UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato finale in **Ingegneria del software**

Tecniche e strumenti di progettazione per applicazioni Android

Anno Accademico 2017/2018

Candidato:

Andrea di Francia

matr. N46/001473

“A chi la sta leggendo”

Indice

Introduzione

Negli ultimi anni si è verificato un notevole sviluppo della tecnologia “mobile”. I dispositivi mobili rivestono un ruolo sempre più importante sia nel mondo aziendale che in quello privato, permettendoci di svolgere attività che erano eseguibili solo attraverso PC.

La motivazione che mi ha spinto a occuparmi dell’ambito Android è stata quella di studiare tramite le applicazioni mobili come sia possibile sviluppare applicazioni Client-Server, laddove il lato Client sia accessibile su un dispositivo mobile, a tal proposito Android è uno dei framework che consentono la programmazione del lato Client, in particolar modo dell’interfaccia utente, disponibile sul dispositivo mobile.

Lo strumento che mi ha permesso di realizzare i servizi lato server è Firebase, una piattaforma online completamente gratuita messa a disposizione da Google, in quest’ambito ho adoperato soluzioni ai problemi specifici di Authentication, Database e Storage.

Un ulteriore aspetto che ho preso in analisi è la programmazione di un’applicazione che detiene tutti i suoi contenuti staticamente cablati all’interno dell’applicazione, ad un’architettura distribuita nella quale alcuni contenuti come le preferenze degli utenti o riguardanti le domande, siano posizionati dinamicamente su risorse persistenti accessibili a tutti.

Infine ho trovato le soluzioni tecnologiche tramite un caso di studio corrispondente alla realizzazione di un applicazione mobile che ne fa uso.

Capitolo 1: Introduzione ad Android

In questo capitolo sono illustrate le caratteristiche del sistema relative alla piattaforma client-server e gli strumenti di sviluppo utilizzati nell'ambito Android.

1.1 Android e Java

Android è una piattaforma per telefoni cellulari, basata sul sistema operativo Linux ed è sviluppata dall'Open Handset Alliance.

Una caratteristica per la quale ho scelto il modello di Android, è che si differenzia da altre piattaforme, per il supporto e il riutilizzo dei componenti. Con componente si intende un qualsiasi programma, applicazione o servizio implementati. Le applicazioni non sono più pensate come entità a sé stanti, ma come un insieme di componenti disponibili nel sistema e richiamabili all'occorrenza. ne scaturisce un vantaggio non indifferente per lo sviluppatore, che si traduce in una maggior velocità di realizzazione di un applicazione.

Nel momento in cui viene richiesto uno specifico componente di una applicazione, Android fa si che il processo legato a questa sia in esecuzione, inizializzandolo se necessario.

Tutti i componenti di un’applicazione hanno un ciclo di vita che ha inizio nel momento in cui Android le istanzia con lo scopo di rispondere ad un intent e termina quando le loro istanza vengono distrutte.

Le applicazioni vengono sviluppate in Android Studio (fig.1) che utilizza il linguaggio Java, che rispetto al C, gestisce automaticamente il ciclo di vita di un’applicazione e anche l’allocazione della memoria, rendendo più semplice lo sviluppo. Android, inoltre è completamente open source, quindi rende possibile reperire, modificare e ridistribuire il codice sorgente, adattandolo ad ogni dispositivo e creandone la propria versione personalizzata ed ottimizzata. E´ anche il sistema operativo mobile più diffuso, ha una gestione di grafica e audio di alta qualità, le applicazioni native e di terze parti sono di uguale importanza e possono essere aggiunte o rimosse senza alcun vincolo.

1.2 Firebase

Firebase (fig.2) è un servizio di Google che supporta gli sviluppatori di app tramite la gestione della parte server e in particolare della parte strutturale; il servizio fornisce le risorse server in base al numero di utenti attivi, e gestisce quindi in automatico le risorse utilizzate.

Inoltre risolve molti dei problemi di chi fa product e mobile marketing perché integra nella medesima piattaforma una serie di funzionalità che prima erano sparse tra strumenti diversi, caratteristica questa che implicava effort molto elevati in termini di implementazione tecnica, capacità di far interagire correttamente gli strumenti tra loro, complessità di lettura e interpretazione dei dati.

I dati immagazzinati in Firebase sono replicati e sottoposti a backup continuamente, la comunicazione con i client avviene sempre in modalità crittografata tramite SSL con certificati a 2048-bit, evitando quindi allo sviluppatore di provvedere a queste problematiche.

Nella mia applicazione ho quindi scelto i servizi di questa piattaforma sicura e flessibile, ma anche completamente gratuita con il pacchetto "Hacker Plan" che mette a disposizione un massimo di 50 connessioni, 5 GB di trasferimento dati e 100 MB di Storage.

Figura n.1

Figura n.2

Capitolo 2: Firebase Tools

Di seguito elencherò nel dettaglio gli strumenti architetturali, messi a disposizione da Firebase, che ho deciso di adoperare per far fronte ai problemi di Autantication, Database e Storage.

2.1 Firebase Authentication

La maggior parte delle applicazioni ha bisogno di conoscere l'identità di un utente. Conoscere l'identità di un utente consente a un'applicazione di salvare saldamente i dati utente nella cloud e fornire la stessa esperienza personalizzata in tutti i dispositivi dell'utente.

L'autenticazione Firebase fornisce servizi di back-end, SDK di facile utilizzo e librerie UI pronte per autenticare gli utenti nell'applicazione. Supporta l'autenticazione utilizzando password, numeri di telefono, fornitori di identità federali popolari come Google, Facebook e Twitter e altro ancora.

