività da svolgere; ricezione delle notifiche (a cui eventualmente possono essere applicati dei filtri) che ricordano di svolgere un'attività e permettono di segnare l'attività come "in corso di svolgimento" oppure di posticiparla; visualizzare i grafici di utilizzo dell'applicazione; visualizzare la cronologia delle attività completate.

Caratteristiche e requisiti

L'applicazione gira sui sistemi Android; il minimo livello API richiesto è 21.

L'applicazione richiede l'autorizzazione all'*Esecuzione all'avvio* del dispositivo, necessaria per il corretto funzionamento delle notifiche dell'app.

Progettazione e scelte implementative

Classe `MainActivity.java`:
Fulcro dell'applicazione, gira tutto intorno ad essa.
Di base mostra la lista di attività in programma.

Mostra solo una parte delle attività in programma se vengono selezionati dei filtri per la visualizzazione.
Mostra le attività completate se viene cliccata la voce *Cronologia* nel menu del `NavigationDrawer`.
Associata con `activity_main.xml`, la quale è gestita con un `RecyclerView` riempito con `TextView`.

Classe `DataSet.java`:
Gestisce un insieme di dati.
È una classe astratta.

Classe `Task.java`:
Gestisce una singola attività.

Classe `TaskSet.java`:
Gestisce un insieme di attività (un insieme di `Task`).
È una sottoclasse di `Dataset`.

Classe `Id.java`:
Permette di ottenere un numero identificativo univoco da associare ad un `Task`.

Classe `Vis.java`:
Gestisce un singolo elemento (una riga) che compone la visualizzazione della `activity_main`.

Classe `VisualizeSet.java`:
Gestisce l'insieme di elementi visualizzati nella `activity_main`.
È una sottoclasse di `Dataset`.

Classe `FormActivity.java`:
Mostra un form per l'inserimento dei dati necessari alla creazione di una nuova attività (cioè un nuovo `Task`).
Associata con `activity_form.xml`, la quale è gestita con un `LinearLayout`.

Classe `DetailTaskActivity.java`:
Mostra le informazioni dettagliate di una singola attività.
Permette di modificare, eliminare e segnare come completata l'attività in questione.
Associata con `activity_detail_task.xml`, la quale è gestita con un `LinearLayout`.

Classe `GraphicsActivity.java`:
Mostra i grafici relativi all'utilizzo dell'applicazione da parte dell'utente: un istogramma (`BarChartFragment.java`) che indica il numero di attività in programma nei mesi dell'anno corrente e un grafico a torta (`PieChartFragment.java`) che mostra la percentuale delle attività in programma divise nelle varie classi d'appartenenza (*Famiglia*, *Lavoro*, *Tempo libero* e *Altro*).
Associata con `activity_graphics.xml`.

Classi `NotificationAlarmReceiver.java`, `NotificationIntentService.java`, `DeviceBootReceiver.java`:
Gestiscono la programmazione delle notifiche per ricordare lo svolgimento di un'attività. Per ogni attività viene programmata una notifica che comparirà nell'orario di svolgimento della stessa (indicato in fase di creazione) e che contiene un bottone *In corso* che permette di segnare l'attività come "in corso di svolgimento" e un bottone *Posticipa* che permette, tramite la visualizzazione della `activity_form`, di impostare un nuovo orario di svolgimento per quella attività.

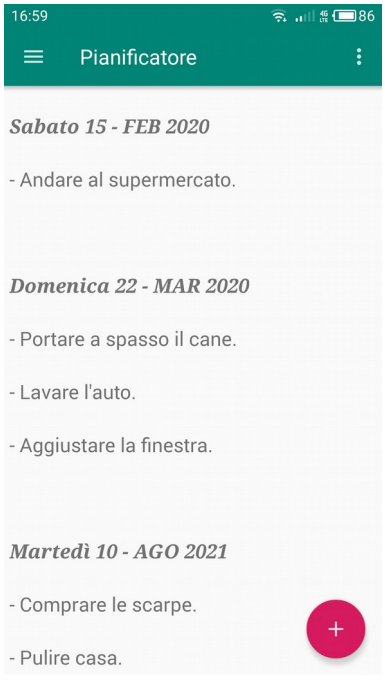
Utilizzata la libreria `Realm` per la persistenza dei dati sul dispositivo.
Ad ogni creazione, modifica (dell'attività stessa o del suo stato) o rimozione di una attività da parte dell'utente avviene il salvataggio dei nuovi dati.
Vengono salvati in memoria: le attività ancora da svolgere e quelle già svolte (i singoli `Task` presenti nei `TaskSet`), l'ultimo valore assegnato come id ad un `Task`.

