Relazione progetto “Runner2”

Cristiano Calicchia
Marco Swid

Strategia di realizzazione

Il progetto assegnatoci per il corso di Mobile Programming riguarda lo sviluppo dell’ applicazione denominata Runner2. Durante tutte le fasi del progetto è risultato necessario utilizzare un database per mantenere i dati relativi al funzionamento dello stesso, presenti in file json. Per la creazione di suddetto database abbiamo deciso di utilizzare un server gratuito presente online, il quale sfrutta la nota piattaforma “Github” per mantenere i file. Grazie ad un’estensione del sito chiamata “typicode”, è possibile trasformare una delle repository del proprio profilo in un server fittizio in grado di fornire servizi REST. Per fare ciò, una volta inizializzata la repository, è necessario creare un file chiamato “db.json”, in cui inserire i diversi oggetti json da memorizzare. E’ possibile dividere il database in diversi gruppi, ciascuno con un proprio nome, in modo da poter salvare dati eterogenei, mantenendoli divisi. I dati devono essere contenuti tra parentesi graffe. Per creare un gruppo si usa la seguente struttura:

“nome gruppo”:[
<oggetti json>
]

Infine, affinché l’estensione funzioni è necessario inserire un ultimo gruppo, come segue:

“profile”{
“name”: “typicode”
}

Per effettuare le chiamate REST abbiamo utilizzato la libreria Volley di android, la quale permette la comunicazione con indirizzi url. L’ url del nostro database è https://my-json-server.typicode.com/lufth/mp_Project , ma per accedere ai dati nei gruppi da noi creati è necessario specificare anche il nome del gruppo richiesto aggiungendo all’ url “ /<nome gruppo>. In base alle necessità dell’ applicazione, abbiamo diviso il server nel seguente modo:

1. Races
2. Users
3. Booking

Dopo aver importato le API di volley necessarie, per richiedere un file json con una chiamata di tipo GET sono necessarie le funzioni del seguente esempio:

```
RequestQueue mRequestQueueUL = Volley.newRequestQueue(this);
JsonArrayRequest requestUL=new JsonArrayRequest("https://my-json-server.typicode.com/lufth/MP_Project/users", getListener, errorListener);
mRequestQueueUL.add(requestUL);
```

In questo caso viene richiesto un array di oggetti json al nostro server nel gruppo users. Una volta inserita la richiesta nell’ ultima riga di codice, entrano in azione i due listener forniti.L’ errorListener fornisce il Toast “Errore di rete” nel caso in cui la GET non sia andata a buon fine, mentre il getListener gestisce l’ array restituito a seconda delle situazioni. Per le chiamate di tipo POST:

```
RequestQueue mRequestQueueUR = Volley.newRequestQueue(this);
JsonObjectRequest requestUR=new JsonObjectRequest("https://my-json-server.typicode.com/lufth/MP_Project/users", newUser, postListener, errorListener);
mRequestQueueUR.add(requestUR);
```

Il funzionamento è analogo alla chiamata GET, con la differenza che per la POST viene fornito il file da caricare nel server con il postListener a gestire la situazione in caso di successo. Durante lo svolgimento del progetto, abbiamo provato ad utilizzare diversi server gratuiti online per creare il nostro json database, ma tutti fornivano lo stesso errore nel momento in cui si provava una chiamata POST, restituendo all' utente "Errore di rete". Ciò dipende da come sono stati programmati questi server , ma è fuori dal nostro controllo ed esula dallo scopo del progetto, perciò non abbiamo risolto questa situazione. I postListener sono stati comunque programmati, nel caso in cui il server funzioni per cause a noi sconosciute. Le chiamate GET avvengono, invece, senza alcun problema. Per creare e gestire gli oggetti json all' interno dell' applicazione abbiamo utilizzato l' apposita libreria di android.

Activity

La prima activity che viene presentata all' utente è quella del login, denominata "LoginActivity" e la cui interfaccia grafica è specificata nel file "activity_login.xml". Sono presenti:

1. Un plain text per l' inserimento dello username
2. Un password field per l' inserimento oscurato della password
3. Due button per richiedere il login o il signup

I primi due fanno entrambi parte della categoria EditText e i dati inseriti dall' utente vengono presi alla pressione di uno dei due bottoni. Per fare ciò, si usa la funzione "findViewById" e il metodo "getText" proprio degli EditText.
Nel caso in cui venga premuto il bottone "Accedi", viene effettuata una chiamata GET al nostro server, prendendo tutti gli oggetti json presenti in "Users" in caso di successo o fornendo l' errore di rete in caso di fallimento. Successivamente, vengono scorsi confrontando l' username fornito dall' utente con quelli presenti. Nel caso in cui ci sia un riscontro, viene confrontata la password fornita con quella associata: in caso di fallimento, si fornisce all' utente un Toast in cui viene comunicato che la password è errata; in caso di successo, viene inizializzata l' activity successiva, chiamata "MainActivity", con un intent, a cui viene affidato anche lo username dell' utente tramite la funzione "putExtra", che consentirà di accedere a questo dato anche nelle altre activity, grazie alla funzione "getIntent().getStringExtra(<nome dell' activity da cui prendo il dato>.<nome del dato>)", in quanto il datò servirà in un' activity successiva. Se lo username non viene trovato, allora all' utente verrà inviato un Toast che comunica che il nome utente inserito non è stato trovato. Quest' ultimo viene mostrato all' utente solo quando lo username non è stato trovato; in caso in cui venga trovato, un flag viene impostato a true, anche se la password è errata e quindi il Toast non viene mostrato. Senza il flag, quest' ultimo sarebbe stato mostrato ad ogni confronto tra username.
Alla pressione del bottone "Registrati", vengono presi i dati nei due EditText, vengono inseriti in un oggetto json e viene fatta una chiamata di POST al gruppo Users nel nostro server, la quale non funziona per i motivi spiegati precedentemente, ma se funzionasse, una volta registrati, si potrebbe effettuare il login. Abbiamo deciso di far effettuare la registrazione solo con questi due semplici

