

LAUREAPP

Progetto sviluppato dal gruppo APPlicatori composto da:

Lotito Vito, 717828

Lopez Giuseppe, 724324

Mancino Andrea, 716475

Mele Giuseppe, 735774

Panza Francesco, 739551

Repository progetto:

https://github.com/LotitoVito/Progetto_SMS_APPlicatori.git

Sommario

DINAMICHE DI SVILUPPO	3
DIVISIONE DEI RUOLI	3
CLASSI	3
DIAGRAMMA ER	6
STRUMENTI UTILIZZATI.....	7
CASI D'USO E SPRINT	8
LIBRERIE ESTERNE.....	10
SVILUPPO FRONTEND.....	10
SVILUPPO BACKEND	14
LIMITI DELL'APP E SVILUPPI FUTURI.....	17
CONCLUSIONI SULL'APP	18

DINAMICHE DI SVILUPPO

Lo sviluppo dell'applicazione è partito agli inizi di Dicembre e durante le prime riunioni si è deciso come organizzarne lo sviluppo: la divisione del team tra sviluppo backend e sviluppo frontend, il diagramma delle classi, il diagramma ER, quali strumenti utilizzare, i casi d'uso da realizzare e la divisione in sprint

DIVISIONE DEI RUOLI

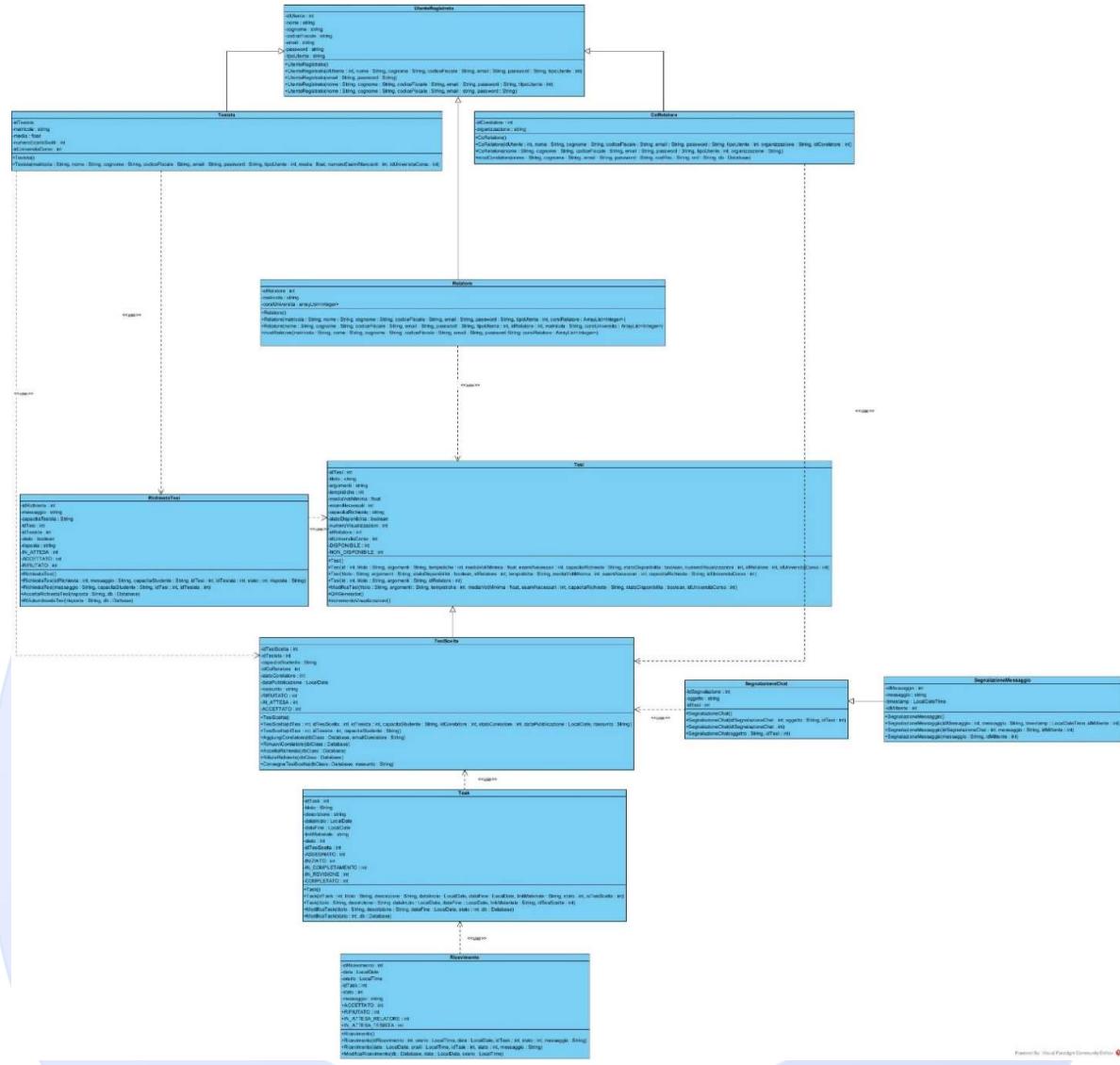
Il gruppo è stato diviso nella seguente maniera:

- Backend: Lotito Vito, Francesco Panza, Andrea Mancino
- Frontend: Giuseppe Lopez, Giuseppe Mele

Questa divisione deriva dalle capacità in Frontend e Backend di ogni membro.

CLASSI

Le classi individuate per il progetto hanno subito diversi refactor e modifiche man mano che lo sviluppo proseguiva. Abbiamo identificato 3 tipi di utenti più l'utente guest: tesista, relatore e corelatore; tutti e tre sono specializzazioni di un utente generale che ha campi come nome, cognome, email, ecc., ma ogni tipo di account ha degli attributi e funzioni peculiari. Assieme agli utenti abbiamo identificato anche classi utilizzate da quest'ultimi come tesi, ricevimento, task, ecc. La versione finale del diagramma delle classi è la seguente:



Di seguito una descrizione del diagramma e delle classi:

- **UtenteRegistrato**: la classe rappresenta l'utente generale che utilizza l'applicazione; ha attributi utili a tutti e tre i tipi di utente: nome, cognome, codice fiscale, email, password; ha anche un campo per specificare il tipo di utente dell'account registrato.
- **Tesista**: la classe rappresenta l'utente studente che frequenta l'università; il tesista ha attributi utili per valutare le sue capacità e la sua situazione universitaria quali media, esami mancanti, matricola; il tesista ha anche un id per identificare quale università e corso frequenta; la classe estende UtenteRegistrato.

- Relatore: la classe rappresenta l’utente professore universitario che svolge il ruolo da relatore per i tesisti; gli attributi riguardano principalmente il suo ambito universitario come universitaCorsi e matricola; la classe estende UtenteRegistrato.
- Corelatore: la classe rappresenta l’utente che collabora allo svolgimento delle tesi assegnate ai tesisti; il corelatore può essere una persona esterna all’ambito universitario, quindi, ha un campo organizzazione per identificare il suo ambito d’appartenenza; la classe estende UtenteRegistrato.
- Tesi: la classe rappresenta le proposte di tesi fatte dal relatore per cui i tesisti si possono candidare; la tesi oltre agli attributi identificativi come titolo, argomento, idRelatore e idUniversitaCorso ha anche attributi vincoli come tempistiche, minimoEsamiMancanti, minimoMedia e capacitàRichieste; la tesi presenta anche un attributo disponibilità per renderla disponibile o limitata ai tesisti che si vogliono candidare; la tesi ha anche un attributo visualizzazioni per ordinare le tesi in base alle più visualizzate; la classe estende Tesi.
- RichiestaTesi: la classe rappresenta le richieste alle proposte di tesi fatte dai tesisti, possono essere risposte dai relatori; la richiesta tesi ha attributi riguardanti le parti coinvolti quali gli id di tesista, tesi e relatore; la richiesta tesi ha anche campi messaggio, stato, e risposta per permettere il dialogo tra le due parti e gestire le richieste.
- TesiScelta: la classe rappresenta la tesi in stato di svolgimento da un tesista la cui richiesta è stata accettata; la tesi scelta ha attributi identificativi delle parti quali gli id di tesista, relatore, corelatore e tesi; la tesi scelta ha l’attributo statoCorelatore per gestire la collaborazione con il corelatore; la tesi scelta ha anche attributi per la consegna come il riassunto, ovvero l’abstract della tesi una volta consegnata, e la dataPubblicazione.
- SegnalazioneChat: la classe rappresenta le segnalazioni che possono essere fatte dagli utenti dell’app nei confronti di una tesi specifica a cui il relatore relativo potrà rispondere; la segnalazione chat ha attributi identificativi della chat come l’oggetto e la tesi a cui fa riferimento.
- SegnalazioneMessaggio: la classe rappresenta i messaggi che vengono scambiati in una chat di segnalazione tra le due parti; il messaggio ha attributi quali l’idMittente, il messaggio e il timestamp per gestire il flusso della chat e la comunicazione tra le due parti. La classe estende SegnalazioneChat
- Task: la classe rappresenta i compiti assegnati da relatore e corelatore nei confronti del tesista della tesi in stato di svolgimento; il task ha attributi di identificazione e per lo svolgimento come idTesiScelta, titolo, descrizione,