Difficoltà e soluzioni

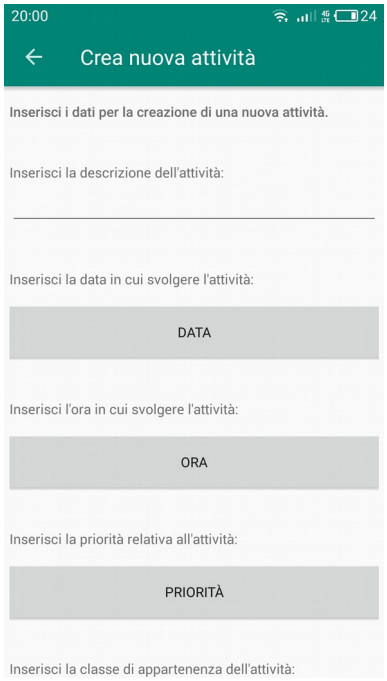
Uno dei principali elementi di riflessione è stata la scelta di come impostare la visualizzazione delle attività nella `activity_main`.
La soluzione adottata è stata quella di gestire ogni riga come una stringa di testo, così da ottenere un maggior controllo sul singolo elemento. Le righe da mostrare sono contenute in maniera ordinata in un `VisualizeSet`.

Casi d'uso ed esempi di funzionamento tipici

a.



b.

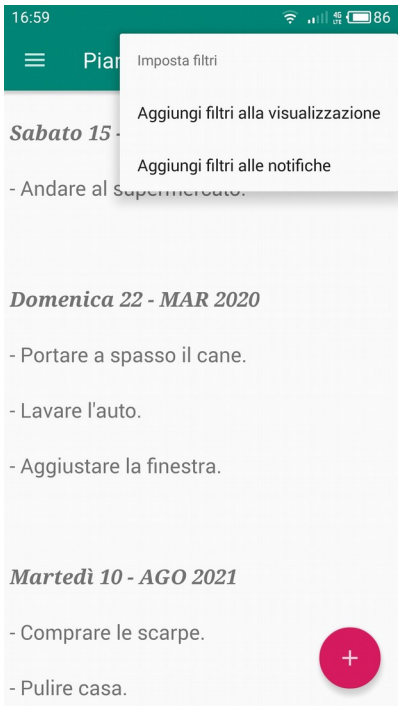


c.

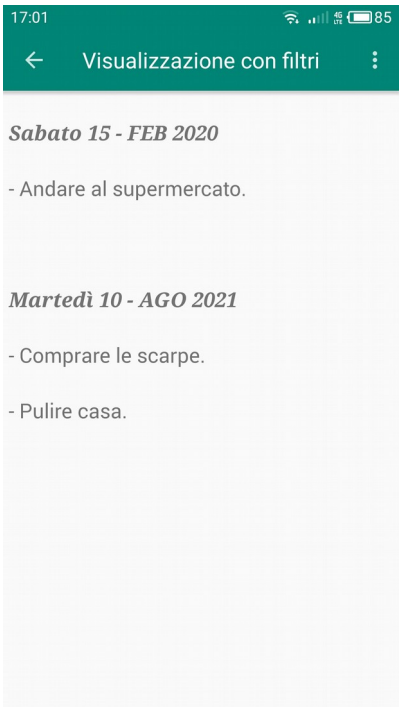
a. Lista delle attività in programma.

b. Creazione di una nuova attività.
c. Dettagli di una attività.

d.



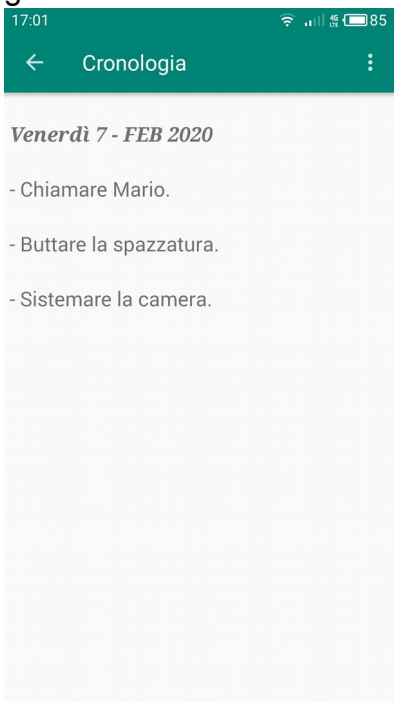
e.



d. Menu per l'aggiunta dei filtri (alla visualizzazione delle attività in programma o alla ricezione delle notifiche).
e. Visualizzazione con filtri delle attività in programma.

