

Capitolo 1

Applicazione

1.1 Funzionalità

L'applicazione aiuta la gestione di attività e problemi riscontrati durante una convivenza fra due o più persone, in ambito lavorativo o fra studenti fuori-sede.

Le funzionalità principali consentono ai membri di un gruppo di gestire una lista di faccende comuni da svolgere, gestire e dividere le spese, organizzare eventi periodici e/o ricorrenti e confrontarsi utilizzando la chat di messaggistica istantanea. L'applicazione avendo funzionalità molto generiche lascia il completo utilizzo di esse all'utente finale, permettendogli di gestire le varie funzionalità come meglio crede. Un esempio potrebbe essere la gestione degli eventi: degli studenti fuori sede potrebbero usare e creare eventi per organizzare i turni di pulizia all'interno della casa, assegnando eventi ricorrenti a coinquilini specifici, in ambito lavorativo invece, i membri del gruppo potrebbero utilizzare la funzione di gestione degli eventi per organizzarsi il lavoro o creare incontri aziendali.

L'utente dopo aver effettuato l'accesso potrà interagire con le funzionalità dell'applicazione selezionando l'icona della relativa funzionalità dal menù.

1.1.1 Gestione gruppo

Effettuata la registrazione o l'accesso all'applicazione, l'utente avrà la possibilità di entrare a far parte di un gruppo o crearne uno nuovo, ogni utente può fare parte di un solo gruppo.

L'utente che sceglie di entrare a far parte di un nuovo gruppo deve aver precedentemente ricevuto il codice invito da un membro appartenente ad un gruppo esistente. Una volta ricevuto il codice di invito, il nuovo utente dovrà inserire il codice e confermare di entrare a far parte del gruppo, se conferma verranno aggiornati i membri del gruppo e il gruppo di appartenenza dell'utente, gli altri membri del gruppo invece riceveranno una notifica.

Un nuovo utente ha anche la possibilità di creare un nuovo gruppo, aggiungendo solamente il nome e un'immagine opzionale.

1.1.2 Accesso e Registrazione

Al primo avvio dell'applicazione, verranno mostrate delle pagine scorrevoli che mostreranno le caratteristiche e funzionalità utilizzabili dall'utente, successivamente dopo una breve introduzione verrà mostrata la pagina di login, che permette di effettuare la registrazione e l'accesso attraverso un solo pulsante che senza differenziare se un utente sia già registrato o meno.

Quando l'utente cliccherà sul pulsante "accedi", l'applicazione automaticamente controllerà se l'utente si era già precedentemente registrato o deve effettuare la registrazione.

Il login e la registrazione possono essere effettuati utilizzando i social più diffusi o attraverso la semplice registrazione via email e password.

I social disponibili sono:

- Google Plus
- Facebook
- Twitter

Se l'utente sceglierá di registrarsi attraverso l'utilizzo di un email, gli verranno richiesti l'email di registrazione, un nominativo (Nome,Cognome) e una password, per effettuare il login invece verranno richiesti solo l'email e la password.

Alternativamente se l'utente seleziona il metodo di registrazione attraverso un social, comparirá a schermo una finestra che chiederá all'utente registrato al social di consentire l'utilizzo dell'email e del nome dell'utente da parte dell'applicazione, una volta ricevuta l'autorizzazione, le volte successive verrá effettuato un login automatico senza richiedere permessi aggiuntivi.

Un utente che ha dimenticato la propria password puó richiederne una nuova inserendo l'email di registrazione, in seguito dopo pochi secondi riceverá email un avviso per reimpostare la password e un link che permetterà di reimpostare la password.

1.1.3 Todolist

Selezionando dal menú dell'applicazione l'icona della "Todolist", l'utente visualizzerá l'interfaccia dedicata per interagire con le funzionalità quali: visualizzare le faccende da svolgere, visualizzare le faccende già svolte, aggiungere, modificare o eliminare una faccenda.

