

Gabriele Zotta - 886101

Alessandro Condello - 887918

Carlo Luca Comotti - 885885

Yanhui Matteo Hu - 879354

# NotInsta - Documentazione



# Sommario

<b>Funzionalità</b>	2
<b>Architettura del Software</b>	4
UI Layer	.
View Models	.
Data Layer	5
Data Source	.
Utils	.
<b>API Utilizzate</b>	6
<b>Activities</b>	7
LoginActivity	.
SignupActivity	.
HomeActivity	.
CameraActivity	8
<b>Fragments</b>	9
Home	.
Settings	10
Camera	11
<b>Possibili Sviluppi</b>	12
Monetizzazione	.
Privacy	.

## Funzionalità



Welcome to !Insta!

### ► Login

Utilizziamo Firebase per effettuare il login e la registrazione dell'utente. Una volta autenticato, l'utente potrà accedere ai servizi dell'applicazione.

Per la registrazione sono richiesti email e password.

### Visualizzazione Profilo ◀

E' possibile visualizzare i profili degli altri utenti, in questa visualizzazione è possibile vedere l'username, la descrizione e l'immagine profilo

## ► Creazione Post

Il core dell'applicazione risiede nella sua componente di social network: l'utente può creare un proprio post caricando o scattando un'immagine dal proprio dispositivo.

Può anche applicare alcuni filtri per modificare l'immagine, scrivere una descrizione e scegliere tag appropriati, che saranno poi utili per la ricerca dei post di altri utenti.

Esiste anche un'opzione per l'eventuale sponsorizzazione del post, che in futuro potrebbe essere usata per la monetizzazione.

La creazione dei post può avvenire anche offline, poiché il post viene creato in locale e poi, alla prima occasione (ossia, appena viene attivata la connessione), viene caricato in remoto.

Va da sé che la pubblicazione avviene solamente in presenza di una connessione. Visualizzazione Post ◀

Dalla home si possono visualizzare i post di altri utenti dell'applicazione, ordinati in maniera tale che i post più recenti siano visualizzati prima degli altri. Organizzata come una galleria di immagini.

## ► Ricerca

E' possibile ricercare utenti e tag dalla seconda schermata nella home. Cercando #tag ci si ritroverà nella home e verranno visualizzati i post con quel tag. Mentre cercando un qualsiasi nome, si cercherà l'utente con l'username uguale a quello inserito.

## Impostazioni ◀

All'interno delle impostazioni dell'applicazione è possibile personalizzare vari aspetti del proprio profilo e delle preferenze dell'applicazione. In particolare è possibile:

- Scegliere la Lingua: l'applicazione offre supporto per l'italiano e l'inglese.
- Cambiare il nome utente: si può facilmente aggiornare il proprio username.
- Aggiornare la Password: la sicurezza dell'account è fondamentale. Si può modificare la password in qualsiasi momento dalle impostazioni.
- Cambiare l'immagine del profilo: si può modificare l'immagine profilo.

# Architettura del Software

## UI Layer

Lo UI Layer contiene tutte le Activity e i Fragment, che gestiscono la parte grafica dell'applicazione.

E' composto da varie schermate:

- Login
- Main (dove vengono visualizzati i post)
- Creazione di post
  - Acquisizione / Scelta dell'immagine
  - Applicazione dei filtri
  - Impostazione di descrizione, tag e eventuale sponsorizzazione
- Profilo
- Opzioni

## View Models

Servono per gestire la permanenza dei dati e per effettuare operazioni su di essi, interfacciandosi con lo strato UI e il database. Sono divisi in tre aree:

- Posts
- Settings
- Users

## Posts

La classe *ProcessedImageViewModel* si occupa di preparare la bitmap da visualizzare nelle schermate di creazione del post, e di gestire le informazioni ad essa correlate (filtri applicati, tag e descrizione). L'applicazione dei filtri viene fatta tramite le *Utils* associate (*FilterUtils*), mentre la pubblicazione del post viene affidata alla *PostManager*.

*PostsViewModel* è la classe che si occupa di mantenere lo stato della *HomeActivity*, salvando tre diversi *LiveData* per i post (uno per fragment) e tutte le informazioni accessorie. Inoltre fa da tramite tra la *HomeActivity* e il *PostManager* per il recupero dei post da visualizzare. Il caricamento delle immagini è affidato a *Glide*.

