Documentazione progetto
Programmazione di dispositivi mobili
Anno accademico: 2021/2022

Meeple!

Gaia Colombo
Sandro Erba

FUNZIONALITÀ OFFERTE

L’applicazione ruota intorno all’ampio mondo dei giochi da tavolo, e permette di gestire un archivio nel quale si potranno cercare e salvare i giochi che più ci piacciono, ma soprattutto scoprirne di nuovi.

Per non perdere i dati relativi ai giochi salvati ogni utente deve avere un profilo, che gli permetterà di accedere anche da altri dispositivi.

Le tre schermate principali dell’applicazione sono: esplora, profilo e amici.

Nella pagina esplora troviamo vari giochi consigliati, divisi per categorie, età, tempo di gioco e altro ancora. È inoltre disponibile una barra di ricerca per dare la possibilità all’utente di cercare altri giochi.

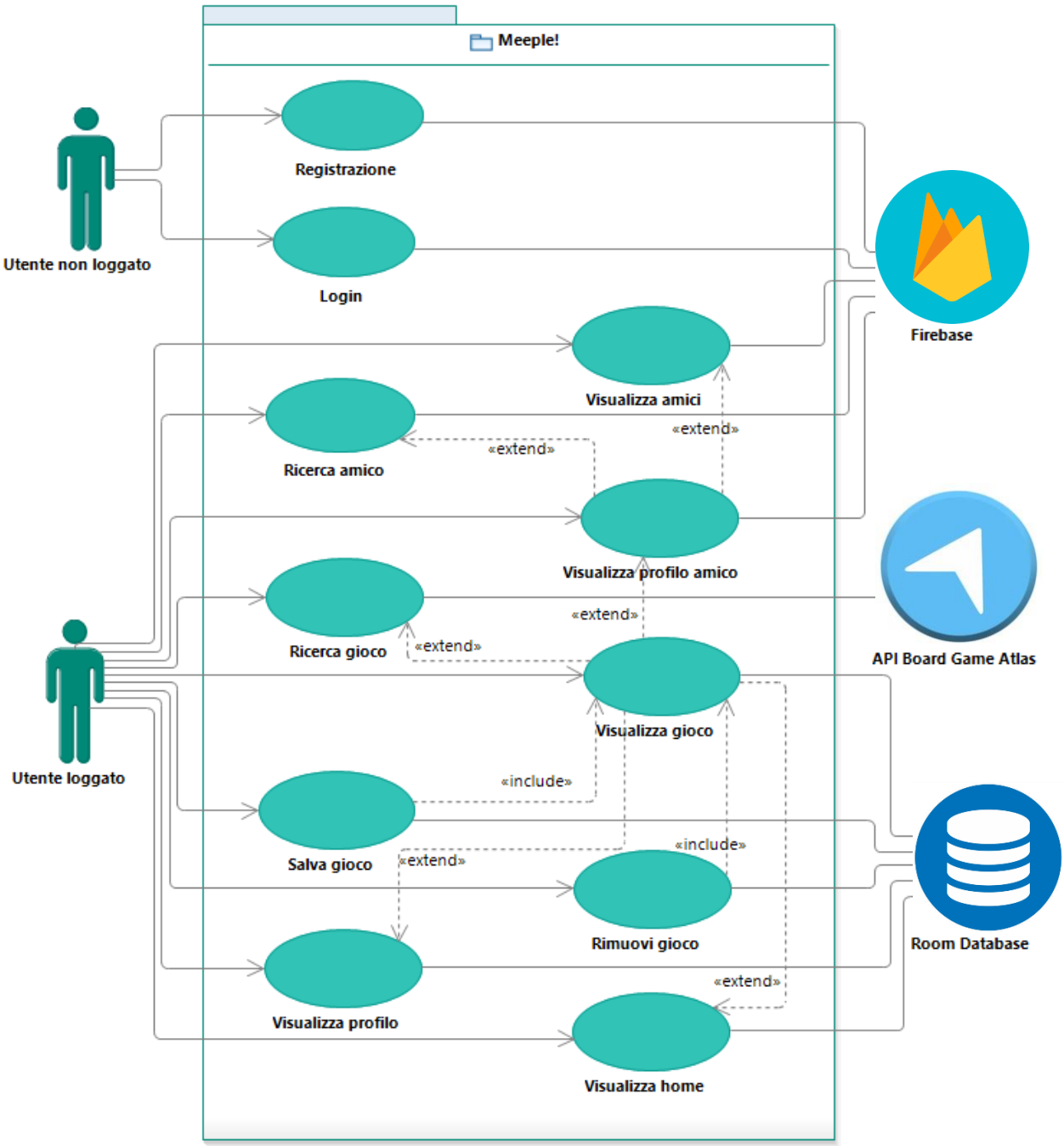
Ogni gioco può essere visualizzato nel dettaglio, per avere tutte le informazioni che possono servire all’utente a valutare un gioco e capire se salvarlo nel suo profilo.