dati, per non riempire il progetto di informazioni secondarie e non richieste nella consegna, lasciando comunque la registrazione per completezza.
Una volta effettuato l' accesso, l' utente visualizza la MainActivity, così chiamata poiché da questo punto si snodano le tre versioni del progetto Runner. Il layout corrispondente si trova nel file "activity_main.xml". Sono presenti tre bottoni, tra cui l' unico funzionante è "Prossime Gare", oggetto di Runner 2. Alla pressione del bottone, viene inizializzata l' activity "NextRacesActivity", in cui vengono mostrate le prossime gare presenti nel database. Innanzitutto viene effettuata una chiamata GET al gruppo "Races" del nostro server e, in caso di successo, gli oggetti json forniti vengono scorsi e ciascuno viene immagazzinato in una nuova istanza di una classe da noi creata, la classe "Race". Gli elementi di questa classe hanno la seguente struttura:

```
public Race(int id, String image, String name, String location, String date, long distance) {
    this.id=id;
    this.image=image;
    this.name=name;
    this.location=location;
    this.date=date;
    this.distance= distance;
}
```

Vengono salvati solo questi dati poiché nella lista non devono essere mostrati tutti quelli presenti nell' oggetto, ma solo quelli richiesti dalla traccia. Tutte le istanze vengono inserite in ordine in una lista precedentemente inizializzata per essere inseriti in una ListView, specificata nel file "activity_next_races.xml". Questa ListView, dopo essere stata inizializzata, viene riempita dalle View che seguono il modello presente in "race_list.xml" grazie all' adapter che viene chiamato sulla lista di istanze Race e sulla ListView stessa. Questo adapter, per ogni elemento della lista, prende i dati salvati e li posiziona negli elementi della singola View. Tutte le view vengono poi messe di seguito per costruire la lista desiderata, creata senza conoscere a priori la quantità di singole view da concatenare.
Tutti gli elementi della ListView sono cliccabili e al clic viene inizializzata l' attività "RaceDetailActivity" tramite un intent, a cui viene associato, tramite una "putExtra", l' id della corsa di cui si vogliono conoscere i dettagli. Per fare ciò, viene calcolata la posizione all' interno della lista precedentemente creata dell' istanza cercata. Questo viene fatto nelle seguenti righe di codice:

```
String index = ""+view.getTag();
int trueIndex = Integer.parseInt(index)-1;
Race mRace=mRaceList.get(trueIndex);
i.putExtra(ID_RACE,  String.valueOf(mRace.getId()));
```

Viene preso il tag della view che è stata cliccata, cioè la sua posizione all' interno della ListView salvata in una stringa. I tag partono da 1, mentre gli elementi di una lista da 0, quindi, dopo aver trasformato la stringa in un intero, si sottrae 1. In questo modo all' indice trueIndex è presente l' elemento Race cercato e al "putExtra" viene passato il suo id.
L' utente ora vedrà l' activity "RaceDetailActivity", il cui layout è specificato nel file "details.xml". Questa activity mostra dei dettagli sulle corse che precedentemente erano stati esclusi. Per fare ciò,

si deve come prima cosa trovare l' oggetto json corrispondente nel server nel gruppo "Users". Viene fatta una chiamata GET e vengono nuovamente salvati tutti gli oggetti presenti, in caso di successo, questi vengono scorsi, confrontando l' idrace dell' oggetto con l' id della corsa su cui l' utente ha cliccato, che viene recuperato grazie al metodo precedentemente descritto, nel nostro caso:

```
passedVarRace=getIntent().getStringExtra(NextRacesActivity.ID_RACE);
int idRace = Integer.parseInt(passedVarRace);
```

Una volta trovato l' oggetto corrispondente, la view viene riempita con i dati richiesti nella traccia e viene mostrata all' utente. E' inoltre presente un bottone che permette all' utente di iscriversi alla gara. Quando viene cliccato, viene recuperato lo username dell' utente, associato all' intent fatto durante l' attività di login, viene creato un oggetto json che contiene questo username e l' id della gara a cui l' utente si vuole iscrivere e, infine, viene fatta una chiamata di POST al nostro server nel gruppo "Booking". In caso di successo, l' utente vedrebbe un toast che conferma l' avvenuta registrazione, ma, come precedentemente spiegato, ciò non accade e verrà restituito un Toast che comunica l' errore di rete.