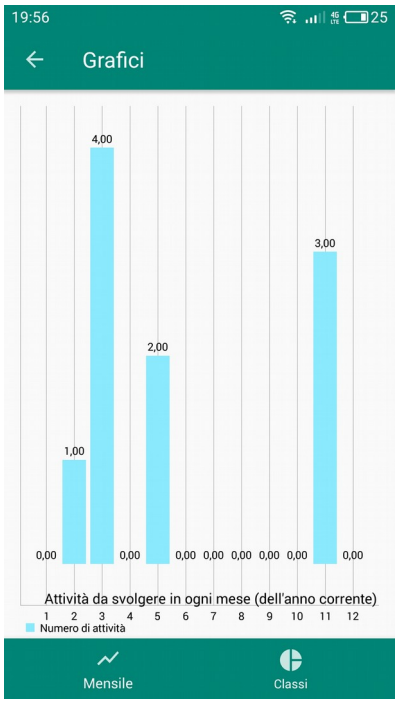
f.

g.

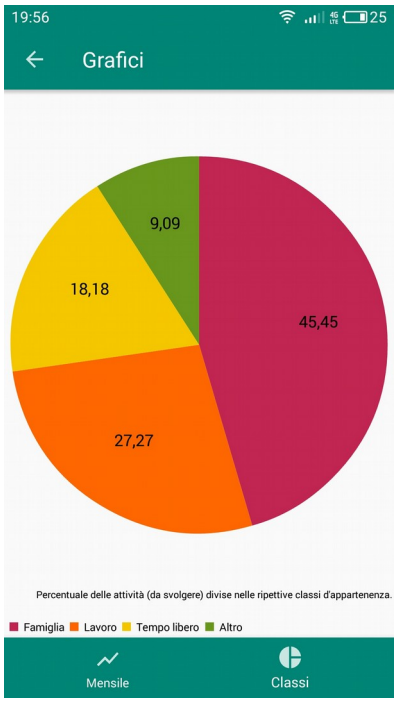


f. Menu a comparsa (sulla pagina principale).
g. Cronologia delle attività completate.

h.

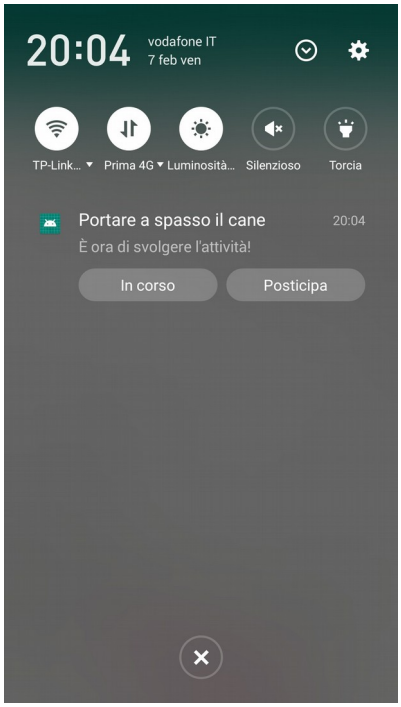


i.



h. Istogramma che rappresenta il numero di attività da svolgere nei mesi dell'anno corrente.
i. Grafico a torta che indica in percentuale la quantità di attività divise nelle rispettive classi di appartenenza.

l.



l. Ricezione della notifica di una attività.

Idee per estensioni

Possibilità di mostrare la lista di attività da svolgere come widget nella home del dispositivo.

Possibilità di ricordare lo svolgimento di una attività con una suoneria prolungata (piuttosto che una semplice notifica, che potrebbe passare inosservata).

Sincronizzare l'applicazione con il calendario di sistema del dispositivo, così che l'utente possa visualizzare le attività in programma insieme ai personali eventi del calendario.

Conclusioni e commenti finali

Utile soprattutto per la sua semplicità e facilità di utilizzo.

A differenza di un classico calendario con eventi questa applicazione corrisponde più a una lista di cose da fare (cioè attività da svolgere) e alla loro gestione e pianificazione.

Daniele Polidori

daniele.polidori2@studio.unibo.it

0000803992