A.A. 2017/2018 – Progetto Finale Mobile Programming

Realizzato da: Polli Valeria, Andrea Lombardo

Runner2

Relazione Finale

Runner2 – Breve Panoramica

0. Server
1.Android Manifest
2.Login
3. Menu Principale
4.Lista delle gare disponibili
5. Scheda della gara e prenotazione

Runner2 – Server

Per effettuare le attività di tipo REST abbiamo
deciso di utilizzare un web server, nel nostro specifico ApacheHTTPServer.

Per utilizzarlo abbiamo istallato WampServer, una piattaforma di sviluppo di tipo WAMP che comprende tre server: Apache, MySQL e MariaDB.

Tutte le informazioni che vogliamo inserire nel server dovranno essere contenute nella cartella ‘’www”, ed è proprio qui che abbiamo inserito tre file : ‘’gare.txt” per la GET, ‘Informazioni.JSON” destinato ai risultati della post e infine server.php per abilitare il server alla post.

Runner2 – Server – come accedervi

- Attraverso la chiamata da terminale ipconfig possiamo leggere l'IPV4 del nostro server, da cui estrapolare l'indirizzo IP che usiamo al posto di localhost, nel caso di accesso da utente esterno alla macchina che fa da server.

- Nel caso invece di utilizzo attraverso stessa macchina basta localhost senza alcun tipo di porta.

- Dobbiamo modificare il file di configurazione:
  httpd-vhost.conf dove al posto di ''require local ..'' usiamo ''all granted'' per rendere accessibile il server a utenti esterni.

Runner2 – Android Manifest

Per abilitare l'interazione con il nostro server e il web, abbiamo bisogno di inserire i seguenti permessi:

```
<uses-permission android:name="android.permission.INTERNET" />
  <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Oltre a questi abbiamo bisogno di:

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
   <uses-permission android:name="android.permission.READ_PROFILE" />
   <uses-permission android:name="android.permission.READ_CONTACTS" />

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Per l'accesso ai contatti e l'abilitazione ai servizi di localizzazione.

Runner2 – Login –Login Activity

Per la nostra applicazione abbiamo deciso di partire da una Login Activity, che offre una base semplice ed efficiente.

La nostra scelta è stata quella di non utilizzare un database per memorizzare gli utenti, ma di inserire le credenziali di accesso direttamente nel codice.

```
private static final String[] OUR_CREDENTIALS = new String[]{
        "polli.valeria@gmail.com:hello", "andrea.lombardo@gmail.com:world"
  };
```

Runner2 – Login –Login Activity

Innanzitutto abbiamo bisogno dell'approvazione dell'utente ad abilitare i servizi prima citati.
Ciò è realizzabile attraverso due funzioni che permettono di visualizzare un messaggio interattivo.

Localizzazione

```
 private boolean mayRequestLocalization() {
      if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
         return true;
      }
      if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
         return true;
      }


      if (shouldShowRequestPermissionRationale(ACCESS_FINE_LOCATION)) {
         Snackbar.make(mEmailView, R.string.permission_localization,
Snackbar.LENGTH_INDEFINITE)
               .setAction(android.R.string.ok, new View.OnClickListener() {
                  @Override
                  @TargetApi(Build.VERSION_CODES.M)
                  public void onClick(View v) {
                     requestPermissions(new
String[]{ACCESS_FINE_LOCATION},REQUEST_ACCESS_LOCATION);
                  }
               });

      } else {
         requestPermissions(new String[]{ACCESS_FINE_LOCATION},
REQUEST_ACCESS_LOCATION);
      }
      return false;

   }
```

Accesso ai contatti

```
private boolean mayRequestContacts() {
      if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
         return true;
      }
      if (checkSelfPermission(READ_CONTACTS) == PackageManager.PERMISSION_GRANTED ) {
         return true;
      }


      if (shouldShowRequestPermissionRationale(READ_CONTACTS)) {
         Snackbar.make(mEmailView, R.string.permission_rationale,
Snackbar.LENGTH_INDEFINITE)
               .setAction(android.R.string.ok, new View.OnClickListener() {
                  @Override
                  @TargetApi(Build.VERSION_CODES.M)
                  public void onClick(View v) {
                     requestPermissions(new String[]{READ_CONTACTS},
REQUEST_READ_CONTACTS);
                     requestPermissions(new
String[]{ACCESS_FINE_LOCATION},REQUEST_ACCESS_LOCATION);
                  }
               });

      } else {
         requestPermissions(new String[]{READ_CONTACTS}, REQUEST_READ_CONTACTS);
      }
      return false;

   }
```

Runner2 – Login –Login Activity

Elemento fondamentale di questa activity è la classe *UserLoginTask* che estende AsyncTask e che permette di effettuare operazioni in background, per poi mostrarne i risultati sull'interfaccia del thread principale.

