DataVirus.it

Federico Rachelli

9 giugno 2020

L'app **DataVirus.it** é una implementazione Android del sito web omonimo. Fornisce all'utente una visualizzazione grafica dei dati del Dipartimento di Protezione Civile sull'andamento dell'epidemia di COVID-19.
I dati sono accedibili dal repository GitHub ufficiale della Protezione Civile. Tali dati vengono aggiornati a cadenza giornaliera (a partire dalle ore 18:00) fino alla fine dello stato di emergenza dichiarato dal Consiglio dei Ministri in data 31 Gennaio 2020.
Per riferimenti precisi sul codice si prega di controllare il Javadoc fornito con la documentazione ed il codice dell'app.

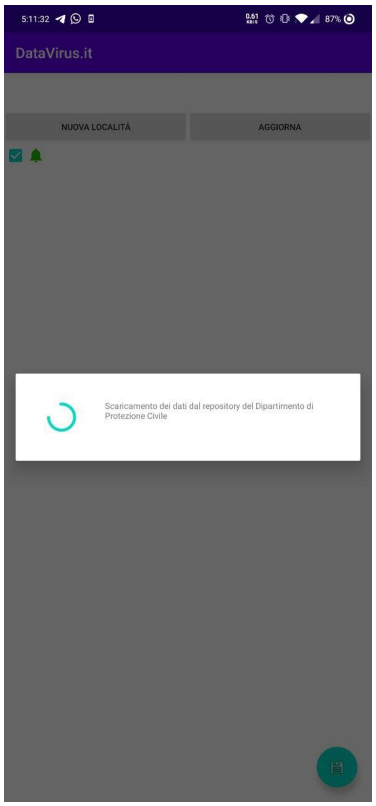
1 Visualizzazione dati

L'applicazione all'avvio visualizza una schermata di caricamento dei dati dal Dipartimento di Protezione Civile. Tali dati, una volta ottenuti, sono elaborati e vengono mostrati all'utente gli andamenti a livello nazionale dell'epidemia (Figura 1a).

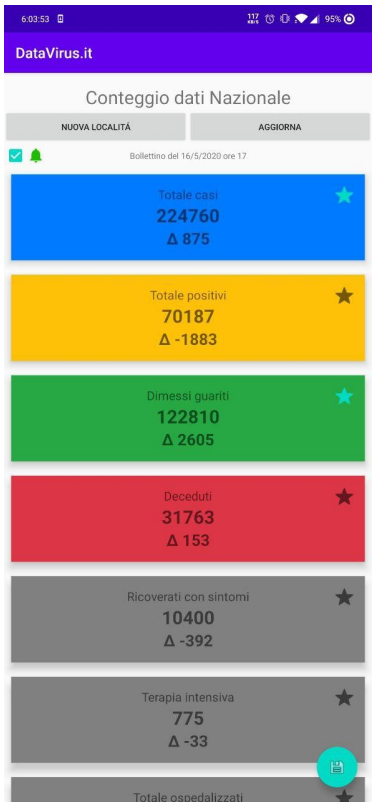
Come si puó notare nella figura 1b, questa é la schermata principale dell'applicazione. Viene mostrato in alto la denominazione territoriale dei dati (nella scermata principale verrá mostrato l'andamento nazionale). Sotto i pulsanti, viene visualizzata la data dell'ultimo aggiornamento disponibile dal Dipartimento di Protezione Civile.
Il pulsante "Aggiorna" ricarica i dati dal repository.
Vengono poi visualizzate le tile contenenti il dato odierno e la relativa variazione (delta) rispetto alla giornata precedente.
Nota: il Dipartimento di Protezione Civile (**DPC**) (ad oggi) fornisce solamente il dato riguardante il totale dei contagiati per tutte le province del territorio italiano.

1.1 Implementazione

La MainActivity, alla sua creazione, istanzia un oggetto di tipo DataParser. La classe DataParser si occupa di eseguire il download dei file JSON dalla repository del DPC in modalitá grafica (verrá visualizzato il dialog che segnala il download in esecuzione. Il dialog é un'istanza di *LoadingDialog*, la quale é un'estensione di *DialogFragment*).
Una volta concluso lo scaricamento, i dati verranno memorizzati in un oggetto di tipo *DPCData*, accedibile dalla callback *onDPCDataReady()* nell'interfaccia *OnDPCDataReady*, o alternativamente dal metodo statico *DataParser.getDPCDataInstance()*.



