Relazione Applicazioni Mobili 18-19

# Autori

Alexandru Stefan Andries 20018146

Nicolò Benella 20018152

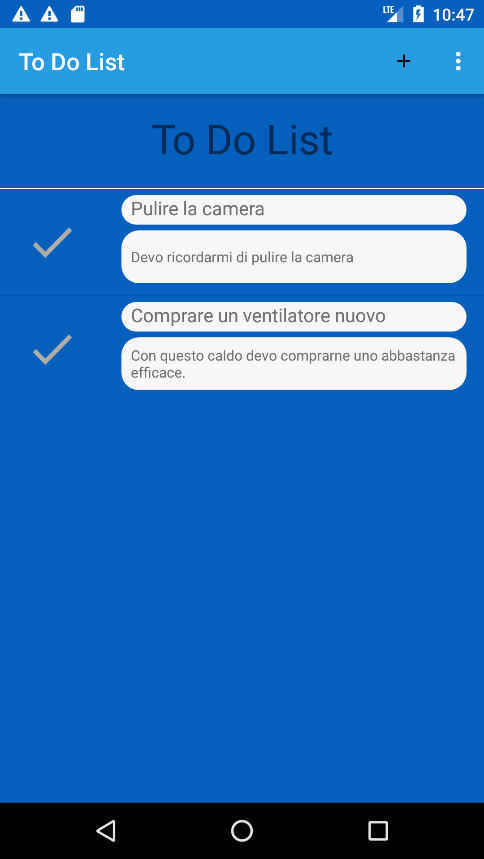
# Descrizione Progetto

Creazione di 4 app per dispositivi Android:

* [To do list](#_Progetto_1_:)
* [Book Search](#_Progetto_2_:)
* [Guida Museo](#_Progetto_3_:_1)
* [UPO Car Sharing](#_Progetto_4_:)

# 

# Progetto 1 : To do list



SSH : git@gitlab.di.unipmn.it:rastercrow/to-do-list.git

L’idea principale dell’applicazione to do list è quella di avere una app che ci permetta di aggiungere delle note che fungano da memo.

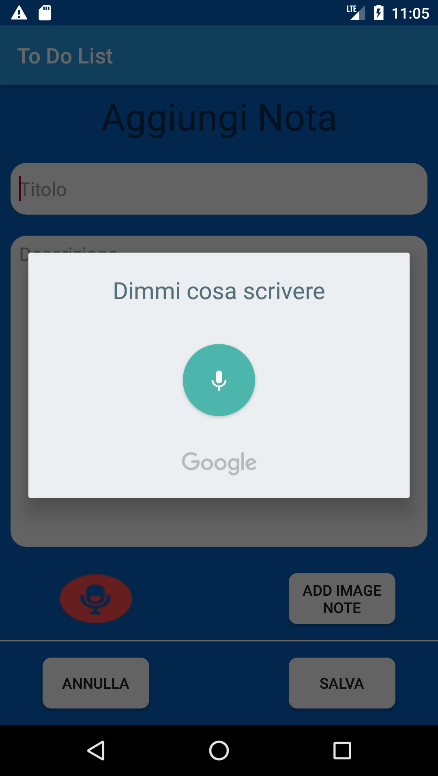
Queste note possono essere anche aggiunte via text 2 speech.

Una delle funzioni più importanti di questa app è la possibilità di salvare le note sia in locale che online.

Durante lo sviluppo, questa caratteristica ha portato a diversi problemi.

Abbiamo implementato l’extra per la visualizzazione via mappa.

## Funzionalità – Speech to text

Il text to speech nel nostro caso è stato implementato per aggiungere la descrizione essendo la parte più lunga da scrivere.

L’implementazione è stata anche molto semplice essendo una semplice funzione che avvia un intent predefinito.

Se c’è già del testo nella descrizione esso viene rimpiazzato con quello nuovo che viene captato.

Una caratteristica efficacie è l’uso di:

intent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, Locale.*getDefault*());

che permette a Google di capire la lingua da ascoltare in base a quello del telefono.

## Funzionalità – Location services

Pur non avendo svolto l-esercizio extra per la localizzazione delle note in base alla loro location al momento della creazione, abbiamo comunque implementato i locationservices di Google e durante la creazione della nota salviamo su firebase (anche se poi non viene mostrato nei dettagli della nota sull’app ) la location attuale del telefono.

Questo implica però che il telefono debba avere una versione aggiornata dei servizi Google, ma in caso contrario viene comunque mostrato un alert a schermo.

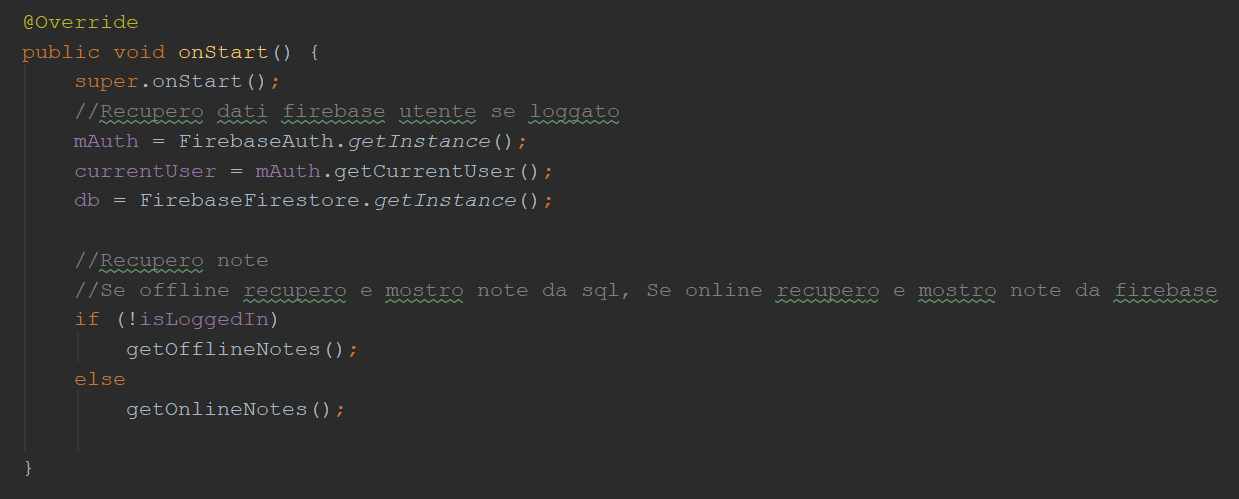
## Funzionalità – Cloud Save

Questa funzionalità, come già specificato è stata una delle funzionalità più difficili da gestire.