```
protected Boolean doInBackground(Void...
params) {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            return false;
        }

        for (String credential :
OUR_CREDENTIALS) {
            String[] pieces = credential.split(":");
            if (pieces[0].equals(mEmail)) {
return pieces[1].equals(mPassword);
            }
        }
```

Funzione che in background controlla se l'account esiste e se la password corrisponde alla mail fornita.

```
 @Override
      protected void onPostExecute(final Boolean success) {
         mAuthTask = null;
         showProgress(false);

         if (success) {
            finish();
            Intent myIntent = new
Intent(LoginActivity.this,MenuActivity.class);
 String nome = mEmailView.getText().toString();
            String pass = mPasswordView.getText().toString();
            myIntent.putExtra("email",nome);
            myIntent.putExtra("password",pass);

            LoginActivity.this.startActivity(myIntent);

         } else {

mPasswordView.setError(getString(R.string.error_incorrect_password))
;
            mPasswordView.requestFocus();
         }
      }
```

Se il login ha successo viene inizializzata una nuova activity corrispondente al menu principale della nostra app alla quale passiamo delle informazioni.

Runner2 – MenuActivity

Collegata a menu_layout,
permette di visualizzare
una lista con:

- Elenco Soci
- Prossime Gare
- Risultati

Come richiesto da traccia il nostro compito è stato quello di implementare questa parte.

Per ragioni di efficienza abbiamo deciso di effettuare all'interno di questa activity il prelievo del JSON, per poi passare il risultato a quella successiva.

Runner2 – MenuActivity - GET

Abbiamo creato una classe chiamata *MyFetcherJsonFunction* che estende AsyncTask e che presenta come metodi:

- onPreExecute()
- doInBackGround(String…params)
- onPostExecute(String result)
- Start(String text, String text1)

Runner2 – Menu Principale - MenuActivity - GET

- onPreExecute()

  ha il compito di mostrare un progress dialog con ‘’please wait”

- doInBackGround(String…params)

- metodo più importante, instaura una connessione HTTP con il server usando un URL di connessione che costruisce al suo interno servendosi di due stringhe.

```
String url0 = params[0] + java.net.URLEncoder.encode(params[1], "UTF-8");
```

http://indirizzo_ipv4/

Serve a codificare la stringa params[1]
secondo la codifica UTF-8.
Nel nostro caso params[1] è ‘’gare.txt”

- InputStream stream = connection.getInputStream();
  reader = new BufferedReader(new InputStreamReader(stream));

```
StringBuffer buffer = new StringBuffer();
String line = "";

while ((line = reader.readLine()) != null) {
        buffer.append(line+"\n");
        Log.d("Response: ", "> " + line);
    }
return buffer.toString();
```

Con lo stream proveniente dalla connessione si costruisce un nuovo BufferedReader, da cui viene prelevato il contenuto, rielaborato e inserito in uno StringBuffer.
La funzione ritornerà una stringa ottenuta tramite l'operazione buffer.toString()

❑ onPostExecute(String result)

Prende in input la stringa ritornata in precedenza e imposta la stringa txtJson(attributo di MainActivity) con result.

```
if (pd.isShowing()){
        pd.dismiss();
    }
    txtJson =  result;
}
```

❑ Start(String text, String text1)

```
public void start(String text,String text1){
    new MyFetcherJsonFunction().execute(text, text1);
}
```

Tale funzione sarà chiamata nella onCreate di MenuActivity.
La stringa txtJson da qui ottenuta verrà a questo punto passata all'activity successiva utilizzando il meccanismo degli ''extra"

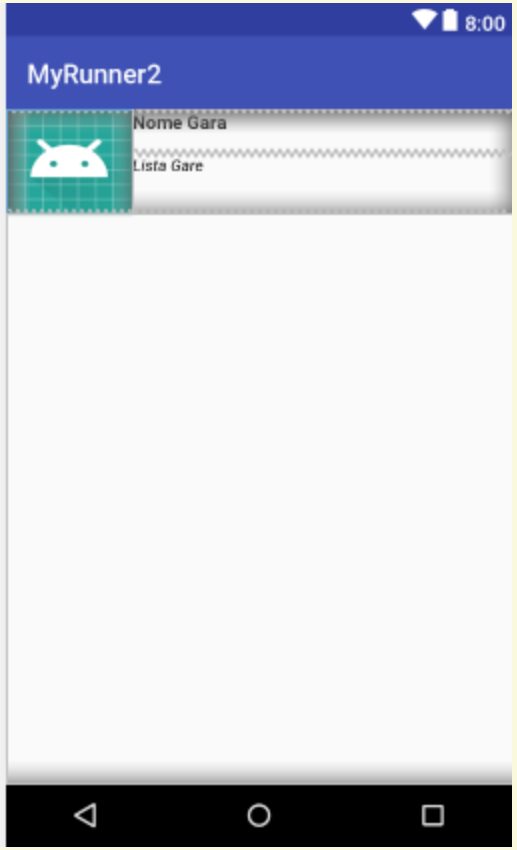
```
switch (text.substring(18)) {
            case "Elenco Soci":
                break;
            case "Prossime Gare":
                Intent intent2 = new
Intent(MenuActivity.this, GareActivity.class);
                intent2.putExtra("stringaJSON",txtJson);
intent2.putExtra("mail",mail);
                intent2.putExtra("pass",pass);
                intent2.putExtra("stringaJSON",txtJson);
                startActivity(intent2);
                break;