(a) Caricamento dei dati



(b) Andamento nazionale

La MainActivity contiene un'istanza di *DataTilesFragment*, che si occupa di visualizzare le tile con i dati (ed incrementi), colorandole e posizionandole in ordine a seconda del parametro (e.g. la tile *Dimessi guariti* viene sempre colorata di verde e posizionata in terza posizione, la tile *Totale casi* viene sempre colorata di blu e posizionata in prima posizione). Non per tutte le tile é previsto un ordinamento oppure un colore (verrá applicato il grigio come colore di default). Per la lista delle tile scrollabili é stata prevista una RecyclerView (integrata in *DataTilesFragment*). Le tile sono realizzate con un layout CardView. Al click su una tile seguirá una chiamata al metodo *onTileClick* presente in una classe che implementa l'interfaccia *OnTileClick*. Nella fattispecie tale classe sará una istanza di *ChartActivity*.
Si é optato per la realizzazione del fragment *DataTilesFragment* per essere riutilizzato nella visualizzazione dei preferiti (si veda la sezione Preferiti).

2 Cambio zona geografica

La denominazione geografica puó essere cambiata premendo sul pulsante “Nuova localitá” dalla schermata principale (vedi 1b).

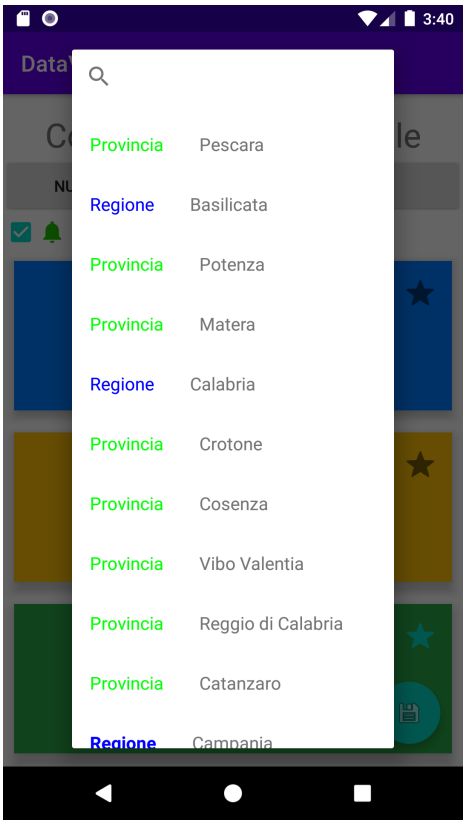


Figura 2: Picker zona geografica

Grazie al picker (mostrato in figura 2) si puó scegliere la zona geografica di interesse. Puó essere una provincia, una regione oppure si puó selezionare l’andamento nazionale. Per comoditá si puó ulteriormente cercare la zona d’interesse premendo sulla *lente d’ingrandimento*. Come risultato verranno visualizzati i dati relativi alla zona selezionata.

Premendo il pulsante back *fisico* si scorreranno all’indietro tutte le selezioni geografiche finora cercate dall’utente. Una volta ritornati alla prima visualizzazione (corrispondente sempre al dato nazionale) l’app si chiuderá.

Figura 3: Esempio visualizzazione regione Lombardia

2.1 Implementazione

La classe *DPCGeoAdapter* é un'estensione di un DialogFragment che, quando aperto, mostrerá una RecyclerView contenente la lista delle regioni e delle province.
In prima posizione si trova la selezione per il dato nazionale.
La lista é strutturata in modo che, per ogni regione, ci sia la lista delle province che la compongono nelle posizioni sottostanti.
Alla pressione di una posizione geografica, il GeoPicker chiamerá la funzione *onDPCGeoClick* di una classe che implementa *OnDPCGeoListener* (nella fattispecie verrá utilizzata la MainActivity). L'oggetto passato é di tipo *GeographicElement*, il quale descrive il nome dell'ambito geografico e la sua tipologia (*NAZIONALE, REGIONALE, PROVINCIALE*).

3 Preferiti

Ogni tile ha in alto a destra una *stellina*: quando premuta, la tile relativa viene salvata tra i preferiti. Per recuperare la lista delle tile salvate é sufficiente premere sul floating button in basso a destra (visibile nella schermata principale dell'app, 1b).