Un utente deve poter gestire delle note sia senza login, quindi offline sul telefono, sia dopo un login, quindi nel cloud.

Nella nostra implementazione se un’utente è loggato, è in grado di vedere solamente le note salvate su firebase mentre se non è loggato può vedere solamente le note salvate sul telefono in un database locale. La stessa cosa vale per la creazione delle note, se loggato le note che creo verranno salvate esclusivamente online.

La funzione del login è la standard per il login tramite e-mail su firebase; si svolgono una serie di controlli per controllare che password e e-mail siano scritte correttamente.

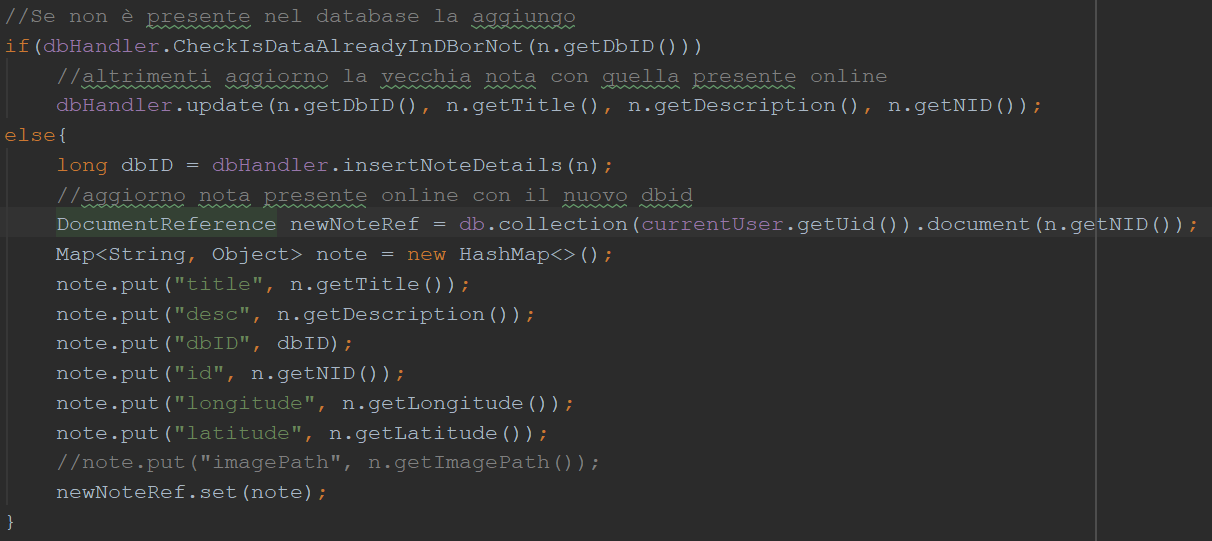


Una volta che l-utente ha fatto il login d’ora in avanti recupereremo esclusivamente le note che online. La parte complicata è stata il tenere conto dello stato attuale di entrambi i database e quindi evitare di recuperare note già presenti.

Ci è sembrato inoltre opportuno dare la possibilità all’utente di caricare le proprie note del telefono sul cloud e viceversa.

Per implementare ciò abbiamo dovuto tenere conto di entrambi gli id di una nota, quello che ha sul cloud e quello che ha localmente. Se durante il download delle note presenti online esiste una nota locale con lo stesso DbID (l’id nel database locale ), vuol dire che quella nota è già stata recuperata, quindi la aggiorna semplicemente con l’eventuale nuovo titolo/descrizione.

Se il DbID non è presente vuol dire che quella nota non è mai stata salvata sul telefono, quindi viene aggiunta al database locale. Inoltre, è necessario aggiornare anche la nota online con il nuovo id nel database locale, cosicché la prossima volta venga fatto solo un eventuale aggiornamento del testo.



Il caricamento delle note sul cloud è ma invertiamo i controlli, anziché controllare il DbID controlliamo il NID (note id su firebase ) e svolgiamo le stesse operazioni, se ha già una NID lo aggiorno, altrimenti aggiungo la nota come nuova nota.

Effettuare queste operazioni ci ha portato a creare diverse query per il db locale, come CheckIsDataAlreadyInDBorNot oppure funzioni di recupero / aggiornamento note.

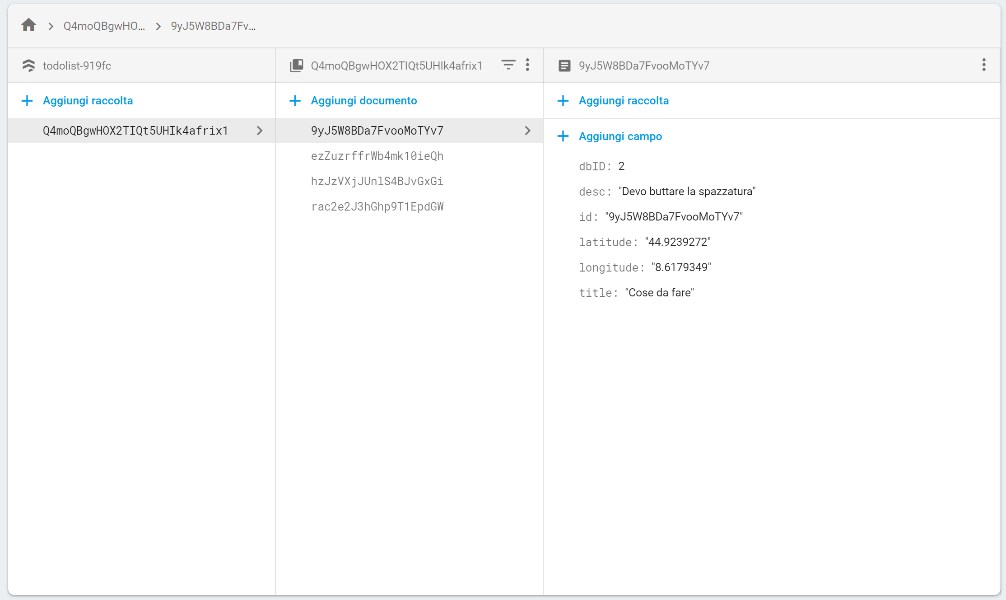
## EXTRA – Map View

Abbiamo implementato come richiesto una visuale dalla mappa delle note da noi inserite. Inoltre la mappa viene caricata in modo tale da riuscire a vedere la posizione di tutte le note contemporaneamente.