L'interfaccia per visualizzare le faccende é composta da due sezioni, la sezione delle faccende da completare, in primo piano e le faccende già completate in un'apposita sezione.

Le faccende sono composte da un nome obbligatorio, una data di scadenza, una priorità ed i membri del gruppo a cui é rivolta la faccenda.

Ogni utente visualizza la faccenda comprensiva di nome e data, la priorità invece viene indicata con un bordo colorato in base all'importanza della faccenda.

Gli utenti possono vedere sia le faccende create dagli altri membri del gruppo sia le loro faccende, in base alle restrizioni di visibilità assegnate durante la creazione, l'unica limitazione imposta riguarda le funzionalità di modifica ed

eliminazione, che sono consentite solamente all'utente che ha creato la faccenda, gli altri utenti invece potranno completare la faccenda marcandola. Le faccende che vengono marcate e completate vengono spostate automaticamente nell'elenco delle faccende completate e ogni utente avrà la possibilità di portare nuovamente una faccenda non completata nella sezione delle faccende da completare, senza dover aggiungerne un'ulteriore, la visibilità e la priorità della faccenda rimarranno inalterate.

L'aggiunta di una nuova faccenda viene effettuata attraverso due modalità differenti: la prima rapida, la seconda personalizzata.

La modalità rapida si trova nella parte superiore dello schermo, sottostante alla toolbar, in questa modalità l'utente può aggiungere un nuovo elemento indicando solamente il nome ed in automatico l'applicazione setterà i campi opzionali, impostando la data di scadenza alla data in cui è stato aggiunto l'elemento, la priorità di medio livello, e la visibilità a tutti i membri del gruppo.

La modalità personalizzata invece permette di inserire tutte le informazioni possibili per una faccenda, questa modalità di aggiunta compare se l'utente clicca la relativa icona presente nella toolbar della pagina "Todolist". Una volta cliccata l'icona si aprirà una finestra con un testo da completare corrispondente al nome della faccenda e tre icone: l'icona di una data, l'icona di un gruppo e l'icona della priorità, che se cliccate consentono all'utente di inserire le informazioni opzionali.

- **Priorità:** la priorità di una faccenda dispone di 3 opzioni: "Bassa priorità", "Alta priorità" e "Media priorità".
- **Visibilità:** la visibilità di una faccenda si potrà indicare selezionando dalla lista di utenti presenti nel gruppo a chi è rivolta la faccenda.
- **Data:** la data può essere selezionata, attraverso un calendario indicando il giorno della scadenza della faccenda.

1.1.4 Spese

La seconda funzionalità principale dell'applicazione é la gestione delle spese condivise, per accedere a questa funzionalità l'utente dovrà cliccare l'icona della funzionalità (un portafoglio) dal relativo menú.

L'interfaccia che si presenta all'utente é molto simile all'interfaccia della gestione delle faccende: é presente la visualizzazione globale delle spese da pagare e pagate, e la possibilità di aggiungerne modificarne o cancellarne una.

Ogni utente appartenente al gruppo avrà la possibilità di visualizzare tutte le spese non completate e quelle già completate e in qualsiasi momento potrà marcare una spesa, segnandola come "pagata". La visualizzazione di una singola spesa comprende il nome, la data di scadenza, l'icona della categoria a cui é associata la spesa, e la quota parziale che dovrà pagare l'utente. Cliccando su una spesa apparirà una finestra di dialogo che mostrerà il resoconto totale della spesa con la lista degli utenti che hanno pagato la loro quota e la lista degli utenti che ancora devono pagarla. Marcando una spesa l'utente segnerà di aver pagato la sua quota e di conseguenza la spesa, per quell'utente, verrà spostata automaticamente nell'elenco delle spese pagate. Le funzionalità di modifica ed eliminazione di una spesa sono consentite solamente all'utente che ha creato la spesa, gli altri utenti invece potranno solamente indicare di aver pagato la quota, marcandola.

Le modalità di aggiunta di una nuova spesa sono due, la modalità rapida e la modalità personalizzata.