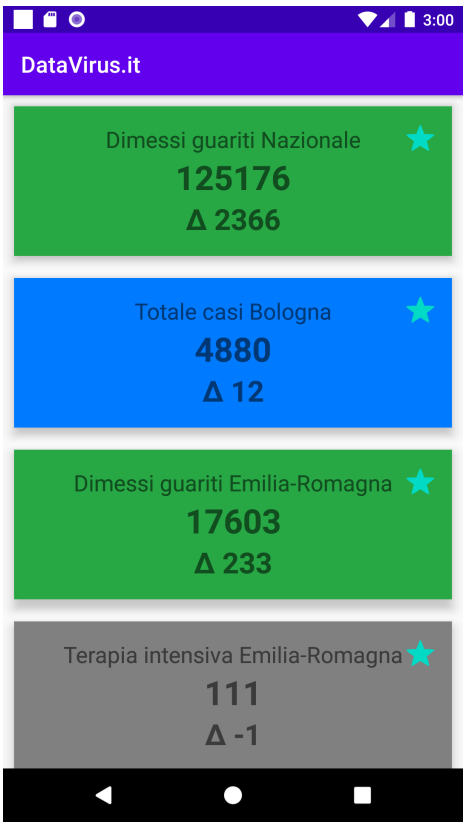


Figura 4: Lista dei preferiti

Come si puó notare dalla figura 4, le tile salvate sono visualizzate con la loro denominazione geografica a seguito. Premendo su una *stellina* precedentemente marcata verrá rimossa tale tile dalla lista dei preferiti.

3.1 Implementazione

Per il salvataggio dei preferiti si fa uso della classe *ManageStarredTiles*, che offre i metodi per aggiungere o rimuovere un elemento dai preferiti. Per identificare una tile da aggiungere o rimuovere viene passato un oggetto di classe *FieldGeographicElement* (la quale é una estensione della classe *GeographicElement*, con l'ulteriore informazione del parametro del dato di interesse, e.g. "Totale casi").

I dati sulle tile marcate come starred vengono salvati in un file privato dell'app, in formato JSON. Si é preferito utilizzare il salvataggio dei dati in un file in quanto, in una implementazione futura, il sito web DataVirus.it produrrá tale file per i dati preferiti nel formato utilizzato attualmente nell'app.
Per la memorizzazione della lista delle tile preferite viene utilizzata la *StarredActivity*, che si occupa di istanziare un *DataTilesFragment* che visualizzerá tutte le tile salvate.
Per la visualizzazione della lista delle tile preferite si riutilizza il fragment *DataTilesFragment*.

4 Grafici

Ogni qualvolta si premerá su una tile, verrá mostrato l'andamento nel tempo del dato selezionato in un grafico. Si presenterá la schermata come quella in figura 5a.
Il dato che ora viene mostrato puó essere confrontato con altri dati. Per ottenere questo risultato sará sufficiente premere sul pulsante "+" e selezionare una nuova tile.
Ad ogni dato aggiunto viene assegnato un colore random per la sua rappresentazione e apparirá nella lista dei campi attualmente disegnati. Per ciascuno di questi si potrá decidere se nasconderlo o mostrarlo (con la checkbox sulla sinistra) oppure cancellare dalla lista premendo sull'icona del *cestino*.
Il grafico é ingrandibile a piacimento, semplicemente utilizzando il *pinch-to-zoom*.

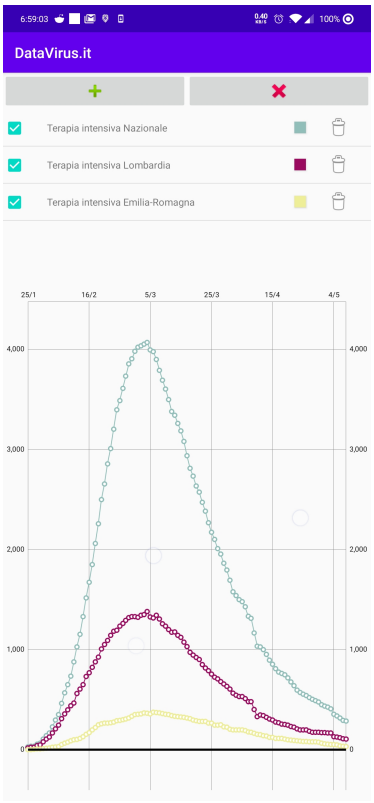
Alla pressione del pulsante "-" si chiuderá la schermata del grafico e il grafico attualmente disegnato verrá scartato, insieme a tutti i dati precedentemente immessi. Il medesimo risultato si puó raggiungere alla pressione del pulsante back *fisico*;

Per una maggior comoditá di interazione, é disponibile anche la visualizzazione del grafico con il dispositivo orientato in *landscape* come mostrato in figura 6

4.1 Implementazione

Per il grafico é stata utilizzata la libreria ***MPAndroidChart***, che fornisce un'interfaccia completa per il plotting fornendo molteplici funzionalitá. Esistono svariate librerie per disegnare grafici: ho optato per MPAndroidChart perché offre un buon dettaglio ed una semplicitá di implementazione ed utilizzo notevole, senza dover ricorrere a tool a pagamento.
Questa viene utilizzata dentro alla *ChartActivity*. I dati sono ottenuti attraverso il singleton *ChartModel*, la quale istanza gestisce i dati da disegnare sul grafico. I dati contenuti in *ChartModel* possono essere modificati dall'utente tramite il fragment *ChartElementsList*, che gestisce la RecyclerView contenente tutti gli elementi aggiunti al grafico e le funzionalitá per nascondere il dato oppure rimuoverlo. Il pulsante "+" termina l'activity corrente senza pulire la lista dei