Il marker viene aggiunto nella location salvata nella nota che a sua volta è stata salvata durante la sua creazione.

Al click sul marker non viene visualizzata la nota, ma solamente il suo titolo.

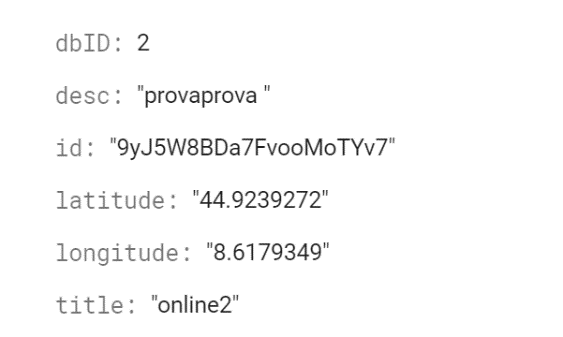
## Uno sguardo a Firebase



Per ogni utente abbiamo una collection identificata dal proprio id e ogni volta che creiamo una nota aggiungiamo un nuovo documento.

Come gia descritto in precedenza, si può notare l’utilizzo dei due id.

## Considerazioni finali

Non siamo totalmente sicuri la nostra implementazione sia la più corretta, semplicemente perché abbiamo gestito due database contemporaneamente e su entrambi i db teniamo traccia dell’id della nota sull’altro database.

Nel caso degli extra invece abbiamo cercato di implementare l’uso delle immagini, ma gestire il loro salvataggio su firebase ci stava complicando troppo il lavoro. Stessa cosa nel caso del database offline.

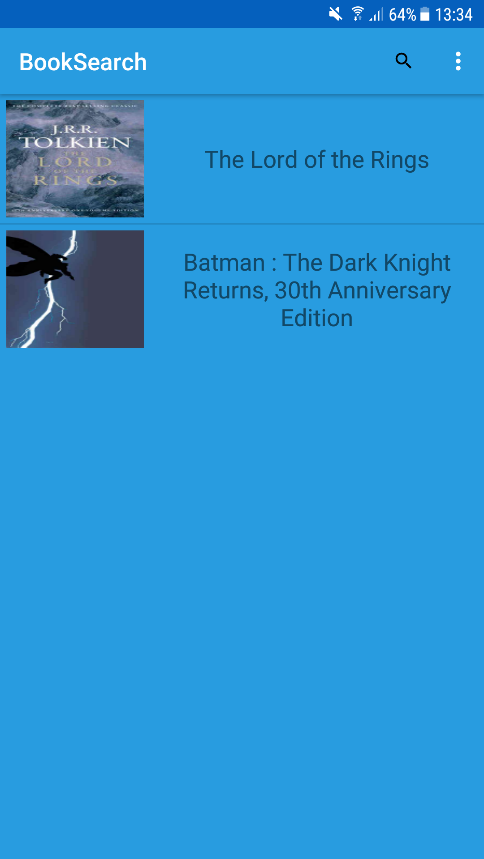
Siamo comunque soddisfatti del lavoro svolto e delle funzionalità dell’app.

Alcune idee che non siamo riusciti/abbiamo avuto tempo di implementare:

* Checklist per le note in modo da gestire note già eseguite.
* Aggiunta di una data per la nota con successiva notifica per ricordarla.
* Barra di caricamento per la lista di note.

# 

# Progetto 2 : Book Search

SSH : git@gitlab.di.unipmn.it:rastercrow/booksearchupo.git

L’applicazione Book Search permette a una persona di recuperare informazioni su di un libro tramite il suo codice isbn. Il codice isbn si può inserire sia tramite tastiera che tramite fotocamera, scannerizzando il codice a barre.

Questa applicazione richiede un account per poter essere utilizzata e di conseguenza una connessione a internet, in quanto è necessario salvare i dati dei libri cercati su firebase. Inoltre, richiede anche l’applicazione barcode scanner, scaricabile gratuitamente dal Google Play Store. Se non è presente sul telefono l’applicazione manderà un alert per farla scaricare.

Anche in questo progetto abbiamo implementato l’extra della map view, anche se non esattamente come richiesto.

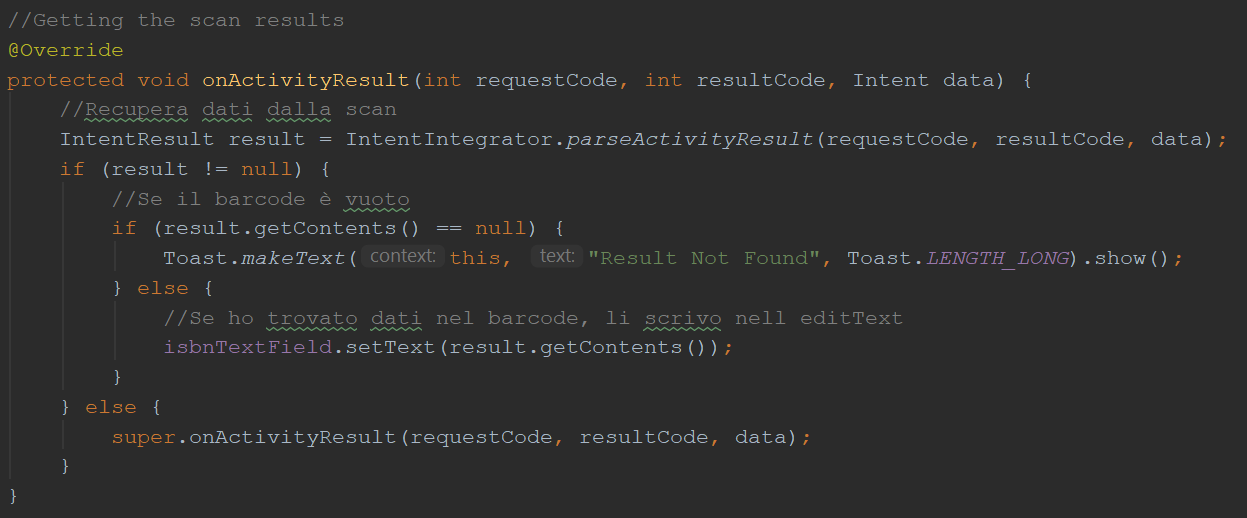
Inoltre per caricare l’immagine dal link specificato ci siamo appoggiato a una libreria esterna chiamata Picasso.