Nel profilo dell’utente i giochi sono suddivisi in tre liste: quelli da giocare, quelli giocati e i preferiti. Ogni gioco presente potrà essere inserito in queste liste, così da creare un profilo che rispecchi gusti dell’utente.

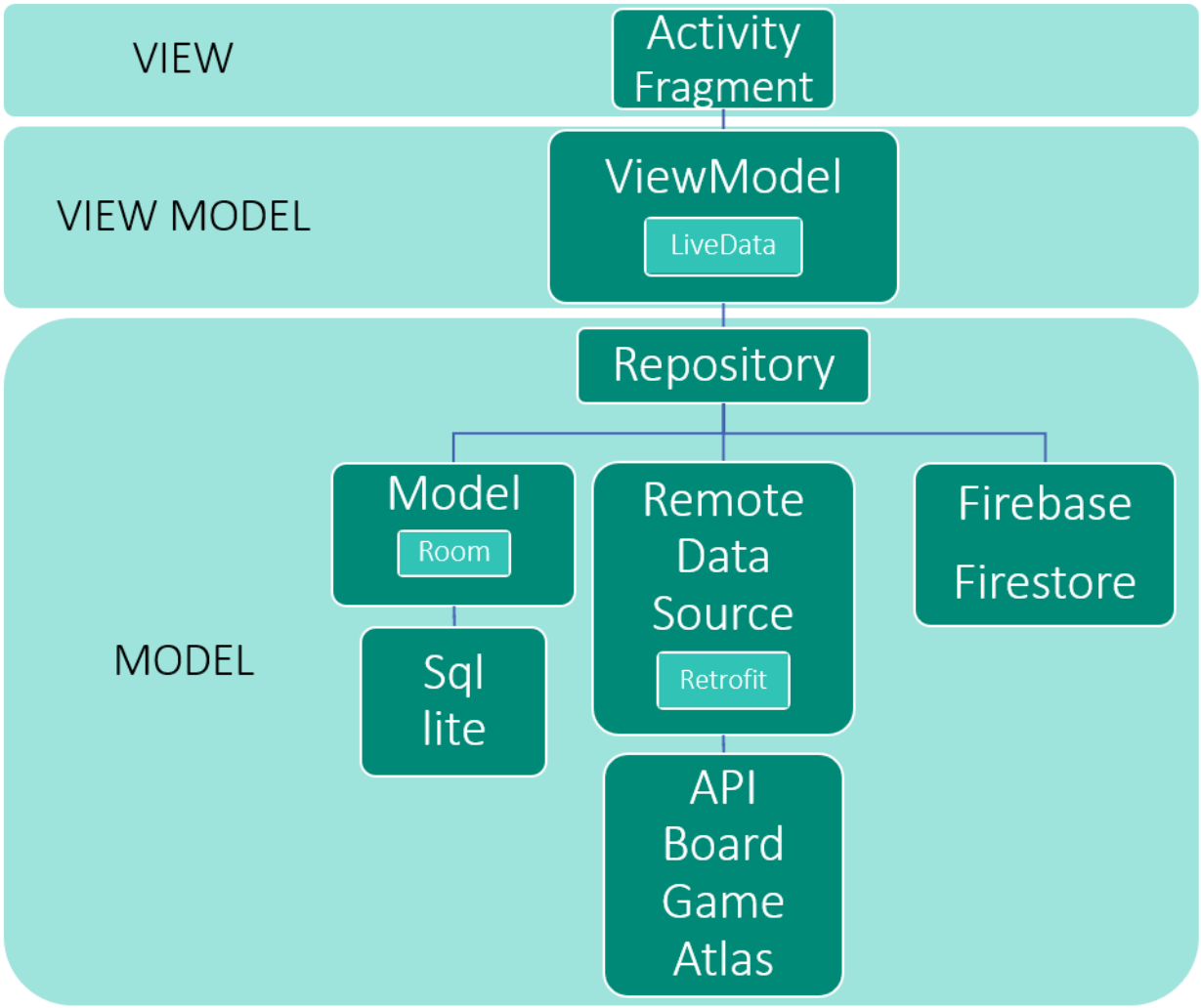
L’applicazione permette inoltre di cercare i profili degli altri utenti, e di aggiungerli alla propria lista di amici. In questo modo si potrà curiosare tra la lista dei giochi salvati da altri utenti, per poter scoprire ancora altri giochi.

L’applicazione può essere utilizzata in due lingue: italiano e inglese. La lingua in cui viene mostrata l’applicazione è la stessa settata nelle impostazioni del dispositivo.

DIAGRAMMA CASI D'USO



ARCHITETTURA



SERVIZI UTILIZZATI

- Retrofit: consente di accedere ai servizi REST per utilizzare l'API Board Game Atlas.