(a) Esempio grafico Milano



(b) Confronto tra terapie intensive

dati nel grafico, al contrario del pulsante "-" che elimina tutti gli elementi finora aggiunti (risultato medesimo quando si preme il pulsante *back* fisico).
Quando si preme su una tile si genera l'*intent* per eseguire *ChartActivity*, passando come extra valori gli elementi contenuti in una istanza di *GeographicFieldElement* (estensione di *GeographicElement*).

5 Notifiche periodiche

Nell'activity principale, a sinistra della data di ultimo aggiornamento, é presente una checkbox con a fianco una campanellina (vedere fig. 1b).
Quando marcata, a partire dalle ore 18:00 di ogni giorno, ogni 10 minuti verrá performata una ricerca di nuovi dati aggiornati (orario nel quale il DPC li rende pubblici). Nel caso in cui i dati scaricati abbiano data odierna, l'utente verrá avvisato attraverso una notifica. Alla pressione di tale notifica si aprirá l'app. In figura 7 un esempio della notifica che verrá mostrata

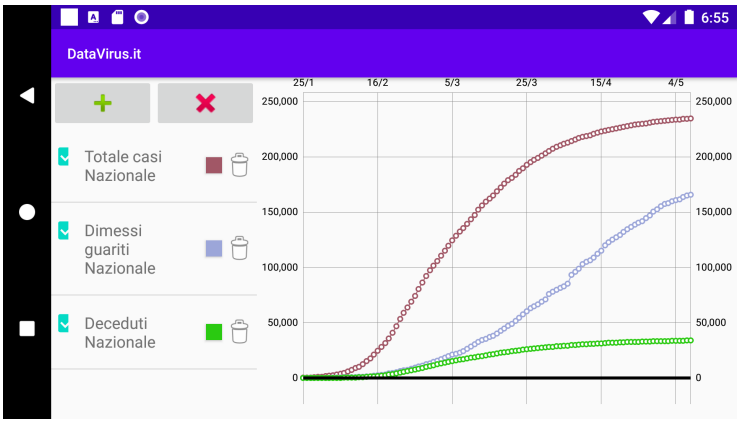


Figura 6: Visualizzazione landscape della *ChartActivity*

5.1 Implementazione

Per gestire le notifiche si é fatto uso di una istanza *DataNotifyReceiver*, classe di tipo "Broadcast Receiver" che alla sua chiamata istanzia un oggetto di tipo *DataParser* (che puó essere richiamato anche senza interfaccia grafica, quindi non mostrando il popup del caricamento) per ottenere dati aggiornati.
La chiamata a *DataNotifyReceiver* viene effettuata attraverso un'*intent* performato dall'*AlarmManager*, per impostare l'orario in cui cominciare la ricerca (ore 18:00). La definizione della chiamata all'*AlamrManager* viene eseguita dalla *MainActivity* attraverso i metodi *setAlarm()* ed *unsetAlarm()*.
La *MainActivity.setAlarm()* chiama l'*AlarmManager::setInexactRepeating()* passando un intent di esecuzione del BroadcastReceiver *DataNotifyReceiver*, "puntando" l'orario (inesatto, per motivi di risparmio energetico) delle 6.00PM del giorno successivo (se le ore 18:00 sono giá passate, altrimenti del giorno stesso).
La *MainActivity.unsetAlarm()*, come intuibile, disabilita il timer (se questo non é impostato, il comando non produrrá alcun risultato).
Nel caso in cui i dati ottenuti dalla repository GitHub siano aggiornati alla data odierna, *DataNotifyReceiver* invia una notifica all'utente (*DataNotifyReceiver.showNotification()*). Altrimenti viene impostata l'esecuzione del *DataNotifyReceiver* differita di 10 minuti per una nuova ricerca (con una chiamata al metodo *DataNotifyReceiver.setNextIntent()*).

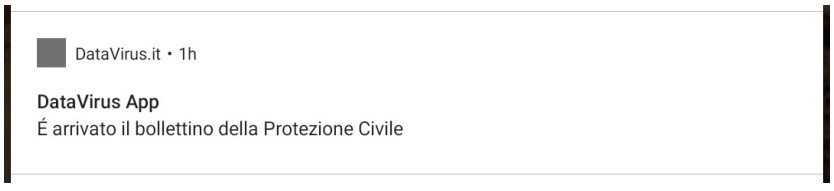


Figura 7: Esempio di notifica