Per registrare un utente nell'applicazione, è necessario ottenere le credenziali di autenticazione dall'utente. Queste credenziali possono essere l'indirizzo email e la password dell'utente o un token OAuth da un provider certificato. Quindi, passate queste credenziali all'ID SDK di autenticazione Firebase. I nostri servizi di back-up verificheranno quindi le credenziali e restituiranno una risposta al cliente.

Dopo avere eseguito un accesso effettivo, è possibile accedere alle informazioni di base del profilo dell'utente e controllare l'accesso dell'utente ai dati memorizzati in altri prodotti Firebase. È inoltre possibile utilizzare il token di autenticazione fornito per verificare l'identità degli utenti nei propri servizi di back-end.

Nella applicazione la registrazione tramite Google (Google Sign-in), prima di poterla implementare si devono seguire i seguenti passaggi:

1. Si aggiunge Firebase al proprio progetto
2. Si inseriscono le dipendenze per Firebase Authentication e Google Sign-In del file build.gradle
3. Si specifica il codice SHA-1
4. Abilito Google Sign-in nella console di Firebase (fig.3)

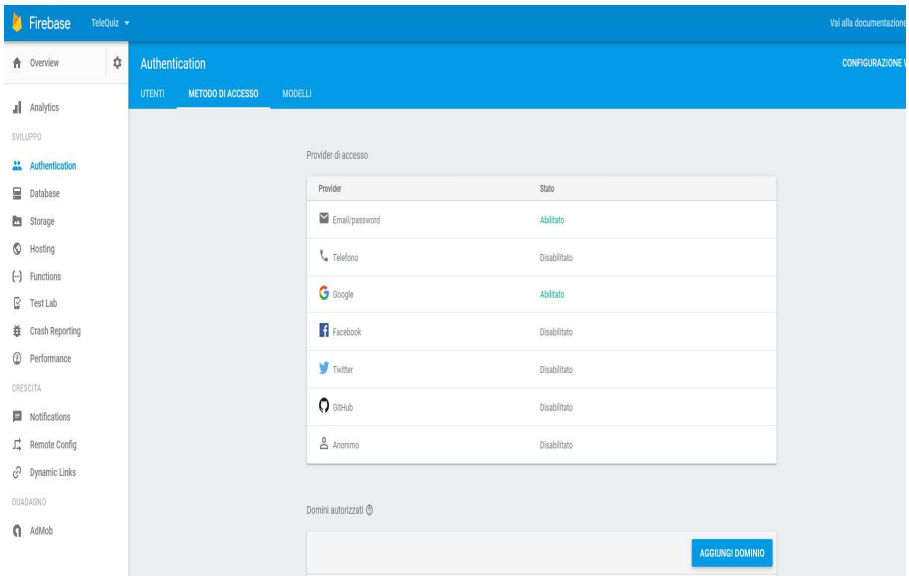

Figura n.3

2.2 Implementazione Authentication

Effettuate queste operazioni non resta che creare un apposito pulsante Google da inserire nell'Activity main.

```
<com.google.android.gms.common.SignInButton

   android:id="@+id/googlebutton"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="94dp"
   android:layout_alignParentBottom="true"
   android:layout_alignParentStart="true">

</com.google.android.gms.common.SignInButton>
```

Ed infine bisogna implementare le funzioni, qui di seguito, riporto il codice presente nella documentazione Firebase.

```
  gbutton = (SignInButton) findViewById(R.id.googlebutton);
  mAuth = FirebaseAuth.getInstance();

  gbutton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
  signIn();
    }
  });


   mAuthListner = new FirebaseAuth.AuthStateListener() {
   @Override
   public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {

   if (firebaseAuth.getCurrentUser() != null) {

            }

        }
    };

    GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN
)

.requestIdToken(getString(R.string.default_web_client_id))
```

```
            .requestEmail()
            .build();

mGoogleApiClient = new GoogleApiClient.Builder(this)
            .enableAutoManage(this /* FragmentActivity */, new
GoogleApiClient.OnConnectionFailedListener() {
@Override
public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {

                }
            })
            .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
            .build();

}
```

Nelle funzioni precedenti prima di tutto si inizializza il pulsante di Google e poi un istanza di *FirebaseAuth.* Poi in seguito si gestisce il tocco del bottone andando a richiamare la funzione *signIn()* che andrà a restituire o meno l'account registrato dell'utente, nella seconda funzione si effettua un controllo tramite la funzione *firebaseAuth.getCurrentUser()* si va a prelevare le informazioni relative all'account e si effettua un controllo di validità di tali informazioni.