            case "Risultati":
                break;
}
```

Runner2 – Lista delle gare disponbili – GareActivity

In questa activity, come richiesto da traccia abbiamo creato una ListView con all'interno tutte le gare in programma.

Per il riempimento ci siamo serviti di una classe chiamata CustomAdapter, la quale consente, per ogni elemento della lista di avere una piccola immagine, nome della gara (evidenziato in grassetto) e altre informazioni correlate.



Runner2 – Lista delle gare disponbili – GareActivity

La stringa passata dall'activity precedente viene utilizzata per creare un JSONArray, il quale servirà poi per creare una lista di oggetti di tipo Gara.

Ciascuna Gara creata, a seconda della posizione nella lista, verrà passata all'activity successiva, per consentire di visualizzare ulteriori informazioni (immagine più grande, url della gara, data di scadenza e breve commento), sempre usando il meccanismo degli extra.

L'immagine verrà scaricata direttamente dall'url selezionato attraverso una classe che estende l'AsyncTask.

Breve esempio:

```
case 0:
                    Gara g1 = gare.get(0);
                    Intent intent = new
Intent(GareActivity.this, InfoGaraActivity.class);
                    intent.putExtra("id",g1.getId());
intent.putExtra("usermail",mail);
                    intent.putExtra("password",pass);
 intent.putExtra("commento",g1.getCommento());
                    intent.putExtra("data",g1.getData());

intent.putExtra("distanza",g1.getDistanza());

intent.putExtra("localita",g1.getLocalita());
                    intent.putExtra("nome",g1.getNome());

intent.putExtra("scadenza",g1.getScadenzaIscrizione());
                    intent.putExtra("url",g1.getUrlGara());

intent.putExtra("immagine",g1.getPicture());
                    startActivity(intent);
```

Runner2 – Schermata della gara e Prenotazione

Al click su ‘’Prenota Gara” viene attuata la POST, ovvero un’attività di tipo REST che consente di inserire sul JSON scelto, un nuovo oggetto di tipo JSON, dove appaiono le informazioni dell’utente che si iscrive (nel nostro caso username e password) e le informazioni della gara.

Runner2 – POST – Configurazione Server.php

```
<?php
if (!empty($_POST)) { // Attendo la ricezione di una richiesta POST


                $arr = array('email' => $_POST['email'], 'password' => $_POST['password'], 'idrace' => $_POST['idrace'], 'description' =>
$_POST['Description'], 'location' => $_POST['Location'],
                                                'distance' => $_POST['Distance'], 'date' => $_POST['Date'], 'before' => $_POST['Before'], 'url' =>
$_POST['url'], 'note' => $_POST['Note'], 'picture' => $_POST['picture']);
                $arr['picture'] = stripslashes($arr['picture']);
                $arr['date'] = stripslashes($arr['date']);
                $arr['before'] = stripslashes($arr['before']);
                $arr['url'] = stripslashes($arr['url']);


   $str = file_get_contents('Iscrizioni.json'); // Carico il contenuto del file 'Iscrizioni.json'
   $jsonDecod = json_decode($str, JSON_UNESCAPED_SLASHES); // Trasformo il file .json in un array
   array_push($jsonDecod, $arr); // Aggiungo il mio array a quelle precedentemente presenti
   $str = json_encode($jsonDecod, JSON_UNESCAPED_SLASHES); // Ricodifico il nuovo Array in json
   $file = fopen('Iscrizioni.json','w'); // Apro il file per la scrittura
   fwrite($file, $str); // Sovrascrivo il contenuto del file
   fclose($file); // Chiudo il file
                }
                else
                {

                                print("Non ho ricevuto alcun POST!");
}
?>
```

Runner2 – POST - AsyncTask

Per effettuare la POST abbiamo bisogno di creare di nuovo una connessione HTTP con il server.

Attraverso una connessione con una HTTPost al giusto indirizzo possiamo capire il tipo di oggetto con cui interagiamo.