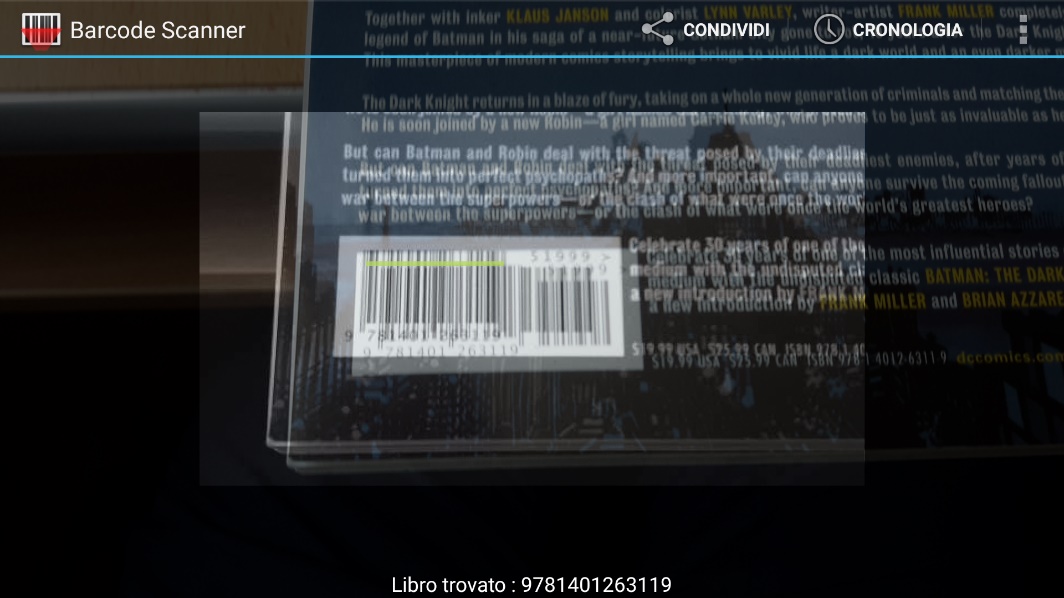
## Funzionalità – Barcode scanner

Per usare il barcode scanner abbiamo usato la libreria messaci a disposizione. Quando viene premuto il pulsante per usare la fotocamera viene inizializzata la scan :

barScan.initiateScan(IntentIntegrator.*PRODUCT\_CODE\_TYPES*)

e in seguito recuperiamo i dati trovati con :



E inseriamo il codice trovato direttamente nella edit text. Se non troviamo dati viene mandato un result not found.

## Funzionalità – Location services

Anche in questo caso, pur non avendo svolto l’esercizio extra per la localizzazione dei libri sulla mappa, abbiamo comunque implementato i location services di Google e salviamo la location nel momento della ricerca

Anche in questo caso, questo implica però che il telefono debba avere una versione aggiornata dei servizi Google, ma in caso contrario viene comunque mostrato un alert a schermo.