La modalità rapida é accessibile attraverso l'interfaccia principale, nella parte superiore dello schermo infatti sono presenti due caselle di testo e un pulsante. Questa modalità permette di indicare solamente i parametri obbligatori: il nome e l'ammontare globale, alternativamente se l'utente vuole specificare anche altre informazioni, dovrà utilizzare il relativo pulsante per accedere alla finestra con tutti i campi opzionali per la creazione di una spesa.

La modalità di aggiunta personalizzata invece prevede un'interfaccia con una casella di testo corrispondente al nome della spesa, e un'altra casella di te-

sto corrispondente all'ammontare globale, sottostante alle caselle ci saranno tre icone: l'icona di un calendario, l'icona di un file, l'icona di un gruppo e l'icona di un etichetta, che se cliccate consentiranno all'utente di inserire le informazioni opzionali.

- Descrizione: casella di testo che permette di inserire una breve descrizione della spesa
- Visibilità: la visibilità di una spesa si potrà indicare selezionando dalla lista di utenti presenti nel gruppo a chi é rivolta la faccenda.
- Data: la data può essere selezionata, attraverso un calendario indicando il giorno della scadenza della faccenda.
- Categoria: Categoria che indica il tipo di spesa effettuata (Affitto, Bolletta, Spesa generica..)

1.1.5 Chat

L'applicazione offre una chat di messaggistica istantanea integrata, che consente di comunicare con tutti i membri appartenenti al gruppo in tempo reale.

L'interfaccia della sezione chat é simile ad altre applicazioni di messaggistica istantanea e consente di visualizzare tutti i messaggi inviati dall'utente e ricevuti dagli altri membri del gruppo.

I messaggi inviati dall'utente saranno contrassegnati con un colore blu e si troveranno nella parte destra dello schermo, mentre i messaggi ricevuti dagli altri membri del gruppo si troveranno nella parte sinistra dello schermo con un colore differente e informazioni aggiuntive come il nome e l'avatar dell'utente che ha inviato il messaggio all'interno del gruppo. Nella parte inferiore dello schermo é presente una casella di testo e un pulsante che permette di scrivere e inviare un nuovo messaggio che sarà inviato in tempo reale a tutti i membri del gruppo, infatti una volta inviati, i messaggi appariranno come notifica a tutti i dispositivi online, se in quel momento l'utente non dispone

di una connessione ad internet il messaggio verrà conservato e l'utente verrà notificato appena si conetterà ad internet.

1.1.6 Eventi

La gestione delle pulizie ed eventi generici e' gestita attraverso un calendario che permette l'aggiunt adi eventi ricorrenti o di singola durata

1.1.7 Menú

L'applicazione offre due menú differenti, il men' delle funzionalità e il menú delle impostazioni.

Il menú delle funzionalità si trova nella parte inferiore dello schermo, mentre per accedere al menú delle impostazioni é, bisogner'liccare la relativa icona del menú presente nella toolbar.

Interagendo con il menú laterle delle impostazioni si potranno gestire informazioni personali dell'utente e gestire il gruppo.

Nella pagina riguardante il profilo dell'utente sarà possibile visualizzare le informazioni personali come il nome, l'email e il gruppo a cui esso appartiene, queste informazioni sono modificabili in qualsiasi momento.

Nella pagina riguardante il gruppo invece sarà possibile visualizzare le informazioni principali riguardanti il gruppo a cui l'utente appartiene, come: il nome e gli utente che appartengono al gruppo. Le informazioni modificabili in questa pagina sono l'immagine del gruppo, il nome del gruppo e la possibilità di invitare altre persone ad unirsi al gruppo tramite invito, (verrà inviato all'utente un codice di invito che inserirá al momento del login).

1.2 Sviluppo

Inizialmente, é stata definita l'architettura dell'applicazione, le funzionalità principali e una bozza del design, come linguaggio di programmazione

per la parte client era stato scelto Java, mentre per gestire l'autenticazione il database e le notifiche é stato utilizzato Firebase come BaS.