- API Board Game Atlas: REST API che fornisce un ampio database di giochi da tavolo con molte caratteristiche per ogni gioco, di cui abbiamo estrapolato solo le più adatte al nostro utilizzo.
  Offre la possibilità di fare GET con vari parametri, funzionalità che non abbiamo sfruttato nella versione finale dell'applicazione, perché abbiamo deciso di salvare sul database locale i giochi che vengono utilizzati di più (https://www.boardgameatlas.com/api/docs).

- Firebase Authentication: sfruttata per dare la possibilità all'utente di creare un proprio account con e-mail e password, oppure con le credenziali di Google.
  Utilizziamo anche la funzionalità di modifica password tramite e-mail offerta da Firebase Authentication.

- Firestore Database: abbiamo creato su Firestore Database una collezione (users), che contiene un documento per ogni utente. L'identificatore del documento di un utente corrisponde all'identificatore dell'account dell'utente stesso in Firebase Authentication, in modo da poter ottenere velocemente il documento relativo ad uno determinato utente.
  Abbiamo utilizzato Firestore per salvare tutti i dati di un utente, in modo da non farglieli perdere, neanche cambiando dispositivo o disinstallando l'applicazione. Per fare questo, ogni documento contiene il corrispettivo di uno User come da noi definito (con e-mail, username, bio, ecc.).
  L'utente autenticato ha quindi i suoi dati salvati su Firestore, ma anche sul database locale. Ogni volta che viene fatta una modifica ai dati relativi all'utente, questi vengono contemporaneamente aggiornati in locale e su Firestore, in modo da mantenere coerenti le informazioni.

- RoomDatabase: permette di salvare i dati su un database locale.
  Abbiamo creato un database games_db con le seguenti tabelle:
  - Game: contiene dei Game, scaricati dall'API Board Game Atlas, che vengono utilizzati per visualizzare rapidamente le pagine principali.
  - Category: contiene tutte le Category, scaricate dall'API Board Game Atlas.
  - GameCategory: associazione molti a molti tra Game e Category, che permette di ottenere velocemente i giochi per categoria.
  - User: contiene una sola istanza di User, che corrisponde all'utente loggato.

CLASSI

MODEL

Game	Rappresenta un gioco, con alcuni degli attributi significativi offerti dall'API, tra cui l'id, utilizzato come identificatore.
GamesGroup	Rappresenta un gruppo di Game, ogni gruppo ha un titolo.
User	Rappresenta un utente, identificato da una e-mail.
Category	Rappresenta una categoria di un gioco (con id e nome della categoria).
APICategory	Rappresenta l'elemento ritornato dalla API alla richiesta delle categorie relative a un gioco. Non viene mai utilizzata direttamente, perché ne viene subito fatto il parsing in Category.
GameCategory	Rappresenta la relazione molti a molti tra Game e Category.
GamesResponse	Contiene quanti e quali giochi sono stati ottenuti tramite la richiesta dei giochi all'API.
CategoriesResponse	Contiene la lista di categorie ottenute tramite la richiesta delle categorie all'API.
WelcomeCard	Rappresenta una card da mostrare all'utente alla prima installazione.

ADAPTER

GamesAdapter	Crea un collegamento tra un set di Game e la View che viene mostrato in una RecyclerView. Del Game si visualizzano titolo e immagine.
GamesGroupAdapter	Crea un collegamento tra un set di GamesGroup e la View che viene mostrato in una RecyclerView. Del GamesGroup si visualizzano titolo e lista di Game contenuti.
UsersAdapter	Crea un collegamento tra un set di User e la View che viene mostrato in una RecyclerView. Del User si visualizzano username e e-mail.
WelcomeAdapter	Crea un collegamento tra un set di WelcomeCard e la View che viene mostrato in una RecyclerView. Del WelcomeCard si visualizzano immagine, titolo e descrizione.

REPOSITORY

GamesRepository	Chiede e ottiene i dati dall'API se: • Il database è vuoto (alla prima installazione) • Serve ottenere un Game che non è nel database • Viene effettuata una ricerca In tutti gli altri casi i dati vengono presi direttamente dal database locale.
UsersRepository	Gestisce i dati relativi all'User presente nel database locale.
CategoryRepository	Chiede e ottiene i dati dall'API se il database è vuoto (alla prima installazione), altrimenti li prende dal database locale.

DATABASE

GamesRoomDatabase	Classe Singleton che inizializza il database locale.
GamesDao	Interagisce con la tabella Game del database, utilizzando le operazioni CRUD tramite query in SQLite.
GameCategoryDao	Interagisce con la tabella GameCategory del database, utilizzando le operazioni CRUD tramite query in SQLite.
CategoriesDao	Interagisce con la tabella Categories del database, utilizzando le operazioni CRUD tramite query in SQLite.
UsersDao	Interagisce con la tabella User del database, utilizzando le operazioni CRUD tramite query in SQLite.