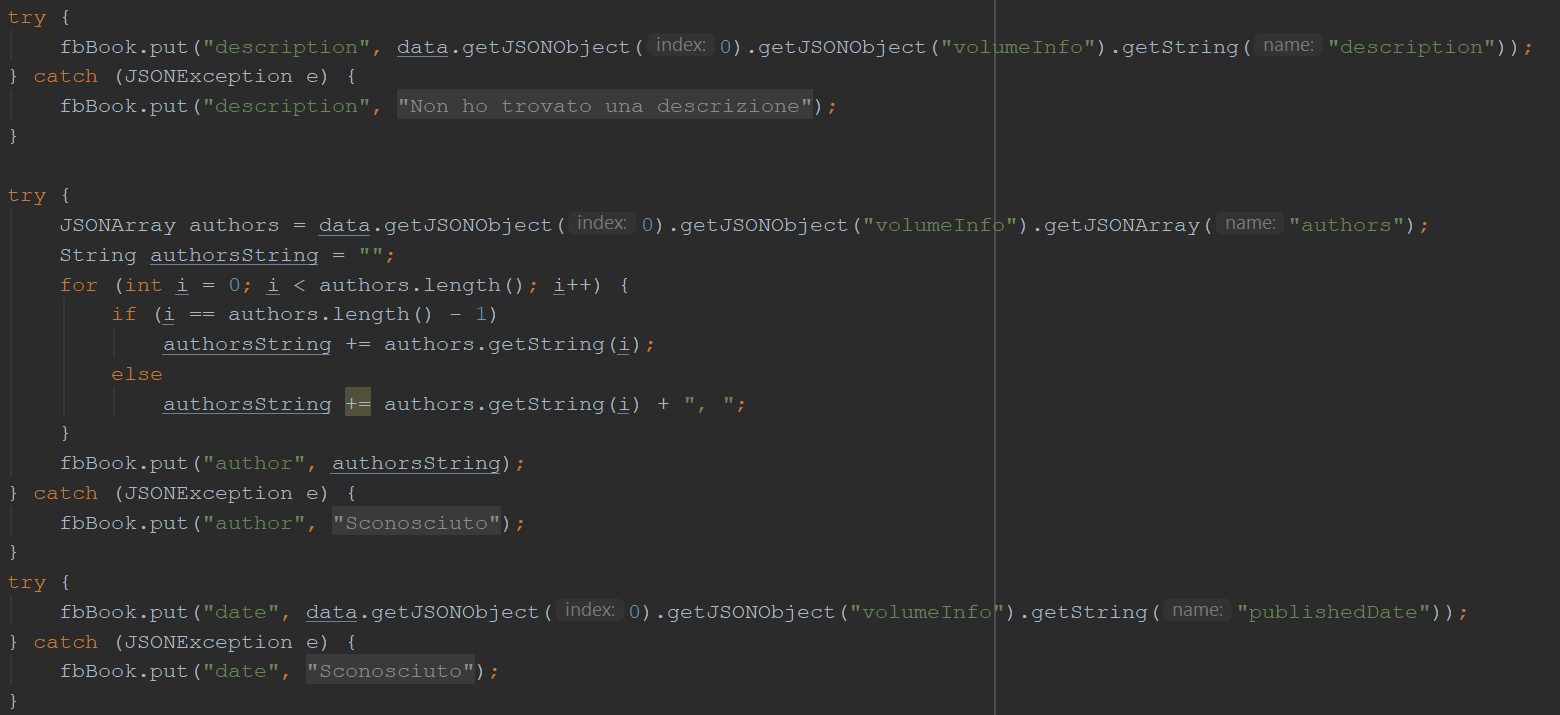
## Funzionalità – Google book services

Per eseguire l’effettiva ricerca dei dati del libro, abbiamo creato una task asincrona per recuperare il json presente al link :

<https://www.googleapis.com/books/v1/volumes?q=isbn>: + isbn

Essendo asincrona abbiamo lavorato sulla funzione doInBackground, che si occupa di recuperare il json facendo tutti i controlli necessari, e sulla onPostExecute che viene gestita non appena i dati sono stati recuperati.

Quest’ ultima è quella più complessa, in quanto una volta recuperato il json, abbiamo dovuto recuperare tutti i dati uno per uno. Per ogni dato è stato necessario inserire una try catch, in quanto non tutti i libri avevano a disposizione quelle informazioni.



1 - Esempio di recupero dati dal json

Dopo aver recuperato i dati abbiamo inserito il tutto su un documento firebase. Ogni utente avrà la propria collezione di libri, con tutti i libri cercati fino ad ora.

Siccome le immagini vengono passate come link web, ci siamo appoggiati a una libreria chiamata Picasso per caricarle velocemente.

## EXTRA – Map View

Anche in questo progetto abbiamo implementato come richiesto una visualizzazione dalla mappa dei libri. Tuttavia non funziona esattamente come richiesto in quanto non carica tutti i libri, compresi quelli di altri utenti, ma solamente i propri. Questo perche la nostra implementazione avrebbe portato a tempi e funzioni troppo lunghe per caricare tutti i libri.

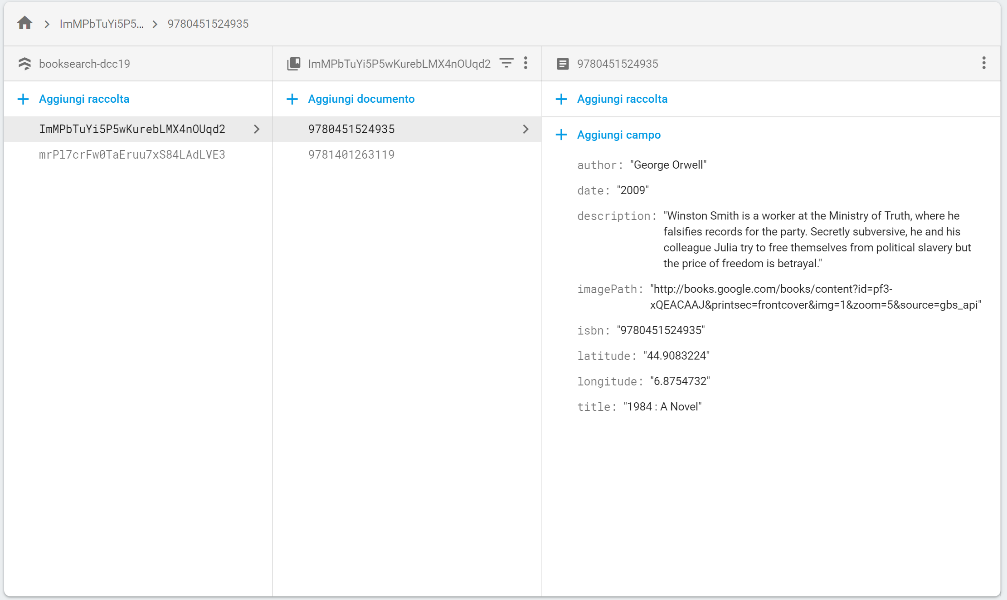
Nel nostro caso su firebase abbiamo una collection per ogni utente con tutti i suoi libri. Avremmo dovuto per ogni collection accedere e recuperare i libri, cosi non possibile in quanto gli id delle collection sono equivalente agli id utenti e solo il proprietario può esserne a conoscenza.

Avremmo potuto implementare una seconda collection con tutti i libri (come vedremo più avanti nel caso del progetto 4), ma abbiamo preferito concentrarci prima su altro.

Il marker viene aggiunto nella location salvata nel libro che a sua volta è stata salvata durante la sua creazione.

