UNIVERSITÀ DELLA CALABRIA

A.A. 2021/2022

Talk2Tag

Progetto Sistemi Distribuiti e Cloud Computing

Simone Giorgio 214575

Traccia Progetto

“Sviluppo di una app Android per l’annotazione vocale delle foto scattate.

Quando l’utente scatta una foto potrà registrare una parola o più parole associate alla foto e memorizzarle associandole alla foto.

L'app deve permettere di cercare le foto tramite una o più parole che l’utente potrà pronuciare.”

Descrizione Progetto

L'applicazione Android è stata sviluppata in *Flutter*, framework open-source creato da Google per la creazione di interfacce native per iOS e Android.

I package utilizzati per la realizzazione del progetto, pubblicati su *pub.dev,* sono*:*

- **image_picker: ^0.8.5+3** , un plugin per selezionare immagini dalla galleria e scattare nuove foto con la fotocamera.

- speech_to_text: ^5.5.0 , plugin che contiene un insieme di classi che semplificano la speech recognition, basato sullo SpeechRecognizer di Android. Utile per comandi e frasi corte, ma non per una conversazione continua o per rimanere sempre in ascolto.

Di fondamentale importanza per il funzionamento dell'applicazione sono due prodotti di Firebase, piattaforma per la creazione di applicazioni per dispositivi mobili e web sviluppata da Google.

- **Cloud Storage for Firebase:** progettato per gli sviluppatori di app che hanno bisogno di archiviare e fornire contenuti generati dagli utenti, come foto o video. Le API di Google Cloud Storage possono essere utilizzate per accedere ai file.
  Nella nostra applicazione è stata utilizzata quindi per memorizzare le foto scattate o selezionate dalla galleria a cui vogliamo associare un tag.

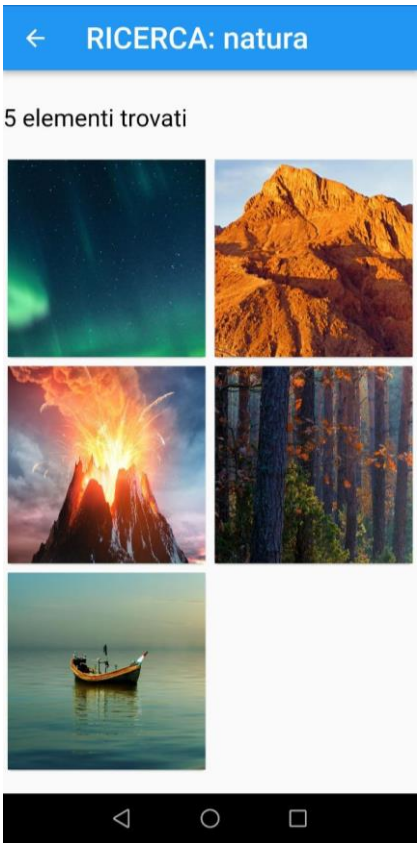
- **Cloud Firestore:** database NoSQL che permette di memorizzare, sincronizzare, ed effettuare query dei propri dati per la propria applicazione web/mobile.
  Nella nostra applicazione è utilizzata, come vedremo più avanti, per associare ad ogni tag ricercato la lista delle foto associate ad esso nel cloud storage, ed inoltre, per associare ad ogni foto nello storage la lista dei suoi tag associati, per fare in modo che, una volta aperta una foto, si abbia la possibilità di prendere visione ed interagire con i vari tag associati.

L’utente si interfaccia con una semplice ed intuitiva interfaccia, che permette subito di effettuare 3 scelte: selezionare una foto dalla galleria, scattarne una con la fotocamera, o effettuare una ricerca vocale delle foto pronunciando un tag precedentemente associato.

Grazie ad ImagePicker è stato possibile implementare in maniera semplice sia la possibilità di scegliere la foto dalla galleria, sia quella di scattare con la Fotocamera.

Supponiamo che l’utente effettui una ricerca vocale pronunciando la parola “natura”: il widget *ButtonRicerca in main.dart ,* dopo aver catturato la parola con *SpeechToText*, crea un child widget chiamando search.dart, il quale si occupa di recuperare le immagini associate ad esso: grazie ad uno streamBuilder su uno snapshot della collezione *tags* in Firestore, recuperiamo gli url delle immagini nel cloud che chiariremo tra un attimo come è strutturata.