## Data Layer

Sono presenti due principali categorie di repository:

- *UserRepository* che gestisce gli utenti, più precisamente la parte dell'autenticazione e della registrazione, facendo attenzione ai vari casi di successo e fallimento.
- *PostManager*, che gestisce la totalità dei post. Dato che i post sono composti da due tipi di dati (immagini + dati strutturati) abbiamo deciso di creare due distinte repository: *PostDataRepository* e *PostImageRepository*, ognuno dei quali gestisce una risorsa locale e una risorsa da remoto. Per la sincronizzazione si affida alle funzionalità di *PostWorkerSource*, che richiama i *Workers* adeguati.

## Data Source

### *PostDataRemoteSource*

Classe che ha il compito di interfacciarsi con la risorsa remota per il recupero dei dati strutturati. Utilizza la API offerta da Firebase Firestore.

### *PostDataLocalSource*

Stessi compiti della precedente, solo che la risorsa è locale. Utilizza Room.

### *PostImageRemoteSource*

Classe che ha il compito di interfacciarsi con la risorsa remota per il recupero delle immagini. Utilizza la API offerta da Firebase Firestore.

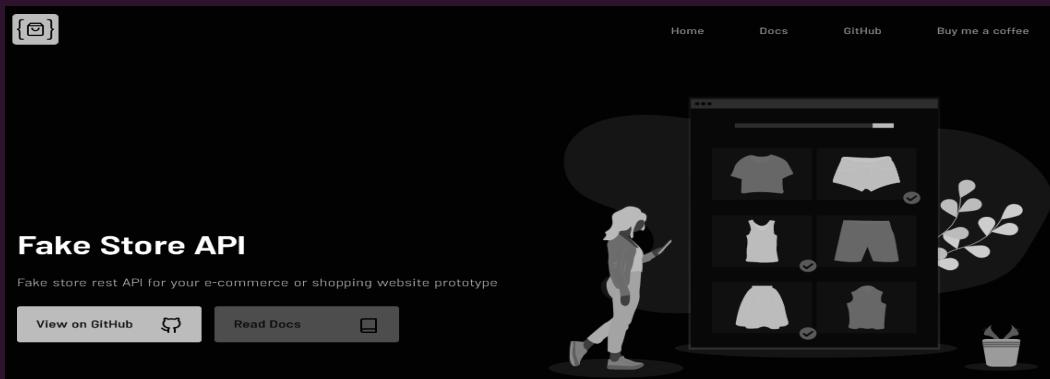
### *PostImageLocalSource*

Gestisce le immagini dei post all'interno della memoria di massa interna del dispositivo

## Utils

Qui abbiamo messo delle classi di utility per alcune operazioni, come l'applicazione dei filtri alla bitmap di un post.

# API Utilizzate - Fake Store API e Room



L'applicazione integra Fake Store API (<https://fakestoreapi.com/>) per generare post sponsorizzati da mostrare agli utenti. Questa REST API fornisce dati in formato JSON riguardanti prodotti pseudo-realistici, rendendo semplice e gratuito l'accesso a informazioni di prodotti per creare contenuti pubblicitari. Gli utenti visualizzano post sponsorizzati che renderanno l'esperienza di utilizzo il più simile possibile a quella dei veri social network.

## Gestione dei Dati Locali con Room

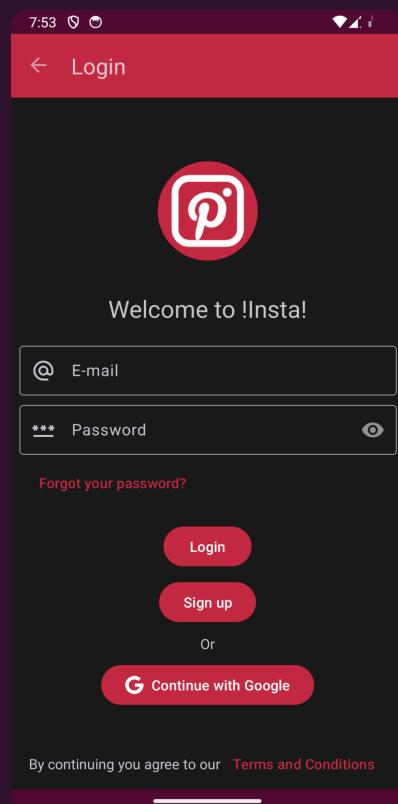
Per il salvataggio e la gestione dei dati strutturati in locale, l'applicazione utilizza Room, una libreria di persistenza di dati fornita da Android. Room permette di memorizzare dati strutturati in un database SQLite locale, offrendo un'interfaccia semplice e robusta per eseguire operazioni di creazione, lettura, aggiornamento e cancellazione (CRUD).