SERVICE

GamesApiService	Effettua le richieste GET all'API Board Game Atlas. In particolare, permette di: • Ottenere una lista di giochi ordinati per rank, saltando un tot di giochi che sono già stati scaricati, in modo da poterli salvare sul database locale • Ottenere una lista di giochi passandogli i loro id • Ottenere una lista di giochi il cui titolo corrisponde a quello cercato, sfruttando l'attributo *fuzzy_match*, per poter ottenere i risultati anche se il nome inserito non corrisponde esattamente al nome del gioco • Ottenere tutte le categorie di gioco contenute nell'API

UTILS

Constants	Contiene alcune costanti utilizzate in tutta l'applicazione, sfruttata per potervi accedere da qualunque classe.
SharedPreferences Provider	Gestisce l'accesso alle SharedPreferences, che permette di salvare dati persistenti su un file in locale. Utilizziamo le SharedPreferences per salvare due attributi: • *first_installation*: viene settata a false non appena viene aperta l'installazione • *is_logged*: true se l'utente è loggato
Converters	Dichiara i converter che permettono di trasformare i risultati in formato JSON ottenuti dall'API in liste, e viceversa. Utilizza la libreria Gson.
ServiceLocator	Classe Singleton per ottenere l'oggetto Retrofit per effettuare le richieste GET all'API e l'oggetto RoomDatabase per accedere al database locale.
CategoryResponse Callback	Interfaccia che contiene i metodi da implementare quando viene effettuata una richiesta di categorie all'API.
ResponseCallback	Interfaccia che contiene i metodi da implementare quando viene effettuata una richiesta di giochi all'API.

UI

Ad ogni classe che accede ai dati del database, dell'API, di Firebase o Firestore è associato un proprio ViewModel, il quale si occupa di ottenere i dati.

I dati del database e dell'API vengono richiesti alla Repository, mentre quelli di Firebase e Firestore vengono richiesti direttamente all'interno del ViewModel.

SplashActivity

Permette di mostrare una schermata con il logo all'avvio e aggiorna la SharedPreference *first_installation*.

Viene sfruttata questa activity per inizializzare il database con alcuni giochi ottenuti dall'API Board Game Atlas.

Se è la prima volta che viene aperta l'applicazione dall'installazione si viene mandati alla WelcomeActivity, altrimenti alla LaunchScreenActivity.

WelcomeActivity

Mostra alcune cards con informazioni su Meeple!, per introdurre l'utente all'applicazione.

Viene visualizzata solo la prima volta che viene aperta l'applicazione. Manda direttamente alla LaunchScreenActivity.

LaunchScreenActivity

Controlla la SharedPreference *logged*, se è true (l'utente è loggato) l'applicazione manda direttamente alla MainActivity, altrimenti alla LoginActivity.

LoginActivity

Permette all'utente di fare il login con e-mail e password, oppure con le credenziali Google. Se l'utente non ha ancora un account con e-mail e password può andare alla SignupActivity per registrarsi.

Quando l'utente clicca su *SignIn con Google* gli viene chiesto con quale account Google desidera accedere. Se è la prima volta fa l'accesso a Meeple! con questa e-mail, viene mandato alla InfoProfileActivity.

Se invece fa il login con un account già esistente (sia con Google che con e-mail e password) viene mandato alla MainActivity.

Se vengono inserite credenziali non valide appare una SnackBar con un messaggio di errore. Questa viene mostrata nel caso in cui non esiste un account con l'e-mail inserita o la password non è corretta.

SignupActivity

Richiede l'inserimento di e-mail e password per creare un nuovo account. Una volta inseriti viene creato l'utente su Firebase Authentication, il suo relativo documento su Firestore Database e un'istanza di User viene salvata nel database locale.

Poi si viene reindirizzati alla InfoProfileActivity.

Si ha anche la possibilità di ritornare alla pagina LoginActivity tramite l'apposito bottone.

Se vengono inserite credenziali non corrette viene mostrata una SnackBar con un messaggio di errore. Questa viene mostrata nel caso in cui l'e-mail non è valida, la password ha meno di 6 caratteri o esiste già un account con l'e-mail inserita.

InfoProfileActivity

Richiede l'inserimento di username (obbligatorio), eventualmente di una bio e se l'utente è interessato a giochi per adulti o per bambini(1).

Manda poi alla InfoPreferencesActivity.

Gli attributi username, bio e isAdult (true se ha scelto giochi per adulti) vengono aggiornati nell'istanza di User salvata nel database locale e nel documento relativo al profilo su Firestore.

InfoPreferencesActivity

Chiede all'utente a quali categorie è interessato[2], in modo da poter personalizzare la HomeFragment.

L'utente deve selezionare almeno una categoria.

Manda poi alla MainActivity.

La lista di categorie selezionate viene aggiornata nell'attributo categoriesOfInterest dell'istanza di User salvata nel database locale e nel documento relativo al profilo su Firestore.

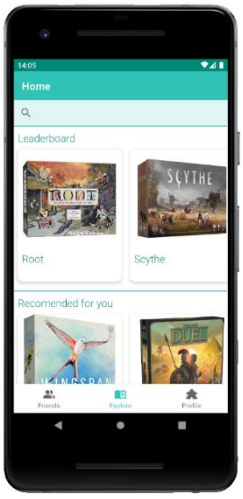
MainActivity

Contiene una Toolbar, un NavHostFragment e una BottomNavigationView.