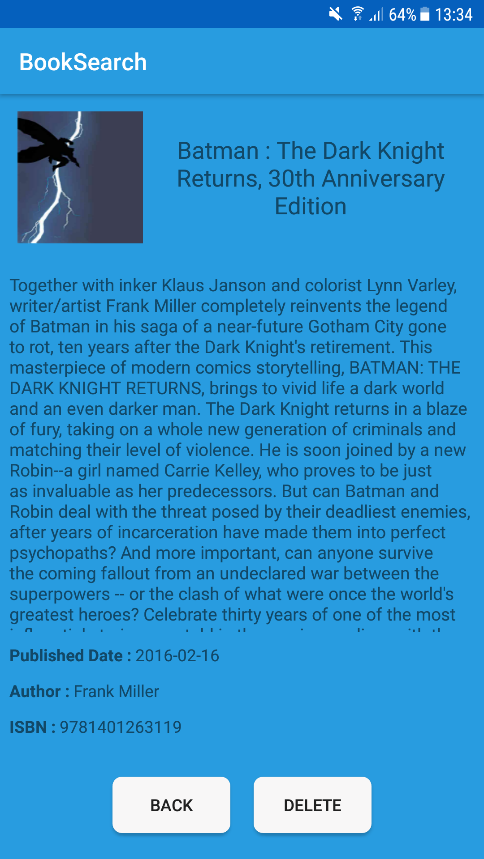
Al click sul marker viene attualmente visualizzato solo il titolo del libro.

## Uno sguardo a Firebase



La struttura del database di Firebase è molto semplice; abbiamo una collection per ogni utente, identificata dal proprio id e ogni volta che un libro viene cercato aggiungiamo un nuovo documento identificato dal isbn di tale libro, con all’interno i dati recuperati.

## Considerazioni finali



Siamo soddisfatti del risultato finale e l’idea dell’app è molto interessante.

Non siamo particolarmente convinti del fatto che bisogna scaricare l’app per il barcode scanner, ma non siamo riusciti a integrarla completamente nel progetto.

Anche in questo caso non mostriamo la location nelle informazioni, ma con una semplice modifica possiamo mostrare sia la location che altre informazioni recuperabili dal json, come publisher, ecc...

Alcune idee che non siamo riusciti/abbiamo avuto tempo di implementare :

* Caricare i libri in ordine di aggiunta ( vengono caricati in ordine da firebase e l’ordine di firebase è alfabetico in base all’id ).

# Progetto 3 : Guida Museo

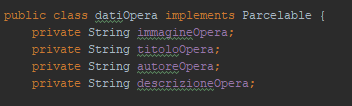
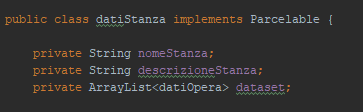
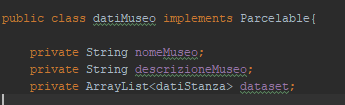
SSH : git@gitlab.di.unipmn.it:nicbene/app\_museo.git

L’applicazione Guida Museo permette a una persona di creare e personalizzare musei inserendo al loro interno, stanze e opere. Ogni museo ed ogni stanza possiede un nome ed una descrizione, mentre per le opere si va più nel dettaglio aggiungendo il nome dell’autore dell’opera ed una foto dell’opera che può essere caricata dalla memoria del telefono.

L’applicazione richiede permessi di accesso alla memoria in quanto ne necessita per poter salvare in locale i diversi musei creati ed il microfono per l’implementazione dello speech2text che abbiamo applicato soltanto alle descrizioni in quanto esse sono le parti più lunghe. L’implementazione del text2speech invece è applicata a tutti i campi del museo in modo da renderla una vera audioguida che supporta 2 lingue, inglese ed italiano.

Anche in questo per caricare l’immagine durante la visualizzazione viene usato Picasso per semplicità.

## Funzionalità – Crea Museo/Stanza/Opera

La funzionalità della creazione è appunto quella di creare e personalizzare un museo inserendo al suo interno stanze ed all’interno di ogni stanza una o più opere. 

Ogni museo è un oggetto personalizzato contenete stringhe per il nome e la descrizione del museo ed un arraylist di elementi che saranno le nostre stanze. Stesso ragionamento viene effettuato con le stanze, ogni stanza è un oggetto contenente stringhe ed un arraylist per le opere. Le opere implementano oltre a nome e autore anche una stringa per l’autore ed una stringa dove noi salviamo l’URI della nostra immagine dell’opera . Durante la creazione di un opera, stanza o museo è possibile utilizzare il microfono per scrivere la descrizione della nostra opera, stanza o museo.

## Funzionalità – Salva/Carica Museo

Questi 2 metodi ci permettono di salvare e caricare dal locale i nostri musei in quanto non abbiamo implementato un sistema di salvataggio online con firebase per via delle problematiche del salvataggio di un immagine su di esso ( come gia spiegato ). Questo sistema semplicemente converte il nostro arraylist di musei in stringa che poi andiamo a scrivere su di un file locale; in maniera inversa invece recuperiamo il file, convertiamo le stringe in un oggetto museo e lo rendiamo disponibile per le eventuali modifiche/visualizzazioni.

Funzionalità – Visualizza Museo/Stanza/Opera

La visualizzazione (qui mostrerò solamente quella di un’opera in quanto contiene anche foto e autore) permette la lettura di tutte le informazioni di cui ha bisogno e inoltre grazie ai pulsanti può eseguire la lettura tramite Text2Speech e fermarla quando vuole. Mentre si visualizza un’opera è possibile premere sulla foto e si accede su un activity che visualizza la foto a schermo intero.

## Considerazioni finali

Nonostante la complessità di un progetto del genere siamo soddisfatti del risultato finale; ideare infatti un metodo che permetta la creazione, con successiva gestione non è stato semplice, ma pensiamo che il nostro sistema sia funzionale e relativamente semplice da capire.

Alcune idee che non siamo riusciti/abbiamo avuto tempo di implementare :

* Come per l’extra delle immagini della to do list, anche in questo non siamo riusciti a gestire il tutto tramite firebase perché abbiamo avuto problemi con il salvataggio delle immagini sullo storage online.

Questo problema inoltre non ci ha permesso di gestire cose come la visualizzazione di altri musei di cui non siamo proprietari, in quanto sono tutti salvati sul nostro telefono.

# 

# Progetto 4 : UPO Car Sharing



SSH : git@gitlab.di.unipmn.it:rastercrow/upocarsharing.git

L’applicazione dell’ UPO Car Sharing permette a tutti gli iscritti dell’uniupo ( studenti e personale ) di condividere un viaggio in macchina tra loro, in modo da ottimizzare il trasporto, condividere spese o semplicemente dare una mano a un compagno.