Il resto del codice è presente nella documentazione messa a disposizione da Firebase presenta ulteriori funzioni per la gestione delle credenziali.

```
private void signIn() {
Intent signInIntent =
Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
public void onActivityResult(int requestCode, int resultCode,
Intent data) {
super.onActivityResult(requestCode, resultCode, data);

// Result returned from launching the Intent from
GoogleSignInApi.getSignInIntent(...);
if (requestCode == RC_SIGN_IN) {
GoogleSignInResult result =
Auth.GoogleSignInApi.getSignInResultFromIntent(data);
```

```
 if (result.isSuccess()) {
   // Google Sign In was successful, authenticate with Firebase
   GoogleSignInAccount account = result.getSignInAccount();
   firebaseAuthWithGoogle(account);
        } else {
            Toast.makeText(MainActivity.this, "Fail SIGN IN ",
Toast.LENGTH_SHORT).show();
        }
    }
}

private void firebaseAuthWithGoogle(GoogleSignInAccount
account) {

AuthCredential credential =
GoogleAuthProvider.getCredential(account.getIdToken(), null);
 mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
@Override
public void onComplete(@NonNull Task<AuthResult> task) {
if (task.isSuccessful()) {
// Sign in success, update UI with the signed-in user's
information
Toast.makeText(MainActivity.this, " Sign in Success ",
Toast.LENGTH_SHORT).show();
Log.d("TAG", "signInWithCredential:success");
FirebaseUser user = mAuth.getCurrentUser();

}
 else {
  // If sign in fails, display a message to the user.
 Log.w("TAG" , "signInWithCredential:failure",
task.getException());
Toast.makeText(MainActivity.this, "Authentication failed.",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
}
```

Il risultato in Firebase é riportato nella Fig.4, dove possiamo notare l'aggiunta di due profili Google.

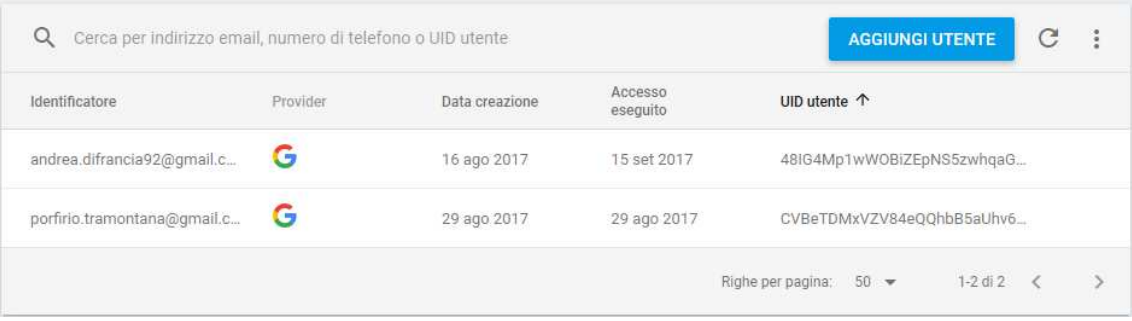

Fig.4

2.2 Firebase Realtime Database

Il database Firebase Realtime (fig.5) é un database di tipo Nosql, dove la persistenza dei dati è caratterizzata dal fatto di non utilizzare il modello relazionale, esso é ospitato da cloud, i dati vengono memorizzati come JSON e sincronizzati in tempo reale per ogni client connesso.

Quando si creano applicazioni cross-platform con SDK di iOS, Android e JavaScript, tutti i clienti condividono un'istanza di database Realtime e ricevono automaticamente gli aggiornamenti con i dati più recenti.

Il database mette a disposizione oltre alla parte dati, anche una sezione dedicata alle regole, quest'ultime consentono di definire come i dati devono esser strutturati, indicizzati e se i dati possono esser letti o scritti. Per impostazione predefinita, l'accesso in lettura e scrittura al database é limitato in modo che solo gli utenti autenticati possano leggere o scrivere i dati. Ma è possibile configurare tale parametro come accesso pubblico, in tal caso i dati vengono aperti a chiunque, anche a persone che non utilizzano l'applicazione.

Questa tipologia di database è adatta alle esigenze di chi progetta una struttura dati non definibile a priori e in tal caso è necessario farla interagire con molta frequentemente con il database.

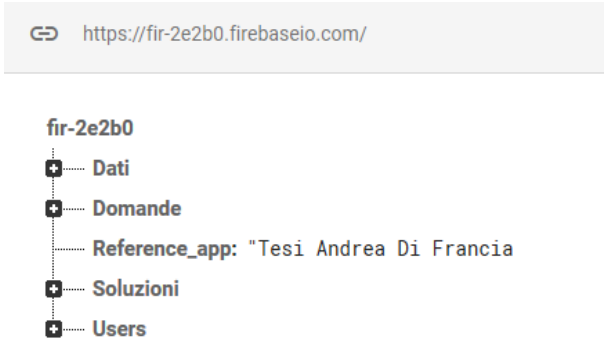

Fig.5

2.3 Implementazione Database

Per poter implementare il servizio Database bisogna prima di tutto andare a sincronizzare Android Studio con Firebase, ma nel caso in cui questo procedimento fosse già stato effettuato non ci sarà bisogno di doverlo ripetere.

Effettuate tutte le operazioni di sincronizzazione possiamo andare ad implementare le funzioni per inserire i dati nel database.

```
mFirebaseDatabase = mFirebaseInstance.getReference("Users");
private void createUser(String name, String email, int score) {
  userId = mFirebaseDatabase.push().getKey();
  User user = new User(name, email , score, userId);
  mFirebaseDatabase.child(userId).setValue(user);
  addUserChangeListener();
  }
```

In questa funzione bisogna andar ad inserire i valori che si vogliono salvare nel database. In seguito tramite le due funzioni *push()* e *getKey()*, messe a disposizione dalla libreria Firebase si inseriranno i dati all'interno del Database nel nodo che va a definire con la funzione *getReference()*.

Il risultato osservato nella schermata Firebase per ogni utente registrato è illustrato nella fig.6, dove possiamo vedere tutti i parametri che avevamo inserito in precedenza, ogni qual volta che ci sará un nuovo inserimento tale tabella verrá aggiornata inserendo un nuovo nodo, sempre figlio dell'istanza Users.