Nella NavHostFragment viene visualizzato il Fragment corrente.

La BottomNavigationView permette di navigare tra FriendsFragment, HomeFragment e ProfiloFragment (HomeFragment è impostato come Fragment predefinito).

Tutti i Fragment sono contenuti nella MainActivity.



HomeFragment

Vengono visualizzate alcune liste di Game:

- BestGames: tutti i giochi in ordine di rank
- GamesByCategory: giochi di una categoria tra quelle di interesse dell'utente (ne viene selezionata una casualmente tra quelle che ha selezionato nella InfoPreferencesActivity)
- NewestGames: giochi usciti di recente
- QuickGames: giochi con una durata breve
- GamesByAge: giochi che si basano sulla scelta dell'utente tra giochi per bambini e giochi per adulti

Cliccando su un Game si viene rimandati al GameDetailsFragment.

È presente anche una barra di ricerca: nel momento in cui si scrive una stringa per la ricerca e si preme invio si viene rimandati a SearchFragment.

GameDetailsFragment

Visualizza le informazioni relative a un Game. Questo Fragment viene visualizzato ogni volta che si tocca su un gioco presente in qualunque pagina (HomeFragment, SearchFragment, ProfiloFragment e FriendDetailsFragment).

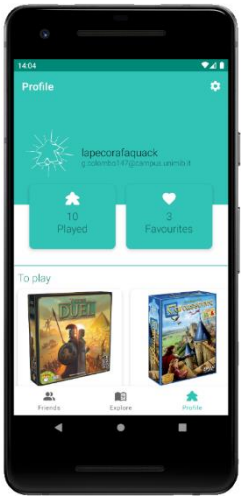
Sono presenti tre bottoni per poter aggiungere il gioco alla lista personale dell'utente di giochi *giocati*, *da giocare* e *preferiti*.

Vengono poi visualizzati il titolo del gioco, la sua immagine, numero minimo e massimo di giocatori, durata minima e massima del gioco, età minima, descrizione e categorie del gioco[3].

SearchFragment

Permette la ricerca di Game per nome del gioco[4] e ne mostra i risultati.

Un gioco può essere toccato per visualizzarne il GameDetailsFragment.

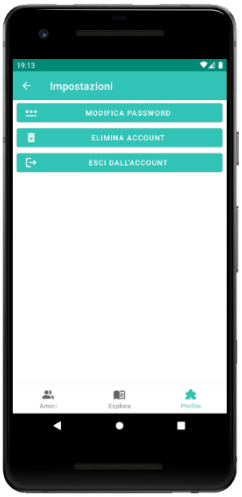


ProfiloFragment

Mostra il profilo dell'utente[5] con username, e-mail, numero di giochi giocati e preferiti, e vengono visualizzate le tre liste di Game:

- ToPlayGames: giochi che l'utente salva come da giocare
- PlayedGames: giochi a cui l'utente ha già giocato
- FavouriteGames: giochi aggiunti ai preferiti

Inoltre, da qui si può passare al SettingsFragment tramite l'OptionsMenu.

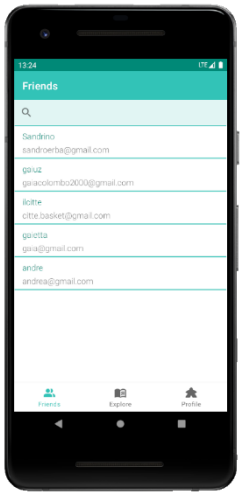


SettingsFragment

Nelle impostazioni è possibile modificare la password (ricevendo una mail gestita da Firebase), eliminare l'account ed eseguire il logout.

Quando viene eliminato l'account, questo viene eliminato da Firebase Authentication e anche il documento associato presente in Firestore.

Sia quando viene eseguito il logout, che quando viene eliminato l'account, si elimina l'istanza dell'utente dal database locale e si viene reindirizzati alla LoginActivity.

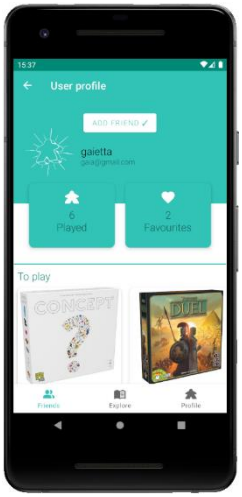


FriendsFragment

Viene visualizzata la lista di tutti gli amici[6] dell'utente corrente, mostrati per username e e-mail.

Toccando su un utente si può aprire il FriendDetailsFragment.