`dataInizio`, `dataFine` e `linkMateriale`; il task ha un attributo `stato` che permette di identificare la percentuale di completamento e consegna.

- Ricevimento: la classe rappresenta un ricevimento proposto dal tesista per discutere dello svolgimento di un task; il relatore può accettare, rifiutare o modificare data e orario che il tesista dovrà poi accettare o rifiutare; i ricevimenti sono visibili anche ai corelatori ma non modificabili; il ricevimento ha attributi per l'organizzazione dell'incontro come data e orario; il ricevimento ha anche attributi per la comunicazione e gestione della proposta tra le due parti.

Ognuna delle classi descritte ha un id personale usato per recuperare le informazioni nel caso di necessità.

L'applicazione ha anche altre classi di supporto quali:

- Classi database: ogni classe sopracitata ha una propria classe database per gestire la registrazione, recuperare i dati e modificarli sul database.
- Classi liste: anche queste sono classi database e recuperano liste di oggetti utili in base alle query dei metodi; le liste fanno riferimento ad alcune classi del modello come tesi, task, ricevimenti, ecc. e ogni lista ha la sua classe.
- Classe Utility: questa classe è usata per tener conto di quali account hanno effettuato l'accesso, contiene costanti di conversione di data e diversi metodi di verifica e utilizzati in più parti di codice.

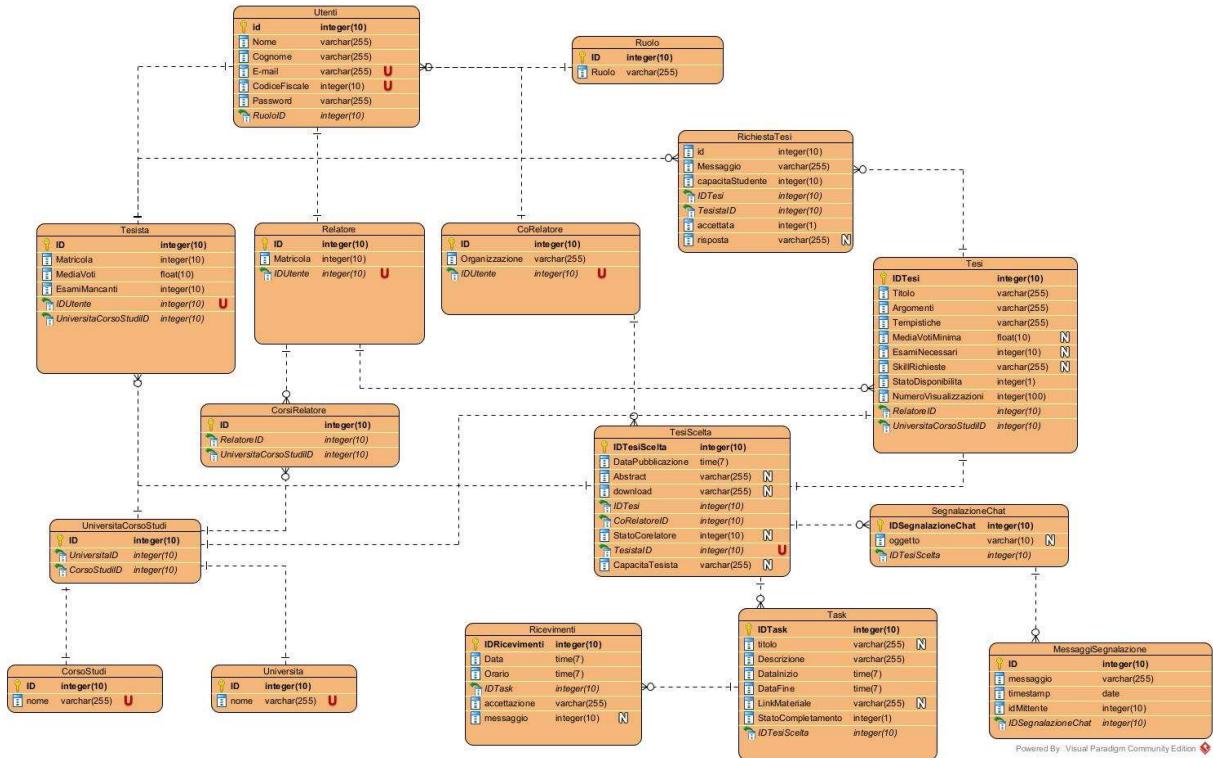
Le classi del modello hanno diversi metodi che per la maggior parte svolgono funzioni di modifica dei dati in collaborazione con le proprie classi database.