Successivamente dopo aver testato le funzionalità di Java e le caratteristiche di Firebase furono presi in considerazione Kotlin come linguaggio di programmazione per Android, in alternativa a Java e Firestore come database alternativo a RealTime Firebase.

Parte del codice dell'applicazione Android é stato quindi scritto in due linguaggi differenti: Java e Kotlin, questo é stato utile anche per avere un confronto in termini di prestazione complessità e linee di codice.

1.3 Client

Il Client é stato strutturato in modo da contenere in Package differenti i componenti principali dell'applicazione.

- Activities: Contiene le Activity utilizzate dall'applicazione
- Pages: Contiene quattro packages corrispondenti alle 4 funzionalità dell'app
- Models: Contiene i modelli, utili per il pattern MVP
- Repositories: Contiene classi che facilitano la gestione e le richieste con il database
- Services: Contiene due servizi per la gestione delle notifiche
- Utils: Contiene classi utili per la gestione della cache, PreferenceShared, Componenti view personalizzati ecc

La struttura di organizzazione dei file e dell'implementazione delle caratteristiche delle quattro funzionalità dell'app é la stessa, inoltre il pattern utilizzato per la gestione fra le varie componenti del progetto per interagire con i dati e l'interfaccia utente é MVP (Model View Presenter).

Esiste una Activity principale chiamato BaseActivity che gestisce il funzionamento dei menú (Menu delle impostazioni, Menú delle funzionalità), e si occupa di mostrare le quattro pagine principali, realizzate estendendo la class Fragment.

I quattro Fragment sono:

- FragmentTodo
- FragmentSpese
- FragmentEventi
- FragmentChat

I fragment vengono cambiati e gestiti dall'Activity principale, che in base all'interazione con il menu delle funzionalità, si interscambiano.

Listing 1.1: Hello.kt in Kotlin

```
supportFragmentManager.beginTransaction().replace(R.id.activity_content,
    TodoFragment()).commit()
```

Quando si seleziona una della pagine, il controllo dell'interfaccia passa al relativo Fragment, che in base alla funzionalità mostrerà e permetterà di agire sull'interfaccia.

1.3.1 Teniche di programmazione

La logica per gestire l'interazione con l'utente e l'aggiornamento dei dati si basa sul pattern MVP, di conseguenza ogni Fragment o Activity che richiede di mostrare dei dati, implementerà i seguenti componenti:

- Adapter: Estensione della class RecyclerView.Adapter che contiene gli elementi da visualizzare
- Presenter: Componente che ha il compito di richiedere al Repository i dati da visualizzare

- View: Interfaccia grafica della pagina con cui l'utente può interagire
- Model: Modello astratto di un elemento del Database
- Repository: Classe che permette di inviare richieste al Database, o di restituire le query necessarie per la richiesta

Per le richieste al database viene utilizzata la tecnica di programmazione reactive, messa a disposizione dalla libreria RxJava2 e RxKotlin...

1.3.2 Repository

All'interno del package Repositories sono presenti le classi necessarie per interagire con il database Firestore.

Sono presenti due tipi di classi, le Query e le Repository, le prime contengono solamente la query necessaria per svolgere una determinata azione all'interno del database, le seconde invece utilizzando le Query effettuano la richiesta e restituiscono una risposta al chiamante.

Listing 1.2: Aggiunta elemento Todolist

```
fun getTodoItems(): Single<List<TodoItem>> {  
    return Single.create<List<TodoItem>> { emitter ->  
        queryTodo.getTodoItems().addSnapshotListener({  
            querySnapshot, exception ->  
                if (exception != null)  
                    emitter.onError(Throwable(exception))  
                else {  
                    emitter.onSuccess(TodoItem.mapping(querySnapshot))  
                }  
            })  
        }  
    }  
}
```

1.3.3 Utils

Il package Utils contiene classi di supporto per azioni comuni che vengono svolte dai componenti dell'applicazione.