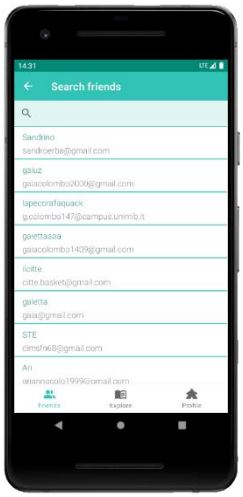
È presente anche una barra di ricerca: nel momento in cui si scrive una stringa per la ricerca e si preme invio si viene rimandati a FriendsSearchFragment.



FriendsDetailsFragment

Visualizza le informazioni relative al profilo di un utente, riutilizzando lo stesso file di layout del profilo.

In più viene data la possibilità di aggiungere il profilo alla lista di amici.

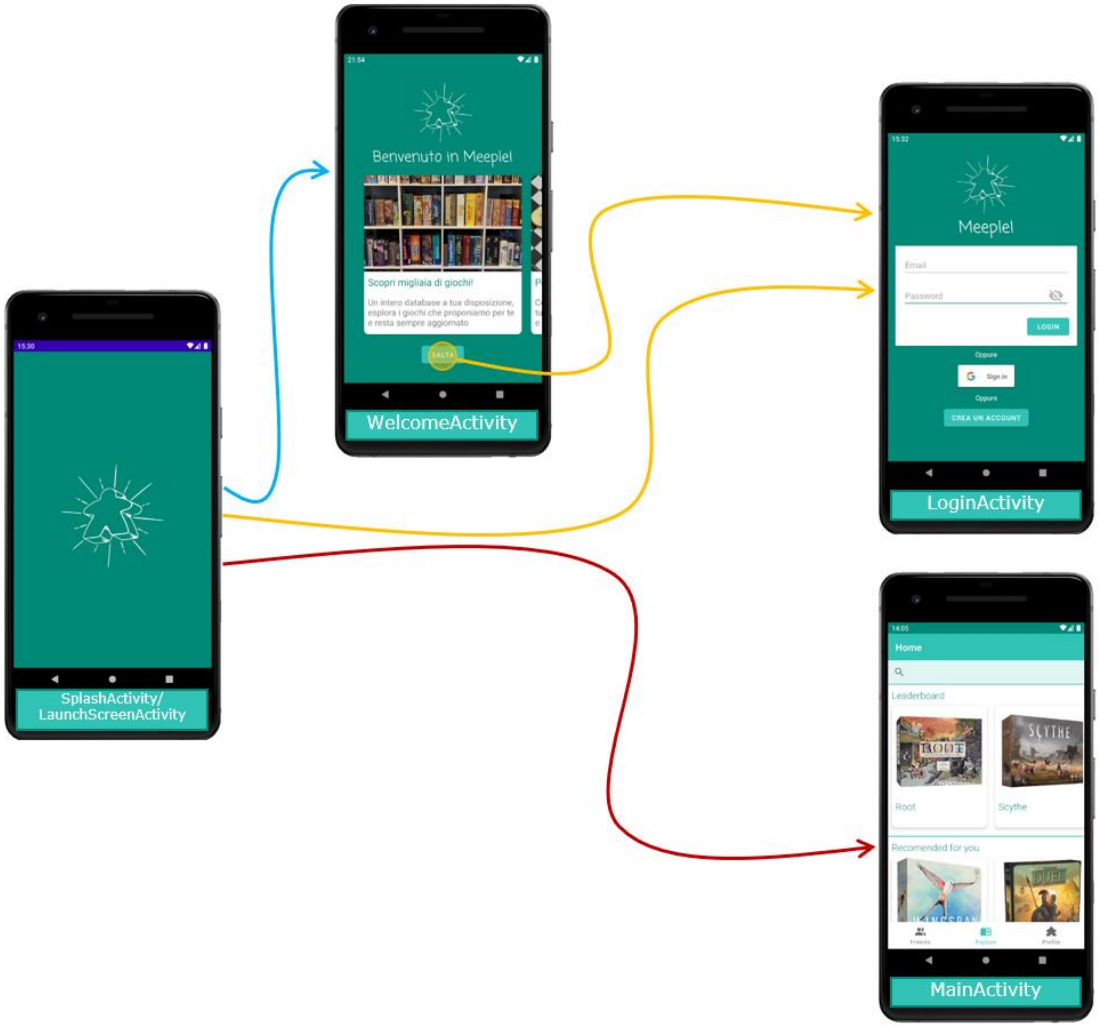


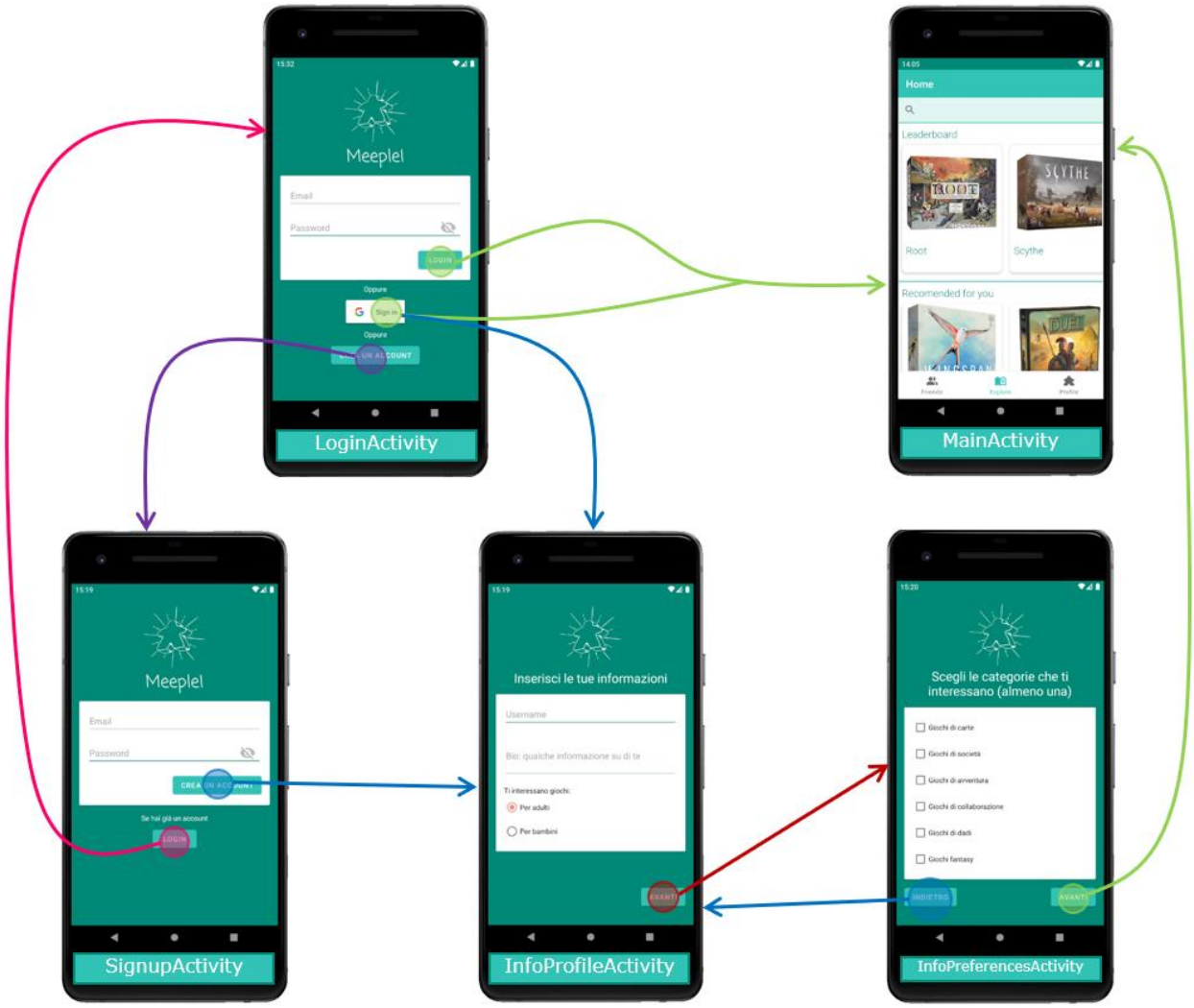
FriendsSearchFragment

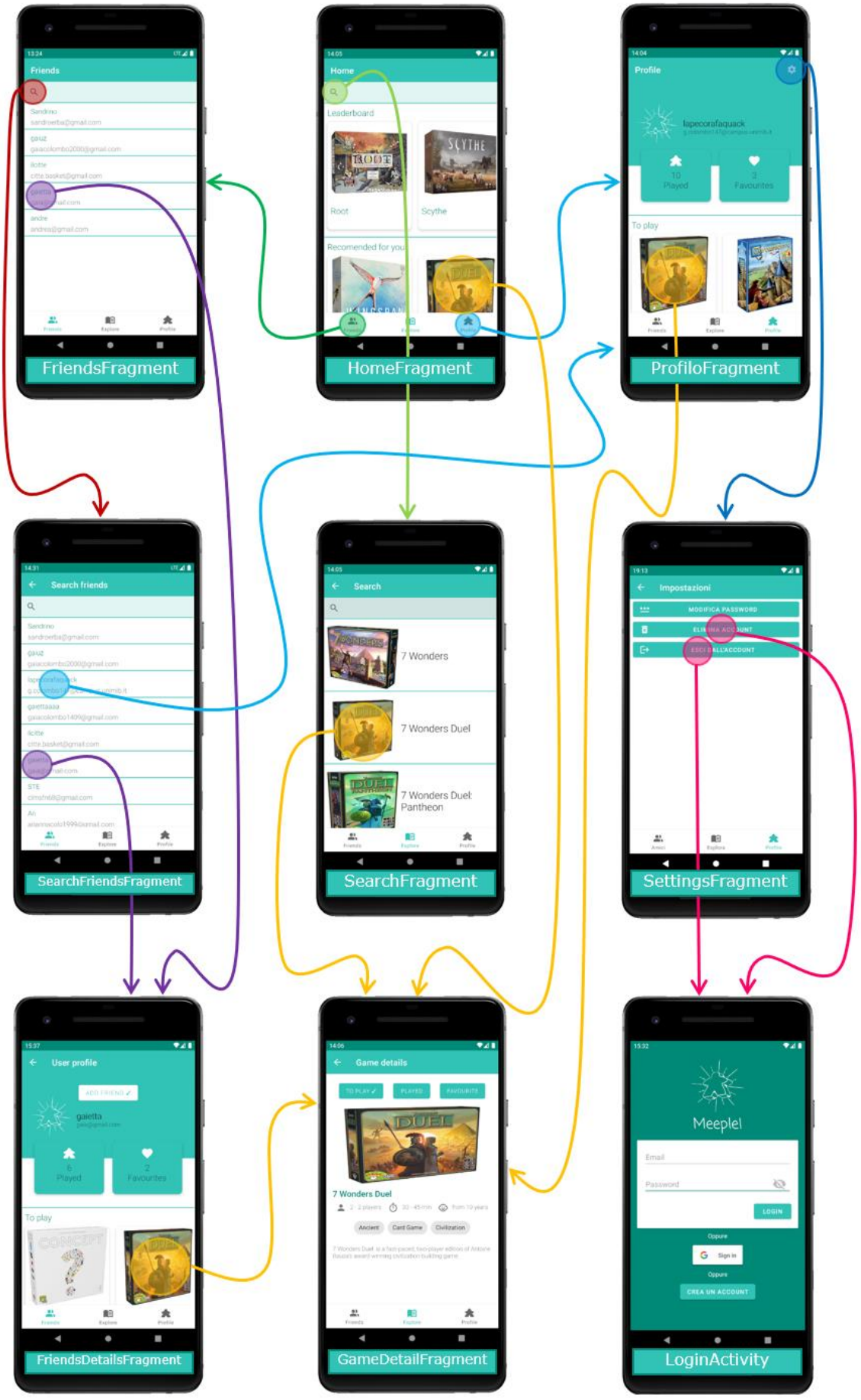
Permette di effettuare una ricerca di utenti per e-mail e mostrarne i risultati.

Vengono mostrati tutti i profili presenti su Firestore che contengono nella e-mail la stringa cercata.

INTERAZIONE TRA FRAGMENT/ACTIVITY




Meeplel
Email
Password
LOGIN
Oppure
Sign in
Oppure
CREA UN ACCOUNT
LoginActivity
Home
Leaderboard
ROOT
SCYTHE
Root
Scythe
Recomended for you
DUEL
MainActivity
Meeplel