Altre classi create sono le activity e fragment che gestiscono il flusso e le view dell'applicazione, hanno generalmente metodi di `Init()` per creare i riferimenti agli elementi delle view utili, metodi `SetTextAll()` per mostrare i testi corretti nelle view e altri metodi per la gestione degli input e controlli.

DIAGRAMMA ER

Il diagramma ER ha anch'esso subito diverse modifiche durante lo sviluppo dell'applicazione in base alle esigenze delle funzioni da realizzare e miglioramenti dal punto di vista della ridondanza.

La versione finale del diagramma ER è la seguente:



Il diagramma ER è basato sul diagramma delle classi e le tabelle aggiuntive non riguardano direttamente gli utenti poiché gestiscono le scelte di università, corso e tipo di account. I valori di queste tabelle sono gestiti dagli admin dell'applicazione in base alle necessità.

Il database viene creato con SQLite nella classe Database con il metodo onCreate(). Nella classe sono presenti costanti per nomi delle tabelle e nomi delle colonne. Per motivi di beta ha anche un metodo di autopopolamento delle tabelle.

Oltre al diagramma ER viene usato Firebase per conservare i file caricati per task o per le tesi completate, questo servizio è descritto in un documento a parte.

STRUMENTI UTILIZZATI

Per quanto riguarda gli strumenti utilizzati si è discusso della permanenza dei dati: inizialmente si è optato per MYSQL, ma nelle prime fasi di sviluppo ci siamo accorti di diverse difficoltà nella configurazione; quindi abbiamo deciso di utilizzare SQLite già nativo di Android Studio al posto di MYSQL poiché supporta SQL che i membri del backend hanno preferito e non ha bisogno di configurazioni particolari. La scelta di usare SQLite è stata per gran parte del progetto molto utile ai fini di testing dell'app, poiché permette modifiche immediate e non necessita connessioni; abbiamo però

riscontrato dei problemi con l'utilizzo di questo strumento nel momento in cui dovevamo implementare download e upload di file. SQLite offre il tipo Blob per il caricamento dei file, ma lo sviluppo è stato complesso e non ha portato i risultati che ci si aspettava; si è deciso quindi di utilizzare per queste funzioni un altro strumento di persistenza dei dati: Firebase. La configurazione di Firebase è stata semplice e ha permesso di sviluppare upload e download in maniera semplice ed efficace; inoltre l'id del file conservato in Firebase è sempre memorizzato nel database SQLite.

L'altro strumento che si è utilizzato è stata la libreria ZXing per le funzionalità relative al QRCode; si rimanda la documentazione della libreria.

CASI D'USO E SPRINT

Dopo diverse revisioni, i casi d'uso identificati e da sviluppare sono stati i seguenti:

1) UTENTE GENERALE

- a) Registrazione
- b) Login
- c) Logout
- d) RecuperoPassword
- e) ModificaProfilo
- f) VisualizzaProfilo
- g) CambioLingua
- h) CambioTema
- i) ListaTesi
- j) ScanQR
- k) Condivisione
- l) ListaRicevimenti
- m) VisualizzaRicevimento
- n) CreaSegnalazione
- o) ListaSegnalazioni
- p) RispondiSegnalazione
- q) VisualizzaTesiScelta
- r) DownloadTesi
- s) ListaTask
- t) DettagliTask
- u) ModificaTask