Questa applicazione richiede un account dell’uniupo per poter essere utilizzata e di conseguenza una connessione a internet.

Non abbiamo implementato gli extra per questa applicazione.

Nonostante il corretto funzionamento di tutte le funzionalità implementate, c’è un problema globale che affligge l’app ovvero quello della sincronizzazione. La nostra implementazione infatti può portare a problemi quando due o piu persone si uniscono/disiscrivono da un viaggio. Non avendo implementato nessun sistema di gestione del parallelismo, le info del viaggio vengono recuperate inizialmente quando vengono mostrate le loro info, quindi se viene eseguita qualche altra operazione le view non vengono aggiornate in tempo reale e questo può portare a problemi.

E’ un problema che non siamo riuscite a risolvere.

## Spiegazione generale

Ogni utente può visualizzare tutti i viaggi creati dagli altri utenti.

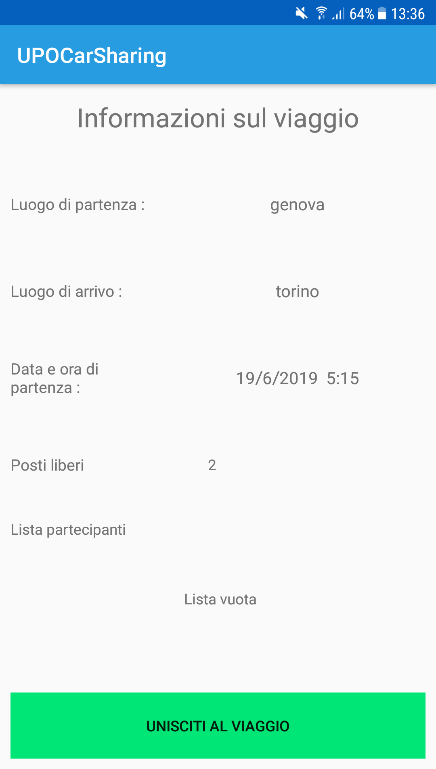
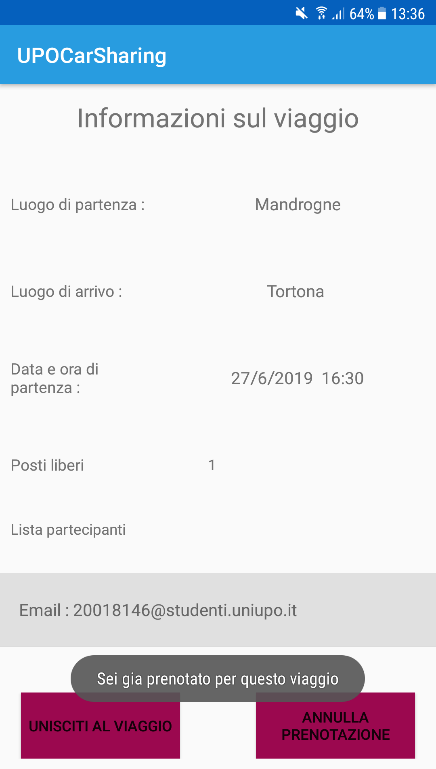
Ogni utente può creare un viaggio, e modificare il tale solamente se nessuno ha ancora partecipato al viaggio; se ci sono partecipanti il viaggio è definitivo.

Ogni persona può partecipare ai viaggi solamente se ci sono ancora posti liberi e se non sono già partecipanti a tale viaggio.

Non c’è gestione dei viaggi completati, essi rimangono e sono ancora joinabili dagli utenti.

## Scelta implementativa – TripInfo Mode

Per evitare troppe ripetizioni di codice abbiamo deciso di implementare 3 modalità principali con le quali visualizzare i dettagli e gestire i viaggi.



3 - Viaggio gia prenotato 4 - Viaggio con posto libero 5 - Viaggio di cui sono proprietario

Effettivamente nel codice sono solo due, TripInfoNormalMode e TripInfoOwnerMode .

Praticamente ogni volta che clicchiamo su un viaggio, viene fatto un controllo, se sono l’owner di tale viaggio, apro in modalità owner, altrimenti modalità normale.

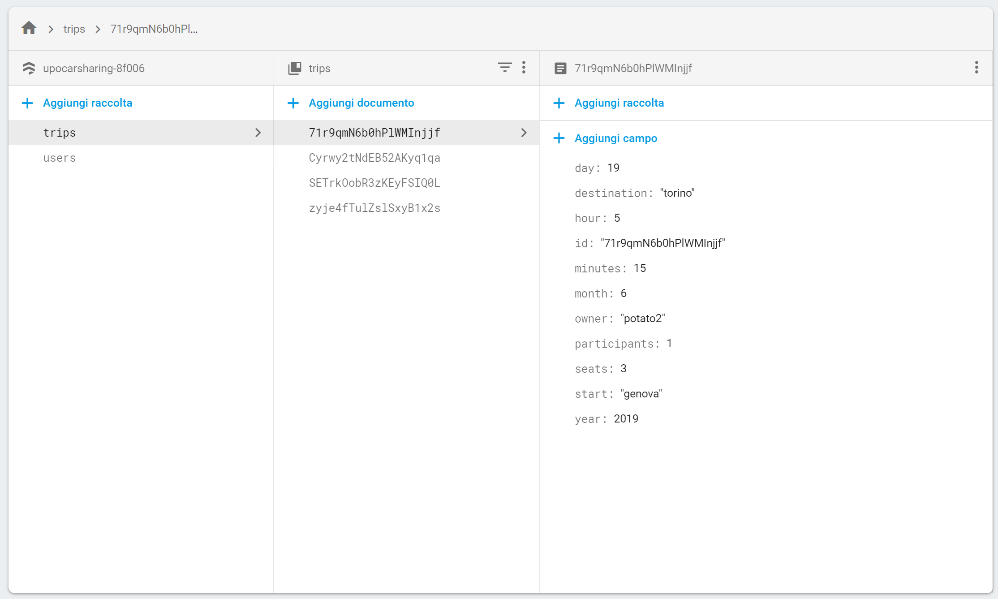
In seguito all’interno di queste classi viene fatto un controllo sul numero di partecipanti/controllo se sono già partecipante e in base ai risultati decide se permettere o meno la modifica del viaggio, nel caso di owner mode, oppure la partecipazione a tale viaggio, nel caso di normal mode.