```
List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(11);
```

```
httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
```

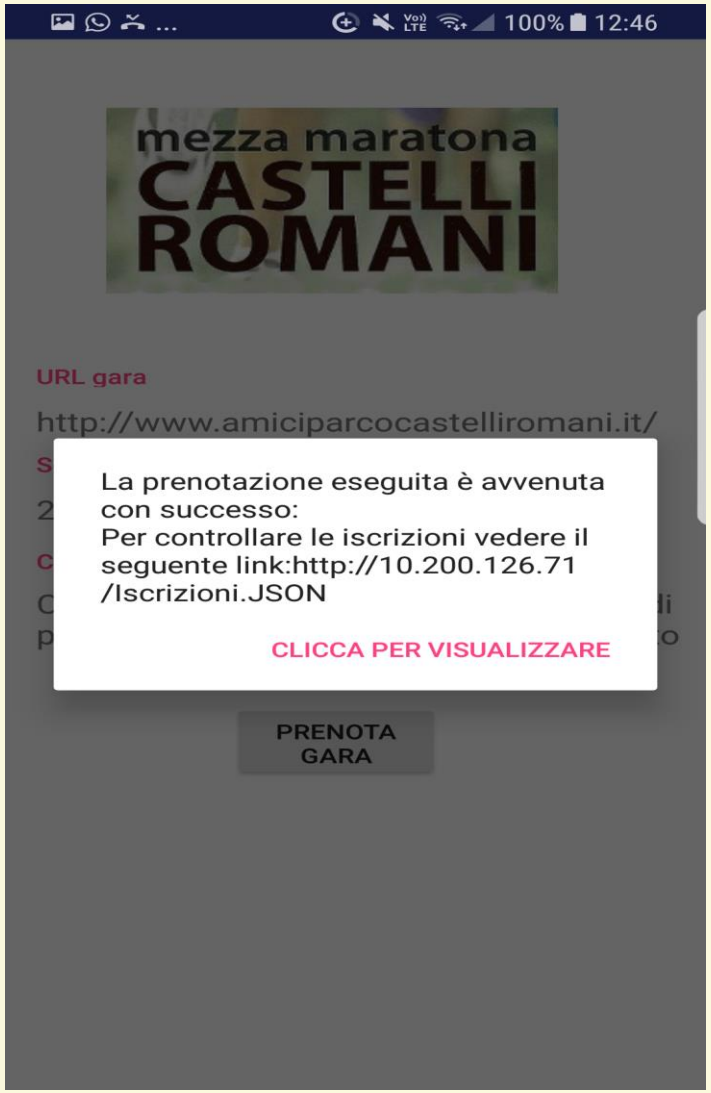
Viene settata l'entità passandogli le informazioni della lista che abbiamo creato sopra, di 11 valori (informazioni utente e gara).

```
HttpResponse response = httpclient.execute(httppost);
```

Esecuzione della post vera e propria.

Runner2 – Schermata della gara e Prenotazione - POST

```
 AlertDialog.Builder builder = new
AlertDialog.Builder(InfoGaraActivity.this);
            builder.setMessage("La prenotazione eseguita è
avvenuta con successo" + ":\n" +
Html.fromHtml("<center>" + "Per controllare le iscrizioni
vedere il seguente link: "+my_url+ my_url2 + "</center>"))
                    .setCancelable(false)
                    .setPositiveButton(‘’Prenotazione avvenuta
con successo!", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
int id) {
Intent viewIntent = new
Intent("android.intent.action.VIEW", Uri.parse(my_url+
my_url2));
                            startActivity(viewIntent);
}
                    });
            AlertDialog alert = builder.create();
            alert.show();
```



Così facendo possiamo accedere alla pagina web interessata che contiene le informazioni che abbiamo inserito.

Runner2 – Bibliografia

- http://www.andrious.com/tutorials/listview-tutorial-with-example-in-android-studio/
- http://www.html.it/pag/57841/le-permission/
- https://loginactivity.blogspot.com/
- http://www.mkyong.com/webservices/jax-rs/restfull-java-client-with-java-net-url/
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON
- https://stackoverflow.com/questions/38895663/get-object-list-with-lists-from-json-with-json-simple
- https://www.mrwebmaster.it/android/accesso-risorse-remote-tramite-http_11734.html
- https://stackoverflow.com/questions/12903244/send-data-using-post-method-android-to-php
- https://www.mrwebmaster.it/android/intent-browser-web-page-url_11503.html