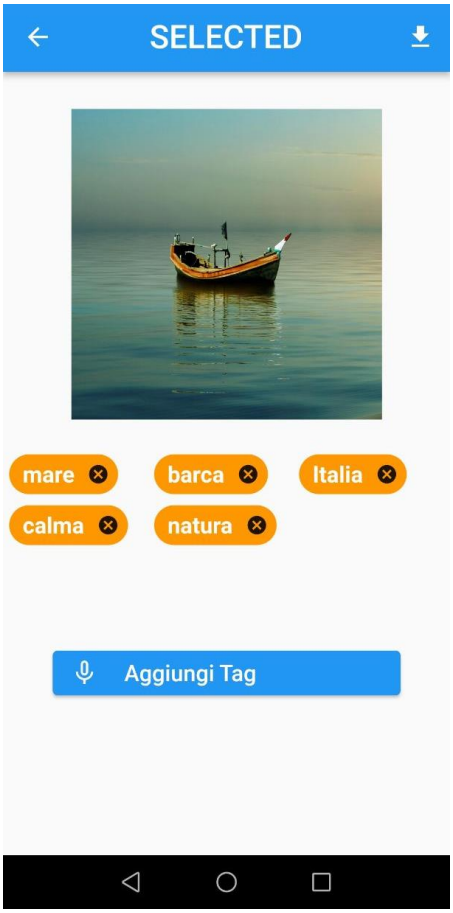
Il risultato della ricerca sarà questo:



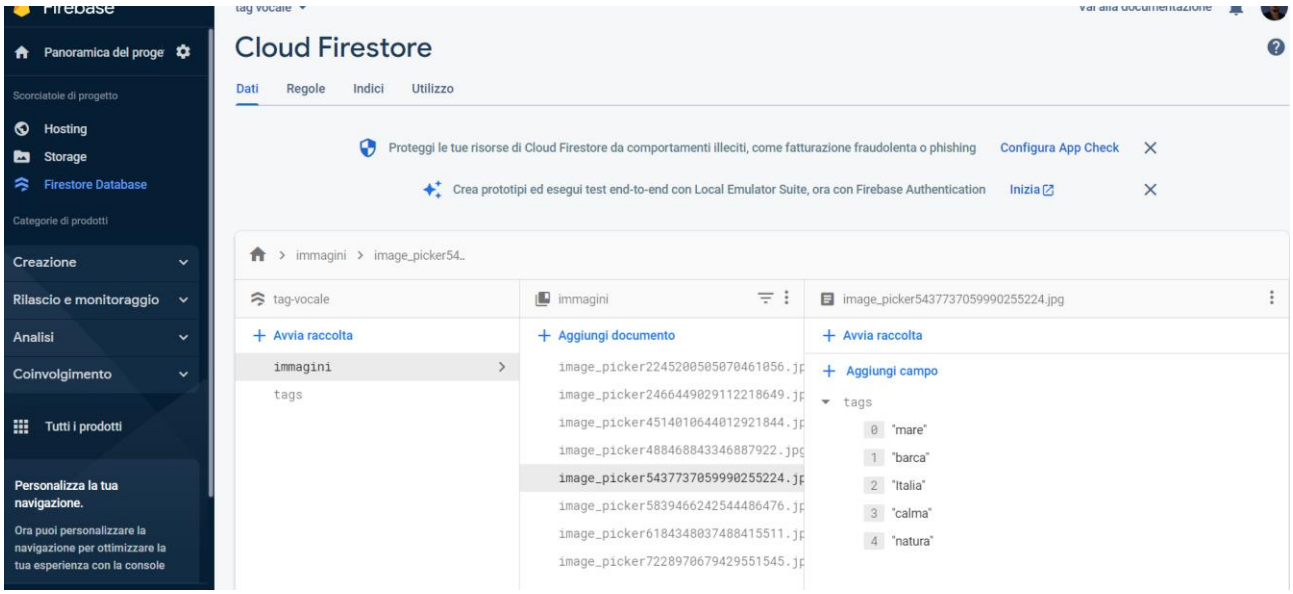
Cliccando sulla singola *Card*, verrà creato un nuovo child widget chiamando image_page.dart.

Qui viene illustrata la foto e viene data la possibilità di aggiungere ulteriori tag ad essa.

Inoltre, grazie ad uno snapshot della collezione *immagini* in Firestore, riusciamo a recuperare l'insieme dei tag associati alla foto, e andarli a rappresentare utilizzando dei *RawChip*, con la possibilità sia di cliccare su una delle etichette e quindi effettuare una nuova ricerca, oppure andare ad eliminarne qualcuna cliccando sulla x.