Alcune modifiche sono leggermente più complesse, ad esempio per gestire i viaggi a cui partecipo abbiamo dovuto creare una seconda raccolta su firebase per ogni utente, con gli id dei viaggi a cui si partecipa. Questa implementazione ci è sembrata la più efficacie.

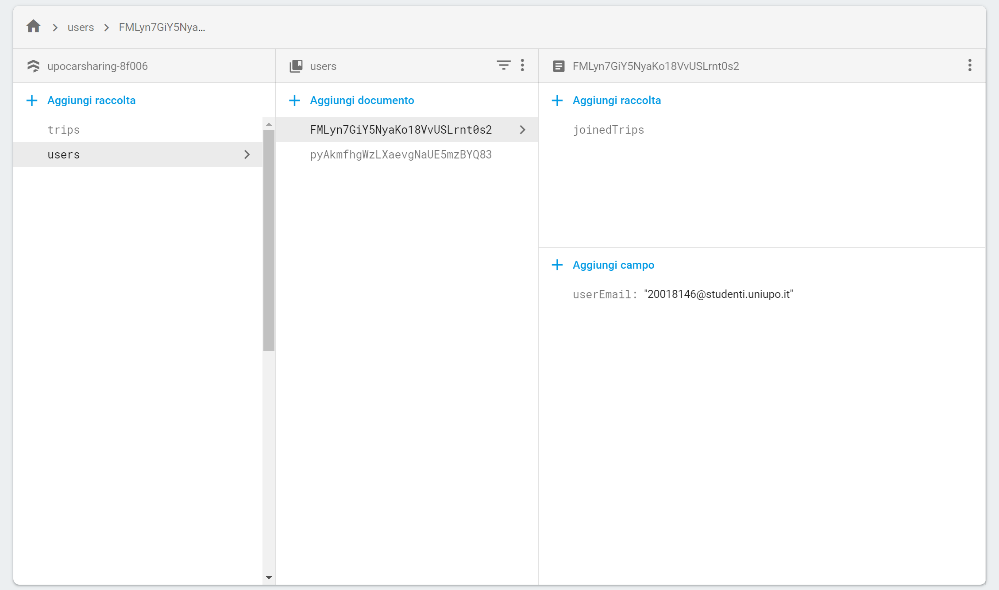
Inizialmente avevamo provato ad aggiungere all’interno di ciascun documento relativo al viaggio, una collection con tutti gli utenti partecipianti, ma senza successo.

Questo è il metodo con il quale visualizziamo i dettagli di un viaggio, ma i viaggi effettivi vengono visualizzati in una lista, che cambia in base alla bottone premuto nel menu inziale. La base della lista è lo stesso layout, ma è stato necessario implementare 3 classi diverse per gestire al meglio i controlli da effettuare e il caricamento della lista.

## Uno sguardo a Firebase



In questo caso la struttura del database è leggermente diversa dalle precedenti. Abbiamo due collezioni; siccome I viaggi devono essere visti da tutti abbiamo deciso di includere in trips tutti i viaggi creati, con id random. I proprietari possiamo gestirlo come campo nel documento.



In users invece abbiamo per ogni utente una seconda raccolta chiamata joinedTrips con all’interno documenti che identificano i viaggi a cui partecipano.

Può sembrare complesso, ma in questo caso non dovremmo mai gestire due collezioni diverse nel caso di eliminazione o modifica. Questo perche un viaggio non può essere ne modificato ne eliminato se un utente è gia prenotato.

Questa implementazione sarebbe stata efficacie anche nel caso del [progetto 2](#_Progetto_2_:_1).

## Considerazioni finali

Nonostante il problema della sincronizzazione, molto grave per il corretto funzionamento dell’app, siamo soddisfatti del modo in cui abbiamo gestito le funzionalità dell’app.

Siamo anche particolarmente soddisfatti della nostra implementazione sui partecipanti e la lista visualizzata in ciascun viaggio.

Alcune idee che non siamo riusciti/abbiamo avuto tempo di implementare:

* Nel codice è ancora presente, abbiamo provato per l’inserimento della città o via di partenza a implementare il servizio di google places, in modo da effettivamente selezionare una città esistente. Per qualche motivo nonostante la corretta implementazione, il servizio google non funzionava e per qualsiasi città o comunque lettera iniziale che inserivamo non trovava nessun risultato. Dopo esserci informati su internet abbiamo trovato altri utenti con lo stesso problema, ma nessuna soluzione. Sarebbe stata una funzionalità molto utile per questo tipo di applicazione.
* Come gia spiegato abbiamo provato a risolvere il problema della sincronizzazione, ma non siamo riusciti a trovare nessun modo.
* Anche in questo caso ci sarebbe piaciuto implementare una barra di caricamento durante il recupero della lista dei viaggi.

# Considerazioni finali sui progetti

I progetti sono stati abbastanza complessi, sia per le funzionalità richieste che per la quantità di lavoro da svolgere.

Riguardo agli extra, siamo abbastanza delusi di non essere stati in grado di gestire il salvataggio delle immagini su firebase. In locale non è stato complicato, ma i problemi arrivavamo quando dovevamo gestire il salvataggio in cloud. Inizialmente pensavamo di salvare il path di un’immagine, anziche l’immagine stessa, ma questo avrebbe portato a problemi nel caso di condivisione di dati, o nel caso un’immagine venga rimossa dal telefono. Sappiamo che firebase offre un sistema di storage, quindi pensavamo di poter salvare le immagini nel cloud, ma sfortunatamente non siamo riusciti a gestire la cosa e abbiamo preferito concentrarci su altro.

Tuttavia dopo averli svolto questi progetti ci sentiamo molto più a nostro agio con questo ambiente di lavoro e anche più motivati a sviluppare e sperimentare con le applicazioni mobili per Android ed eventualmente iOS.