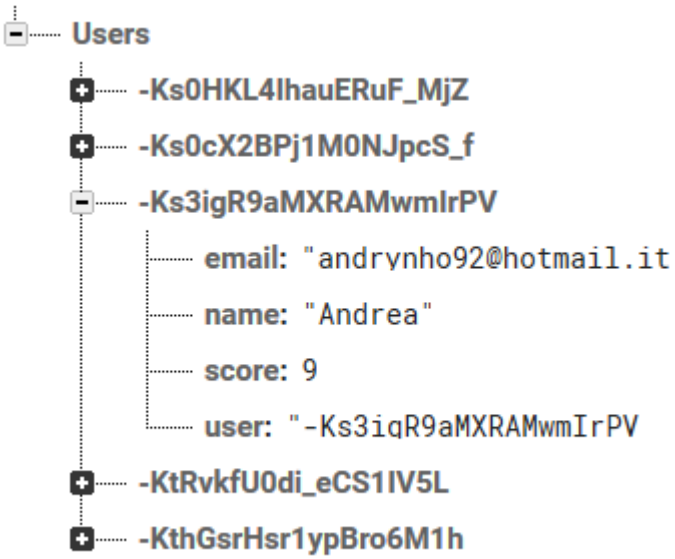

Fig.6

2.4 Firebase Cloud Storage

Cloud Storage è costruito per gli sviluppatori di applicazioni che hanno bisogno di memorizzare e pubblicare contenuti generati dagli utenti, ad esempio foto o video.

Le SDK di Firebase per Cloud Storage aggiungono la sicurezza di Google per upload e download di file per le applicazioni, indipendentemente dalla qualità della rete.

Questo strumento permette di salvare file multimediali nel server, facendo in modo che l'applicazione non sia eccessivamente grande in termini di riempimento della memoria del dispositivo, in quanto tali file vengono prelevati tramite apposite funzioni ed inseriti in cloud presso server messi a disposizione da Firebase. Inoltre questo strumento permette di aggiornare i contenuti multimediali senza dover apportare modifiche ai contenuti del dispositivo mobile.

Il pannello dello strumento si presenta in Fig.7 dove è possibile caricare file esclusivamente di estensione .png o eventualmente poter fare l'upload tramite applicazione, ma non ho implementato quest'ultima funzione.

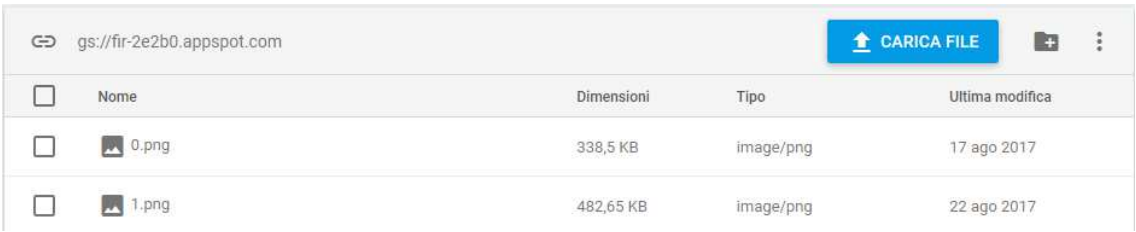

Fig.7

2.5 Implementazione Storage

Questa di seguito è un esempio implementativo del caso sopra citato, prima di tutto bisogna far riferimento allo Storage a cui si vuole far la richiesta, infine bisogna cercare il nodo con il nome indicato della funzione *storageRef.child().*

Se tale funzione va a buon fine, prima di tutto bisogna inizializzare una nuova variabile Bitmap e poi sará caricata nell'interfaccia tramite la funzione *setImageBitmap()*.

```
FirebaseStorage storage = FirebaseStorage.getInstance();
storageRef.child(url).getBytes(Long.MAX_VALUE).addOnSuccessListe
ner(

new OnSuccessListener<byte[]>() {

@Override
public void onSuccess(byte[] bytes) {
// Use the bytes to display the image
Bitmap bitmap = BitmapFactory.decodeByteArray(bytes, 0,
bytes.length);
immagine.setImageBitmap(bitmap);

    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        // Handle any errors
    }

});
```

Capitolo 3: TeleQuiz

Nell'ultimo capitolo descriverò le fasi di sviluppo che mi hanno permesso di realizzare *TeleQuiz*, l'applicazione da me ideata per andare ad implementare le soluzioni tecnologiche studiate. Partendo dall'analisi dei requisiti, in cui spiegherò da un punto di vista concettuale sul come funziona l'applicazione, fino ad arrivare alla fase di progettazione e implementazione in cui, attraverso opportuni diagrammi, andrò ad analizzare più nel dettaglio gli aspetti funzionali ed implementativi dell'applicazione stessa.

3.1 Analisi dei requisiti

TeleQuiz è ideata allo scopo di offrire all'utente un servizio di intrattenimento e svago. Essa consentirà a tutti gli appassionati dei Telefilm di poter mettersi in gioco e testare le proprie conoscenze attraverso domande di qualsiasi tipologia.

Ogni utente prima di poter accedere all'area quiz dovrà autenticarsi con i due metodi messi a disposizione: autenticazione classica o con Google.

Sarà anche prevista una classifica generale di tutti gli utenti registrati con i diversi punteggi da loro ottenuti nel test, nel caso in cui una persona non sia riuscita a migliorare il suo score quest'ultimo non sarà aggiornato.