# Activities

## LoginActivity

Attività dedicata a mostrare la sezione per l'autenticazione. Qui l'utente avrà la possibilità di autenticarsi, registrarsi oppure recuperare la password in caso sia dimenticata. L'activity utilizza i seguenti fragment: LoginFragment, SignupFragment, PasswordResetFragment, TermsConditionsFragmet.

Inoltre, in fondo alla schermata sono disponibili i termini e le condizioni d'uso, ovviamente sono un "placeholder", essendo un progetto didattico non abbiamo approfondito ulteriormente sul lato legale.



## SettingsActivity

La SettingsActivity è un'attività che fornisce un'interfaccia utente per le impostazioni dell'applicazione, utilizzando NavController per la navigazione tra i frammenti e una Toolbar configurata come barra delle azioni.



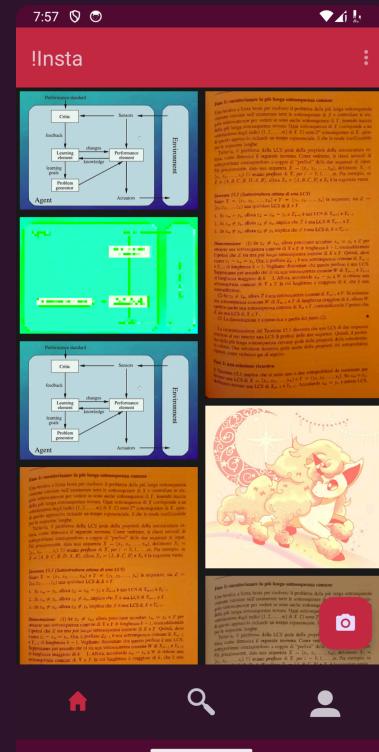
## ShowPostActivity

Activity che ha il solo scopo di presentare a schermo i dati di un post selezionato dall'utente.

## HomeActivity

È la prima *Activity* che un utente già loggato vede; contiene una barra di navigazione per i *Fragment* in esso contenuti. In ogni *Fragment* è possibile cliccare su un'immagine per avviare la *ShowPostActivity* per vedere le informazioni del post correlato. Utilizza:

- *StartingFragment*
- *ProfileFragment*
- *SearchFragment*
- *GenericGalleryFragment*, *fragment* base della galleria delle immagini, specializzata in tre diverse sottoclassi, una per ogni precedente *Fragment* e incluse all'interno di esse.



## CameraActivity

Questa Activity funge da contenitore per i Fragment relativi alla creazione di un post (*ImageChooserFragment*, *ImageViewerFragment* e *PostDescriptionFragment*).

Gestisce la navigazione tra i fragment e la visualizzazione della bitmap dell'immagine scelta, che viene tenuta in memoria come *MutableLiveData* nella classe *ProcessedImageViewModel*.

Visualizza inoltre le modifiche apportate alla bitmap durante l'applicazione dei filtri tramite un observer apposito.



# Fragments

## Home

---

### StartingFragment

*Fragment* di partenza. Ha il solo scopo di visualizzare tutti i post, ad esclusione dei propri. Da qui si può accedere alle impostazioni.



### ProfileFragment

Qui è possibile vedere le informazioni relative al proprio profilo e i propri post

### SearchFragment

*Fragment* dedicato alla ricerca dei post: è possibile cercare i post in base all'ID dell'autore oppure in base ad un tag; in quest'ultimo caso il tag deve partire con il carattere '#'. La ricerca inizia cliccando il pulsante di ricerca; una nuova ricerca cancella quella precedente

## Login

---

### LoginFragment

Il fragment permette all'utente di autenticarsi nel sistema tramite email e password, oppure utilizzando un account google.