In particolare le principali sono:

- PreferenceUtils: classe utilizzata per gestire le SharedPreferences
- ImageUtils: classe che interagendo con la libreria Glide, aggiunge funzionalità alla gestione delle immagini
- RxFirestore: classe che estende alcuni tipi dell'SDK Firestore per implementare l'Observable
- UserSpinner: Componente View che estende AppCompatActivity, creato per mostrare e selezionare graficamente gli utenti attraverso uno spinner
- FirestoreCMUtils: classe che gestisce i Token utilizzate da Cloud Messaging

1.3.4 Modelli

I modelli sono stati realizzati utilizzando le data class offerte da Kotlin, in questo modo non è stato necessario implementare i metodi get e set, l'unica aggiunta effettuata è stata la creazione di due nuovi metodi chiamati "mapping" che permettono di convertire i tipi DocumentSnapshot e QuerySnapshot restituiti dal'SDK Firebase come risposta del database, in modelli utilizzabili come oggetti all'interno dell'applicazione, mappando quindi ogni valore contenuto negli Snapshot all'interno dei singoli valori del modello.

1.3.5 Servizi

I servizi utilizzati dall'applicazione sono stati creati per interagire con il servizio Cloud Messaging di Firebase.

I servizi sono due:

- `FirebaseMessagingService`
- `FirebaseInstanceIdService`

Il primo `FirebaseMessagingIdService` gestisce i token necessari per utilizzare Cloud Messaging, in particolare gestisce l'aggiornamento del token del dispositivo, inviando una richiesta di aggiornamento al Database Firestore, qualora il token sia aggiornato o eliminato.

Il secondo servizio invece `FirebaseInstanceIdService` gestisce i messaggi ricevuti dal server di Cloud Messaging, implementando infatti il metodo `onMessageReceived` sarà possibile inanzitutto capire il tipo di messaggio: messaggio di notifica o messaggio contenente dati, successivamente in base al tipo di messaggio ricevuto vengono effettuate modifiche o mostrate le adeguate notifiche.

In particolare quando viene ricevuto il messaggio di aggiunta di un nuovo membro nel gruppo, oltre ad inviare la notifica per avvisare i dispositivi, vengono anche aggiornate le `PreferenceShared` che contiene localmente una copia delle informazioni riguardanti il gruppo senza dover contattare ogni volta il database.

1.3.6 Todolist

Il fragment `TodolistPage`, si occupa di gestire l'aggiunta rapida di un nuovo elemento e la logica delle due Tab "Da completare" "Completato".

L'aggiunta un elemento con la modalità rapida inserendo solamente il nome della faccenda, viene svolta da un Thread che preleva dalla view il nome della faccenda, inizializza un nuovo elemento `Todolist`, e la relativa `Repository`, successivamente utilizzando la tecnica di programmazione `RxJava`, un observer resterà in ascolto della richiesta al database per aggiungere un nuovo elemento, finché questo non sarà aggiunto o in caso negativo mostrerà un errore.

Listing 1.3: Aggiunta elemento Todolist

```
val todoItem = TodoItem(name = itemName, date = Date(), members =
    members, created_by = userID)
todoRepo.add(todoitem =
    todoItem).observeOn(AndroidSchedulers.mainThread()).subscribeOn(Schedulers.io()).do
{
    todolist_edittext_newitem.setText("")
    Toast.makeText(context, "todoitem successfully created!",
        Toast.LENGTH_SHORT).show()
}.doOnError {
    Toast.makeText(context, "error!",
        Toast.LENGTH_SHORT).show()
}.subscribe()
```

Il componente `TabLayout` che mostra le due `Tab` richiede l'utilizzo di un `Adapter` che estende la class `FragmentStatePagerAdapter`, in questo modo quando un utente interagirà con una delle due `Tab`, l'`Adapter` si occuperà di instanziare il `Fragment` corrispondente per visualizzare la lista delle faccende completate o non completate.