3.2 Progettazione

Per capire al meglio come ho sviluppato TeleQuiz e comprendere i requisiti del progetto analizzerò i casi d'uso.

3.2.1 Use case Diagram

Il caso d'uso (use case diagram), in informatica, è una tecnica usata nei processi di ingegneria del software per effettuare, in maniera esaustiva e non ambigua, la raccolta dei requisiti al fine di produrre software di qualità. Vengono utilizzati per l'individuazione e la registrazione dei requisiti funzionali scrivendo come un sistema possa essere utilizzato per consentire agli utenti i raggiungere i loro obiettivi. Come si può notare dalla (fig. 8) é presente un attore: l'utente.

L' attore, una volta effettuata l'operazione di login, potrà accedere alle funzionalità offerte dall'app come poter accedere al quiz e sarà abilitato alla visualizzazione della classifica

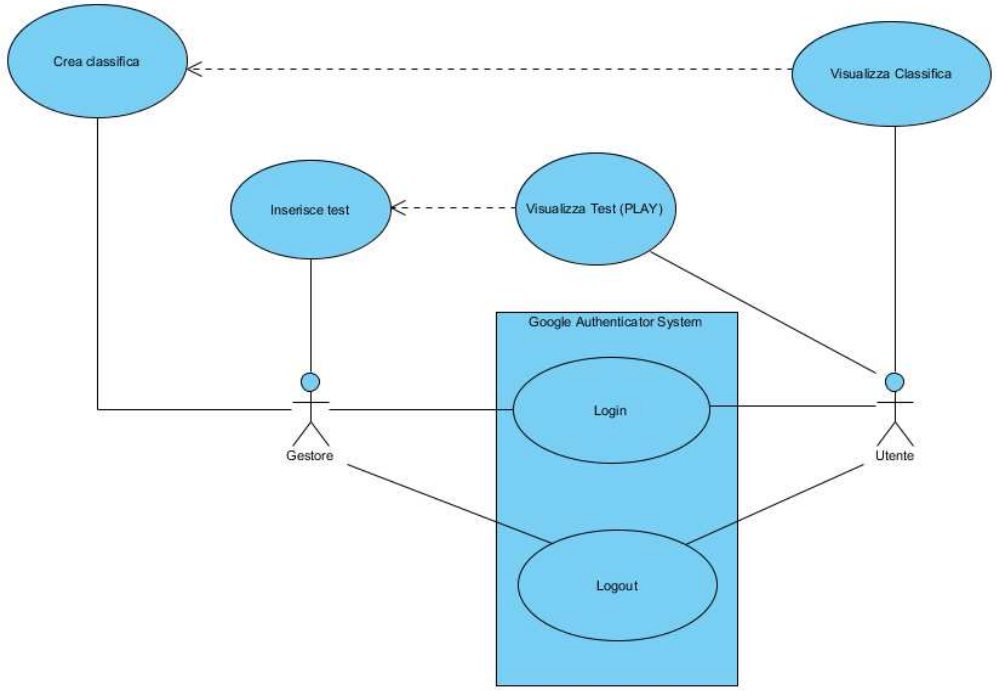

Fig.8

3.2.2 Class Diagram

Il Class Diagram presente in figura 9 mostra la struttura dell' app e la relazione fra i vari componenti. Le classi *Utente* e *Gestore* sono le uniche classi con attributi in quanto descrivono le caratteristiche del cliente.

La classe *Login* serve per la registrazione e l' autenticazione dell'utente mentre la classe *Logout* permette all'utente di potersi de-autenticare e poter scegliere un altro account.

Crea Classifica e *Inserisci Test*, infine, permettono al *Gestore* di poter creare la classifica con tutti i nomi e gli annessi punteggi totali mentre la seconda classe permette di poter visualizzare nell' interfaccia le domande e le immagini del test. L' utente potrà accedere a queste classi private tramite classi pubbliche quali *Visualizza Classifica* e *Visualizza Dati*.