Email
Password
CREA UN ACCOUNT
Se hai già un account
LOGIN
SignupActivity
Inserisci le tue informazioni
Username
Bio: qualche informazione su di te
Ti interessano giochi:
Per adulti
Per bambini
AVANTI
InfoProfileActivity
Scegli le categorie che ti interessano (almeno una)
Giochi di carte
Giochi di società
Giochi di avventura
Giochi di collaborazione
Giochi di dadi
Giochi fantasy
INDIETRO
AVANTI
InfoPreferencesActivity


Friends
FriendsFragment
Home
Leaderboard
Root
Scythe
Recomended for you
HomeFragment
Profile
lapecorafaquack
10
Played
3
Favourites
To play
ProfiloFragment
Search friends
SearchFriendsFragment
Search
7 Wonders
7 Wonders Duel
7 Wonders Duel: Pantheon
SearchFragment
Impostazioni
MODIFICA PASSWORD
ELIMINA ACCOUNT
ESCI DALL'ACCOUNT
SettingsFragment
User profile
gaietta
6
Played
2
Favourites
To play
FriendsDetailsFragment
Game details
TO PLAY
PLAYED
FAVOURITE
7 Wonders Duel
Ancient
Card Game
Civilization
GameDetailFragment
Meeple!
Email
Password
LOGIN
Oppure
Sign in
Oppure
CREA UN ACCOUNT
LoginActivity

RELEASE FUTURE

Abbiamo in programma di aggiungere, nelle prossime release, questi aggiornamenti per l'applicazione:

(1) Possibilità di poter aggiungere un'immagine profilo, scegliendo se caricarla dalla galleria personale o selezionarla tra un insieme di immagini predefinite.
(2) In fase di creazione del profilo, avere la possibilità di cercare tra tutte le categorie disponibili e non solo tra un elenco predefinito.
(3) Dare la possibilità di aprire una ricerca per categoria quando si tocca sulla Chip relativa a quella categoria.
(4) Filtrare i risultati della ricerca per categoria, numero di giocatori, durata del gioco e altre informazioni utili, oltre che per nome.
(5) Poter modificare il profilo in tutti i suoi campi, anche le categorie (scelte solo in fase di creazione del profilo).
(6) Notificare l'utente quando un suo amico aggiunge qualche gioco ai *giocati*.