Dato che per visualizzare gli elementi della `todolist` sono necessari due `fragment` che avrebbero lo stesso compito, si è scelto di realizzarne solamente uno, che in base al valore del parametro `type` passato nella creazione dell'istanza del `fragment`, visualizzerà elementi differenti.

Listing 1.4: `FragmentTodo.kt`

```
companion object {
    fun newInstance(type: Int): TodoFragment {
        val fragment = TodoFragment()
        fragment.type = type
        return fragment
    }
}
```

1.3.7 Spese

Il fragment `TodolistPage`, si occupa di gestire la logica dell'aggiunta rapida di un nuovo elemento e la gestione delle due Tab "Da completare" "Completato".

Il componente `TabList` che mostra le due Tab per funzionare richiede l'utilizzo di un Adapter che estende la class `FragmentStatePagerAdapter`, in questo modo quando un utente interagirà con una delle due Tab, l'Adapter si occuperà di instanziare il Fragment corrispondente per visualizzare la lista delle faccende completate o non completate.

Ricapitolando l'Activity principale ospita il Fragment `Tosolist` che a sua volta per una corretta interazione con il componente `TabLayout` richiede di instanziare tante fragment quante sono le Tab, in questo caso due.

Dato che i due fragment per visualizzare gli elementi della todolist hanno lo stesso compito, si è scelto di realizzarne solamente uno che in base al valore del parametro `type` passato nella creazione dell'istanza del fragment, verranno visualizzati elementi differenti.

Il `FragmentTodo` basandosi sul pattern MVP implementerà l'interfaccia `ToDoListView`, istanziando il `Presenter` e l'Adapter per richiedere i dati della todolist e visualizzarli nella View.

1.3.8 Chat

La chat come le precedenti pagine, utilizza il pattern MVP, di conseguenza all'interno del fragment verranno istanziati il `Presenter` l'Adapter e implementata la View.

La chat per facilitare la visione dei messaggi ricevuti e inviati all'interno dell'Adapter prevede due differenti `ViewHolder` per differenziare il messaggio inviato da quello ricevuto.

Per Realizzare questa distizione grafica fra i due tipi di messaggio sono stati sovrascritti alcuni metodo del Fragment e creati degli Holder e classi astratte apposite.

Inizialmente é stata creata una classe astratta ChatHolder che avesse come unico metodo la funzione "bind" e prendesse come parametro il messaggio, i due holder verranno implementati basandosi e estendendo questa classe.

Successivamente sono state sovrascritti i metodi getItemViewType e onCreateViewHolder del Fragment in modo tale da introdurre il controllo necessario per distinguere i messaggi.

All'interno del metodo getItemViewType, viene controllato il campo "SenderId" del messaggio, corrispondente all'ID dell'utente che ha inviato il messaggio nel gruppo, questo ID viene poi confrontato con l'ID dell'utente loggato all'interno dell'applicazione in modo da restituire l'identificativo del layout da utilizzare nel ViewHolder.

Listing 1.5: FragmentTodo.kt

```
when (senderUid == userUid) {  
    false -> return R.layout.item_message_recived  
    true -> return R.layout.item_message_sent  
}
```

Una volta differenziato il tipo attraverso il metodo getItemViewType, é stata sovrascritto il metodo onCreateViewHolder che dovendo restituire un singolo Holder, come valore di ritorno restituira un tipo ChatHolder (class astratta implementata dalle due ViewHolder). In questo modo in base al tipo restituito da getItemViewType la classe onCreateViewHolder restituirá l'holder corrispondente al messaggio.

Listing 1.6: FragmentTodo.kt

```
when (viewType) {  
    R.layout.item_message_sent -> holder =  
        MessageChatSentHolder(itemView = view,
```

```
        dateOfLastMessage = lastItem?.timestamp)
    R.layout.item_message_recived -> holder =
        MessageChatRecivedHolder(itemView = view,
            dateOfLastMessage = lastItem?.timestamp)
}
```

1.4 Server