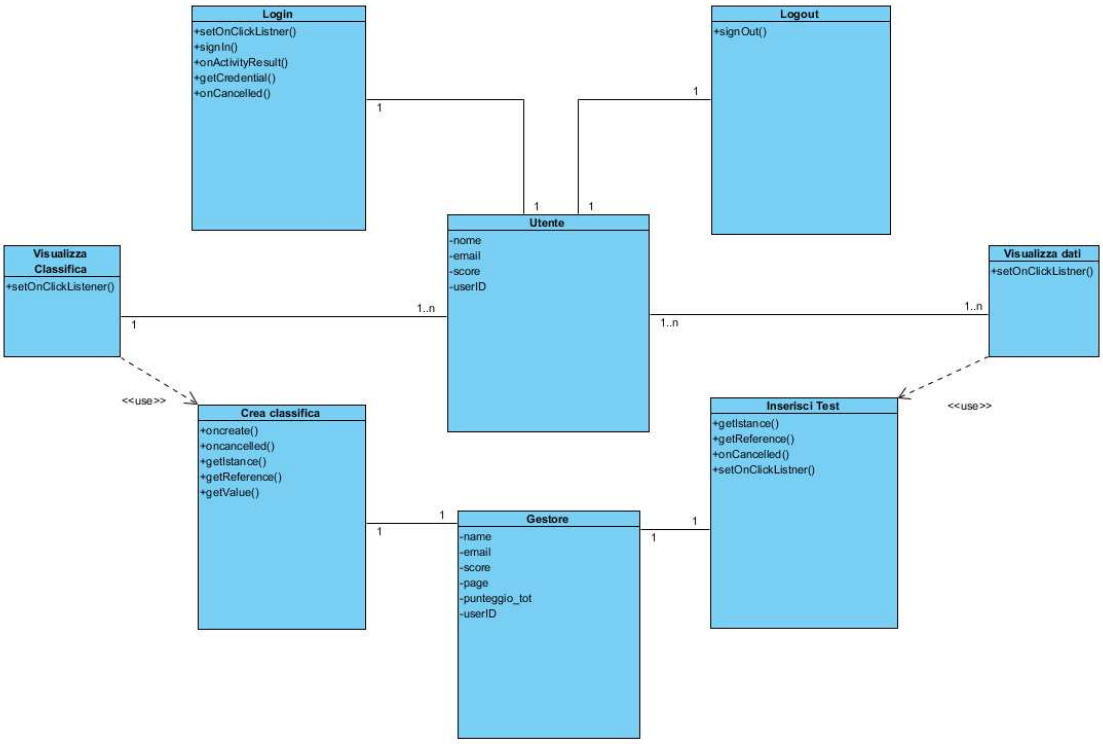

Fig.9

3.3 Implementazione

Entriamo adesso nel cuore dell' app analizzando nel dettaglio il codice per l'implementazione di tutti i casi d' uso.

3.3.1 Fragment Main

Qui di seguito, riporto il codice del *MainActivity* relativo alla registrazione che ho implementato tenendo conto dei casi in cui il nickname e l'e-mail era vuoti oppure che il nickname era già presente nel database.

```
btnSave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        final String name = inputName.getText().toString();
        final String email = inputEmail.getText().toString();


 if(name.equals("") || email.equals("")){
  Toast.makeText(MainActivity.this , "name o email vuoti Re-
Inserire i dati",Toast.LENGTH_SHORT).show();
            startActivity(new Intent(MainActivity.this,
MainActivity.class));

        }
        else {

 final DatabaseReference myRef =
mFirebaseInstance.getReference("Users");
 Query query = myRef.orderByChild("name").equalTo(name);


   query.addListenerForSingleValueEvent(new ValueEventListener()
{
 @Override
public void onDataChange(DataSnapshot dataSnapshot) {

for (DataSnapshot messageSnapshot: dataSnapshot.getChildren()) {

   String name2 = (String)
messageSnapshot.child("name").getValue();


  if(name.equals(name2)) {

 Toast.makeText(MainActivity.this , name+" UTENTE GIÁ ESISTENTE
",Toast.LENGTH_SHORT).show();
```

```
flag=1;
startActivity(new Intent(MainActivity.this,
MainActivity.class));
                        }
                    }
 if(flag==0){

createUser(name, email, score);

Intent nuovaPagina = new Intent(MainActivity.this,
Main2Activity.class);


                        nuovaPagina.putExtra("key1", name);
                        nuovaPagina.putExtra("key2", email);

                        startActivity(nuovaPagina);
                    }

                }

@Override
public void onCancelled(DatabaseError databaseError) {

                }
            });
```

L'utente che invece già ha effettuato la registrazione potrà effettuare semplicemente il log-in. Per poter capire se le credenziali inserite sono corrette ci sarà una query all' interno del database andando a selezionare il nodo *Users* e l' attributo name, di seguito il relativo il codice.

```
  login.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {

         String name = inputName.getText().toString();
         String email = inputEmail.getText().toString();

         //Query per verificare le credenziali
   final DatabaseReference myRef =
 mFirebaseInstance.getReference("Users");

  Query query = myRef.orderByChild("name").equalTo(name);

 query.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot)
 {
 for (DataSnapshot messageSnapshot: dataSnapshot.getChildren())
 {
```

```
String name2 = (String)
messageSnapshot.child("name").getValue();
String email2 = (String)
messageSnapshot.child("email").getValue();

Long score = (Long) messageSnapshot.child("score").getValue();

Intent nuovaPagina = new Intent(MainActivity.this,
Main2Activity.class);

                    nuovaPagina.putExtra("key1",name2);
                    nuovaPagina.putExtra("key2",email2);
                    nuovaPagina.putExtra("score",score);

                    startActivity(nuovaPagina);
                }

            }
```

Se il nome inserito verrà trovato all'interno del database allora sarà permesso l'accesso all' area menu dell'applicazione, passando alla nuova Activity, tramite la funzione *startActivity()* e *putExtras()*, i valori del nome, email e punteggio dell'utente.

Mentre se tale valore non è presente del database all'atto della query verrà visualizzato un messaggio di errore di registrazione.

3.3.2 Classe User

Di seguito si riporta il codice relativo alla *classe User* funzionale alla creazione dell' oggetto *utente*:

```
public class User {

    public String name;
    public String email;
    public int score;
    public String user;

    public User() {
    }

    public User(String name, String email, int score, String user) {
        this.name = name;
        this.email = email;
        this.score = 0;
        this.user = user;
    }
}
```

Come si può notare dal codice , in essa troviamo i costruttori con e senza argomenti e i quattro metodi relativi alle variabili dell'oggetto *User* i quali ritornano rispettivamente il nome, l'e-mail, l'inizializzazione dello score iniziale e il codice univoco dell'utente.