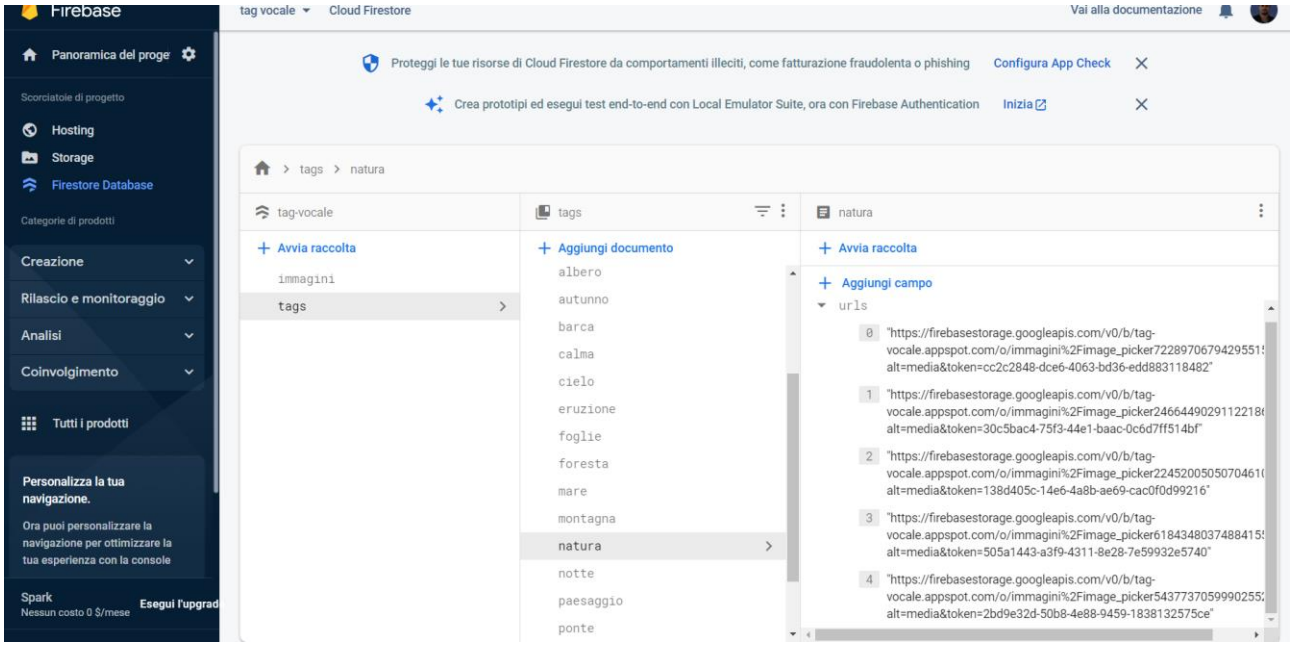


Se questa è la situazione su image_page, parallelamente avremo in Firestore questa rappresentazione:

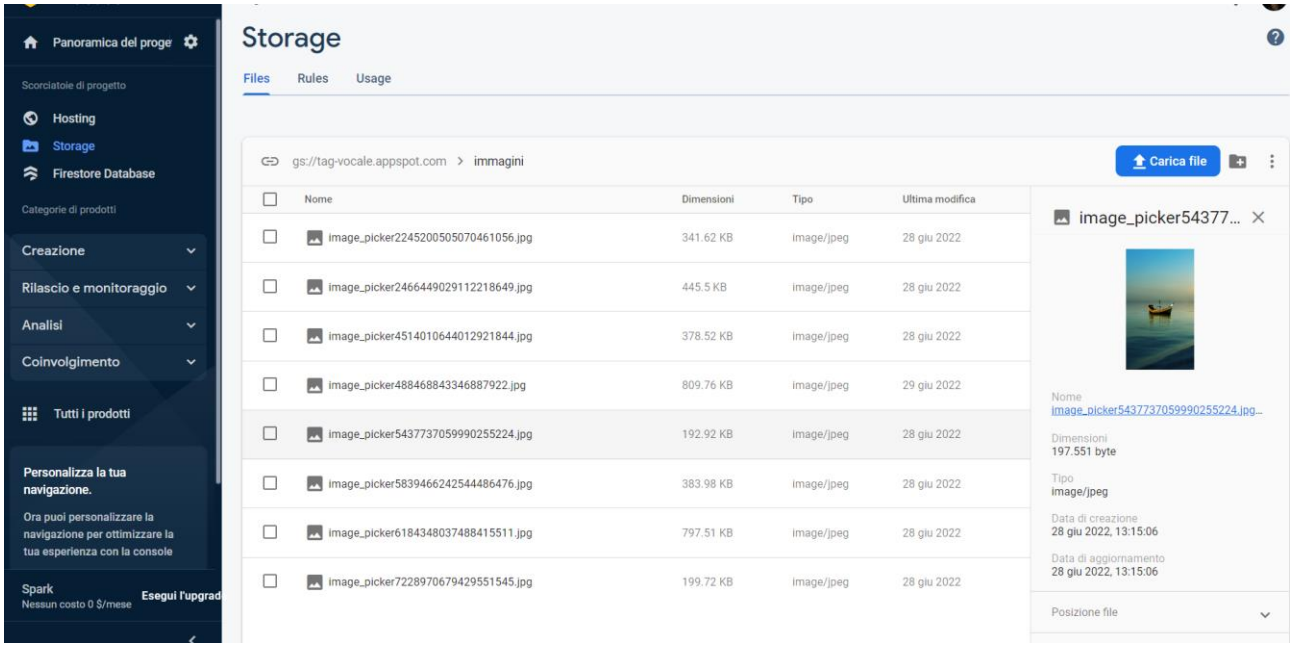


Nella raccolta *immagini* abbiamo un documento per ogni *nome_immagine.jpg,* a cui è associata la lista dei tags.

Discorso duale per la raccolta *tags*, in cui abbiamo un documento per ogni tag a cui è associata la lista degli urls delle immagini nello Storage.



Nell'immagine sottostante uno screenshot della situazione nello Storage, in cui nella cartella immagini vengono memorizzate le foto a cui associamo i tag.



A seguire il codice che carica l'immagine nello storage e crea un FirebaseFile, classe di supporto creata per memorizzare Reference, name ed url dell'immagine, prima di passarla ad ImagePage.

```
final pth= imm.path;
final fileName= Path.basename(imm.path);

File file = File(pth);

final firebase_storage.FirebaseStorage strg =
    firebase_storage.FirebaseStorage.instance;

//loading circle
showDialog(context: context, builder: (context) {
  return Center(child: CircularProgressIndicator());
},);

//CARICO FILE IN STORAGE
var ref= await  strg.ref('immagini/$fileName').putFile(file);
String imgUrl= await strg.ref('immagini/$fileName').getDownloadURL();


//CREO FIREBASEFILE
FirebaseFile ff= FirebaseFile(ref: strg.ref(imgUrl), name: fileName, url: imgUrl);

//rimuovo circle
Navigator.of(context).pop();

Navigator.of(this.context).push(MaterialPageRoute(
  builder: (context) => ImagePage(file: ff,tag:' '),
));  // MaterialPageRoute
```

A seguire il codice in ImagePage che permette di inserire in Firebase la parola pronunciata al rilascio del bottone “Aggiungi tag”:

```
DocumentReference docRef = FirebaseFirestore.instance.collection('immagini').doc(imgname);

onTapUp: (_) async { //onLongPressEnd
  text_visual=' ';
  DocumentSnapshot doc = await docRef.get();
  _stoplisten();
  if(text=='') return;

  DocumentReference docReftag = FirebaseFirestore.instance.collection('tags').doc(text);
  DocumentSnapshot docTag = await docReftag.get();//onLongPressEnd

  //METTO URL UNDER TAG IN FIREBASE
  String imgUrl= widget.file.url;
  if(!docTag.exists) { await FirebaseFirestore.instance.collection('tags').doc(text)
    .set({ 'urls' : [imgUrl]}); } else{
  docReftag.update({
    'urls' : FieldValue.arrayUnion([imgUrl]),
          });}

  //METTO TAG UNDER URL
  if(!doc.exists) { await FirebaseFirestore.instance.collection('immagini').doc(imgname)
      .set({ 'tags' : [text]}); } else{
  docRef.update({
    'tags' : FieldValue.arrayUnion([text]),
  });}

  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
    content: Text('Tag Aggiunto'),
    ));  // SnackBar
```

Infine cliccando su elimina dal singolo RawChip per rimuoverlo:

```
onDeleted: () async {
  text=tags[index];
  String x= tags[index];
  DocumentSnapshot doc = await docRef.get();
  docRef.update({
    'tags': FieldValue.arrayRemove([x]),
  });

  //AGGIORNO COLLECTION TAGS
  DocumentReference docRefTag = FirebaseFirestore.instance.collection('tags').doc(tags[index]);
  DocumentSnapshot docTag = await docRefTag.get();//onLongPressEnd
  String imgUrl= widget.file.url;
    docRefTag.update({
    'urls': FieldValue.arrayRemove([imgUrl]),
    });

  ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
      content: Text('Tag Rimosso'),
      )  // SnackBar
  );
```