RELAZIONE MOBILE PROGRAMMING

RecyclerView

Le RecyclerView sono uno strumento molto utile quando c’è bisogno di organizzare una lista di informazioni secondo uno schema preciso. Nell’applicazione, ogni Model di tipo Hero, Comics, Series, Events o Creators ha altre liste di Model a lui collegate (es i fumetti, le serie e gli eventi collegati ad un eroe) che vanno quindi mostrati all’utente.

Le liste sono di dimensione variabile, andando da 20/30 elementi fino anche a più di 50. La RecyclerView manitene tutti gli elementi che non possono essere mostrati per questioni di spazio nel display in una apposita cache, così da andare a caricare i nuovi quando viene eseguito uno scroll per visualizzare gli altri elementi, o ricaricare i vecchi quando si vuole tornare verso l’inizio della lista. La gestione di liste con dimensioni considerevoli è quindi totalmente delegata alla RecyclerView.

Le RecyclerView sono state introdotte pressoché in ogni Activity, specialmente in quelle che mostravano il dettaglio di uno specifico Model, quindi la HeroDetailActivity, ComicsActivity, SeriesActivity, EventsActivity e CreatorsActivity.

Ogni RecyclerView è un contenitore di singole View, che possono avere uno schema più o meno complesso, in questo caso le View hanno una struttura molto semplice, in quanto contengono View gestite da un LinearLayoutManager con opzione di scorrimento orizzontale.

Adapter

Ogni elemento della RecyclerView è rappresentato mediante una View indipendente, con un preciso layout che deve essere gestito mediante appositi ViewHolder.

Ogni ViewHolder è a sua volta gestito da un Adapter, che svolge la funzione di ponte fra la UI e i ViewHolder.

Abbiamo scelto di avere una apposita classe Holder per ogni Adapter, che estende la classe Recycler.ViewHolder e gestisce quindi gli elementi del layout. In questo modo, il codice è molto più manutenibile e leggibile.

L’Adapter si occupa di collegare i dati agli Holder mediante il metodo onBindViewHolder(), andando quindi ad assegnare ad ogni Holder una specifica posizione nella lista.

Ognuno degli elementi nella RecyclerView è inoltre clicabile, infatti ogni Adapter implementa il metodo di onClick(), andando a caricare la schermata di dettaglio per l’elemento selezionato dall’utente.

Menu contestuale (lo farà Adrian)

Per permettere all’utente di eliminare dalla lista dei preferiti un elemento, senza dover per forza aprire l’activity di dettaglio, è stato implementato un piccolo menù contestuale con la singola voce “elimina”, gestito dal FavouriteFragment.

Per far attivare il menu contestuale è stato usato il meccanismo dell’onLongClick(), che è stato implementato negli Holder gestiti dal FragmentAdapter. Ogni volta che viene effettuato un long press su uno dei preferiti, viene mostrato il menù contestuale e a questo punto l’utente può selezionare / deselezionare altri elementi con un semplice press ed eliminarli tramite la voce del menù, che si chiude subito dopo l’eliminazione.

Caching con Shared Preferences

Per poter mantenere sempre identici i suggeriti del giorno, è stato neccessario aggiungere un meccanismo di caching, per fare questo la scelta è ricaduta sulle Shared Preferences. Dovendo salvare 3 Model , un eroe, un fumetto ed una serie, le SharedPreferences si prestano molto bene in quanto permettono il salvataggio basato su una coppia chiave-valore in un file. Per poter salvare tutte le informazioni legate ai model, la scelta è stata quella di convertire i singoli model in un Json tramite l’API Gson, per poi riconvertirli ogni volta che viene mostrata l’ Home Page.

Il metodo per ottenere il file su cui salvare i dati è getPreferences(), in quanto è necessario un solo file per contenere tutto, tale file viene resettato ogni giorno con i nuovi suggeriti del giorno.