3.3.3 Fragment Menù

Mentre nella seconda Activity ho implementato l'accesso al test, visualizzazione della classifica e il logout.

```
button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

mAuth.signOut();
startActivity(new Intent(Main2Activity.this,
MainActivity.class));

        }
    });

 //Test Implementation
  Button play = (Button) findViewById(R.id.Playtest);

  play.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {

 final DatabaseReference Refndom =
mFirebaseInstance.getReference("Dati");
  final Query queryn = Refndom.orderByChild("numero_domande");

queryn.addListenerForSingleValueEvent(new ValueEventListener()
{

@Override
public void onDataChange(DataSnapshot dataSnapshot) {

for (DataSnapshot messageSnapshot : dataSnapshot.getChildren())
{
  long numero_domande = (long) messageSnapshot.getValue();

 Intent Play = new Intent(Main2Activity.this, PlayTest.class);

   Play.putExtra("punteggio_tot", punteggio_tot);
   Play.putExtra("key1",value1);
   Play.putExtra("User",user );
   int numerodomande= (int)numero_domande;

Play.putExtra("ndomande",numerodomande);

 startActivity(Play);
                        }
                    }
```

```
 @Override
  public void onCancelled(DatabaseError databaseError) {

                    }
                });

            }
        });
 //Classifica Implementation
 Button classifica = (Button) findViewById(R.id.Classifica);
 classifica.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {

Intent Best = new Intent(Main2Activity.this, Classifica.class);

Best.putExtra("punteggio_tot", punteggio_tot);
Best.putExtra("key1",value1);
Best.putExtra("User",user );

startActivity(Best);

            }
        });
    }
```

L'implementazione del bottone per l'accesso al test permette una volta premuto di fare una query al database per ottenere il numero di risposte presenti e in seguito passare tramite la funzione *putExtra()* i valori del punteggio, del nome e dell'identificativo utente, in maniera uguale è implementato anche il pulsante per la visualizzazione della classifica degli utenti.

Per il logout è implementata la funzione *mAuth.signOut()* per scollegare l'utente e *StartActivity()* per ritornare alla pagina iniziale.

3.3.4 Fragment PlayTest

In questa pagina mi sono sincronizzato con Firebase in modo da poter caricare a video l'immagine e la domanda corrispondenti ad un numero generato dalla funzione *random()*, ciò mi ha permesso di prelevare domande casuali in modo da non annoiare.

Ho prima di tutto recuperato i dati passati da un'altra Activity tramite la funzione *getExtras()* e di seguito ho inizializzato tutte le variabili.

```
    Bundle ex = getIntent().getExtras();
    // il numero di punti che ha totalizzato
    int point = ex.getInt("point");
    // numero di volte che si sta ripetendo la pagina
    int pagine = ex.getInt("page");
    //numero di domande presenti nel database
    final int n_domande = ex.getInt("ndomande");
    final String value1=ex.getString("key1");//nome dell'utente
    //identificare l'utente
    final String user=ex.getString("user");
    int punteggio_tot = ex.getInt("punteggio_tot");
    punti = point;
    page = pagine;
    final TextView punt = (TextView) findViewById(R.id.score);
    String puntitxt = ""+ punti;
    punt.setText(puntitxt);


mFirebaseInstance = FirebaseDatabase.getInstance();

Random random = new Random();
final int x = random.nextInt( n_domande);

final String y = x + "";
String url = x + ".png";
```

Per il passaggio del numero di domande che sono presenti nel database ho utilizzato la funzione *random.nextInt()*, inizialmente avevo previsto una variabile globale intera *numero_domande*, ma questa soluzione mi portava ad ogni aggiunta di una nuova domanda, al dover ricompilare tutto il codice per poter modificare il valore di quella variabile.

Per questo motivo ho adottato un' altra soluzione che prevede l'utilizzo del database di Firebase. Ho creato un nuovo nodo rinominandolo *Dati* nel quale ho inserito il numero delle domande che erano presenti del nodo *Domande*, in questo modo tramite una semplice query è possibile ricavare tale valore senza dover andare a modificare il codice dell'applicazione.

Infine ho implementato il bottone *buttonrisp* di modo da poter controllare tramite query se la risposta dell'utente è corretta o meno ed aggiornare la variabile *punti*.

Per poter caricare altre domande ho anche previsto una variabile *page* di modo da fermare il quiz dopo un numero prestabilito di domande e alla fine passare tutti i valori nel MenuActivity.

```
final EditText risp = (EditText) findViewById(R.id.risposta);
Button buttonrisp = (Button) findViewById(R.id.ButtonRisposta);
buttonrisp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

 final String risposta_utente = risp.getText().toString();

Query query = Refsol.orderByKey().equalTo(y);
query.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(DataSnapshot dataSnapshot) {

for (DataSnapshot messageSnapshot : dataSnapshot.getChildren())
{

String risposta_corretta = (String) messageSnapshot.getValue();
```

```
if (risposta_utente.equals(risposta_corretta)) {
 punti = punti + 3;
 Toast.makeText(PlayTest.this, "Found: " + risposta_corretta +
"punti " + punti, Toast.LENGTH_LONG).show();

 page=page+1;

 if(page ==5){
Intent Finish = new Intent(PlayTest.this, Main2Activity.class);
Finish.putExtra("point", punti);
Finish.putExtra("key1", value1);
Finish.putExtra("user",user);
startActivity(Finish);
                        }
  else {
Intent Refresh = new Intent(PlayTest.this, PlayTest.class);
startActivity(Refresh);
                        }
                    } else {
Toast.makeText(PlayTest.this, "WRONG " + risposta_utente +
"CORRECT " + risposta_corretta, Toast.LENGTH_LONG).show();


           }

       }
```

Nell' ultima parte di codice ho dovuto condizionare tutti i possibili casi del tipo di risposta che l'utente ha selezionato, quindi se ha risposto nel modo corretto e quindi aggiornare la variabile *punti* oppure ha risposto in modo sbagliato e quindi bisognerà soltanto aggiornare la pagina con una nuova domanda.