## SignupFragment

Il fragment permette all'utente di registrarsi all'interno del sistema utilizzando una email ed una password (da ripetere). Le ulteriori informazioni dell'utente come il nome utente potranno venire assegnate dopo che l'utente si è autenticato nel sistema.

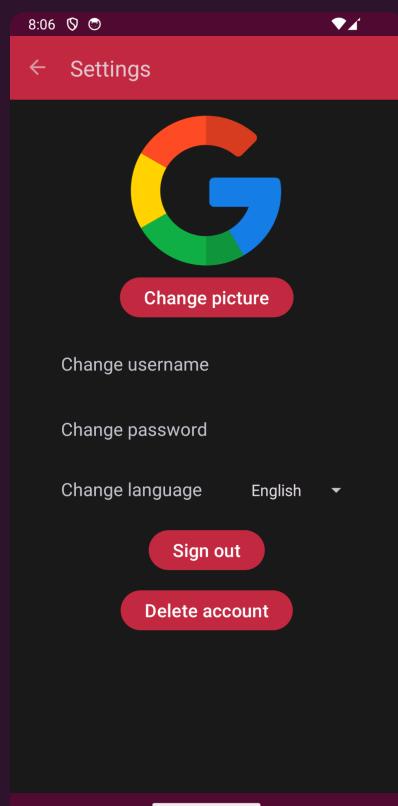
## PasswordResetFragment

Il fragment permette all'utente di recuperare la password dimenticata inserendo l'email dell'account. All'interno della posta elettronica ci sarà un link che porterà ad una pagina dedicata all'impostazione di una password nuova.

## TermsConditionsFragment

Il fragment permette all'utente di visualizzare i termini e le condizioni d'uso. Accedendo oppure registrandosi, l'utente accetta automaticamente. Ovviamente sono un "placeholder", essendo un progetto didattico non abbiamo approfondito ulteriormente il lato legale.

# Settings



## SettingsFragment

Questo fragment gestisce l'interfaccia utente per le impostazioni dell'applicazione, consentendo agli utenti di cambiare lingua, nome utente, password e immagine del profilo. In fase di creazione, carica le preferenze dell'utente e imposta lo spinner per la selezione della lingua. Fornisce anche funzionalità per la gestione della navigazione tra i vari frammenti di impostazioni e l'esecuzione di azioni come il logout e la cancellazione dell'account.

## ChangePasswordFragment

Frammento che permette agli utenti di cambiare la loro password. Verifica che la nuova password sia diversa dalla vecchia, che abbia una lunghezza minima e che contenga caratteri speciali.

## ChangeUsernameFragment

Frammento che consente agli utenti di modificare il loro nome utente. Verifica che il nuovo nome utente abbia una lunghezza minima di 5 caratteri.

# Camera

---

La cartella contiene i tre Fragment relativi alla creazione di un post, che vengono sostituiti nella parte inferiore dell'Activity *CameraActivity*.

## ImageChooserFragment

Il primo Fragment visualizzato permette di scegliere l'immagine da utilizzare per il nuovo post. L'utente può decidere di prenderne una dalla galleria, o scattare una nuova foto utilizzando la fotocamera standard del dispositivo.

La scelta dell'immagine chiama una callback in *CameraActivity* che aggiorna l'immagine visualizzata e invia la bitmap al View Model.

## ImageViewerFragment

Una volta confermata la scelta dell'immagine si passa a questo Fragment, dove l'utente può impostare dei filtri per modificarla. I parametri vengono passati al View Model che poi si occuperà di aggiornare la bitmap e applicare i filtri.

## PostDescriptionFragment

L'ultimo Fragment permette infine di aggiungere tag, descrizione ed eventuale sponsorizzazione del post.

E' possibile anche rimuovere i tag semplicemente cliccando su di essi.



# Possibili Sviluppi

## Monetizzazione

Come accennato in precedenza, si lascia spazio alla possibilità di permettere la monetizzazione dei post da parte degli utenti. Inoltre, si potrebbero aggiungere dei banner pubblicitari da inserire tra i post visualizzati nella schermata home.

## Privacy

Un'altra importante funzionalità potrebbe essere la possibilità di bloccare o seguire altri utenti, e di cambiare la visibilità del proprio profilo (pubblico / privato).