In entrambi i casi ci sarà l' incremento della variabile *page* che determinerà o meno la fine del test.

3.4 Le Activity

Illustrati i vari fragment, analizzeremo ora le activity presenti in *TeleQuiz*.

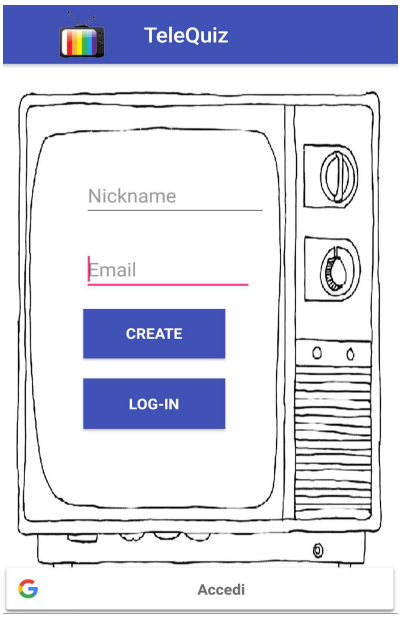

Fig.10

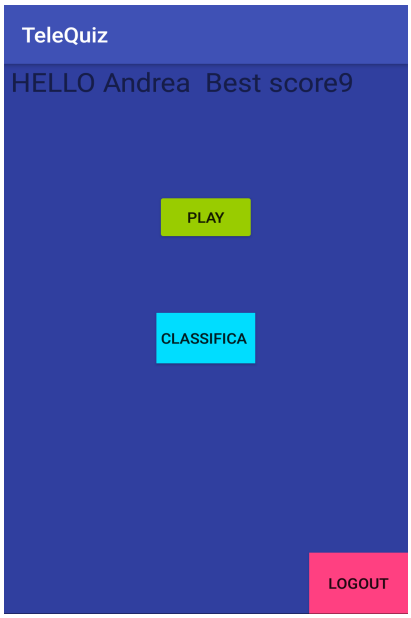

Fig.11

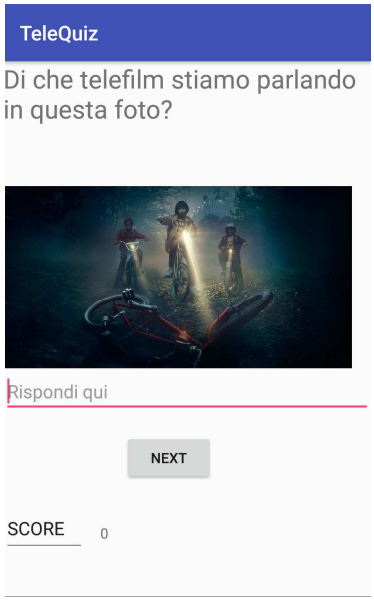

Fig.12

3.4.1 Activity Main

Questa Activity (fig. 10) è costituita sostanzialmente da due *TextView* e *EditText* per visualizzare ed inserire, il nickname e dell' e-mail.

In basso troviamo due *Button* per potersi registrare e poter accedere all'area menu, mentre più in basso troviamo il tasto Google che ci permetterà di accedere selezionando un profilo valido di Google.

3.4.2 Activity Menù

Come si può notare nella figura n.11, quest' activity prevede un campo *TextView* personalizzato dove comparirà il nome dell'utente e il suo punteggio migliore.

Di seguito ci saranno tre *Button* che permettono all'utente di poter iniziare il quiz se si preme il tasto *Play*, mentre se si vuole scoprire a che posizione si è arrivati bisognerà premere il tasto *Classifica.*

Infine se si vuole scollegare il proprio account con l' app, c'è il tasto *Logout*.

3.4.3 Activity Test

L'Activity test come riportato dalla fig.12, prevede una *TextView* per l' output della domanda, un *ImageView* per la visualizzazione della foto inerente alla domanda ed un *EditText* per poter rispondere al quesito.

Una volta inserita la risposta ci sarà il tasto *Next* per poter passare alla domanda successiva, se la risposta data sarà corretta lo *Score* formato da un *EditText* e un *TextView* aggiornato di 3 punti.

Conclusioni

Gli strumenti presi in esame sono stati in grado di risolvere i problemi specifici di Authentication, Database e Storage.

Tali soluzioni mi hanno permesso la realizzazione di un'applicazione che possiede un'architettura distribuita, tale architettura presenta molteplici aspetti positivi, tra i quali una ridotta dimensione totale dell'applicazione, avendo la parte multimediale nella parte client, e di conseguenza la possibilità di poter aggiornare tale parte senza dover modificare il codice della parte client.

In futuro potrebbero nascere ulteriori possibilità di studio con l'utilizzo di ulteriori servizi messi a disposizione da Firebase.

Bibliografia

[1] Html, http://www.html.it/articoli/firebase-2/

[2] Wikipedia, https://it.wikipedia.org/wiki/Android_Studio

[3] Firebase, https://firebase.google.com/docs/database/

[4] Firebase, https://firebase.google.com/docs/storage/

[5] Firebase, https://firebase.google.com/docs/auth/android/start/

[6] AndroidHive, https://www.androidhive.info/firebase-realtime-database/