- v) DownloadTask
 - w) UploadTask
- 2) TESISTA
- a) CreaRicevimento
 - b) RispondiRicevimento
 - c) RichiediTesi
 - d) VisualizzareRichiesteTesi
 - e) UploadTesi
 - f) ConsegnareTesi
- 3) RELATORE
- a) RispondiRicevimento
 - b) RegistrazioneTesi
 - c) ModificaTesi
 - d) AccettaRichiestaTesi
 - e) ListaTesiScelte
 - f) UploadTesi
 - g) AggiungiCorelatoreTesiScelta
 - h) RimuoviCorelatoreTesiScelta
 - i) CreaTask
- 4) CORELATORE
- a) AccettaPartecipazioneTesi
 - b) ListaTesiScelte
 - c) CreaTask
- 5) GUEST
- a) ListaTesiCompletate

L'organizzazione degli sprint è stata realizzata ad ogni inizio sprint. Sono stati compiuti 4 sprint:

- Primo sprint: Son state realizzate le funzioni relative ai profili degli utenti come Registrazione, Login, ecc.
- Secondo sprint: Son state realizzate le funzioni relative alle tesi del relatore come RegistrazioneTesi, ListaTesi, ecc.
- Terzo sprint: Son state realizzate le funzioni relative a ricevimenti, segnalazioni e richieste se si appoggiano sulle tesi registrate

- Quarto sprint: Son state realizzate le funzioni di Tesista, Corelatore e la gestione delle Tesi Scelte dai Tesisti

Lo sviluppo ha incontrato diversi problemi che son stati risolti con dei refactor quali riguardanti campi mancanti ad alcune classi e refactor di alcune tabelle del db.

LIBRERIE ESTERNE

Le librerie estere utilizzate sono:

- Firebase, utilizzato per le funzioni di upload e download dei file; i file caricati sono relativi ai task e tesi completate o in via di sviluppo, l'id di questi file è caricato sul database SQLite;

link:

<https://firebase.google.com/> ;

dipendenze gradle:

```
'com.google.firebaseio:firebase-storage:20.0.1'  
'com.google.firebaseio:firebase-database:20.0.4'
```

;

- ZXing, utilizzato per le funzioni relative ai QRCode; sono stati realizzati lo scanner e il QRCode per ogni tesi;

link:

<https://github.com/journeyapps/zxing-android-embedded> ; dipendenze gradle:

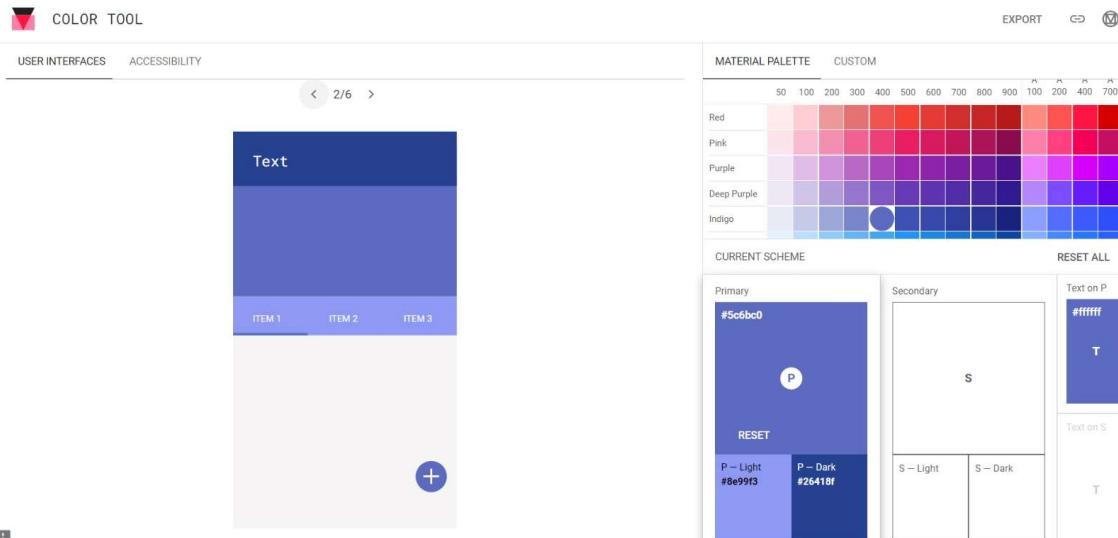
```
'com.journeyapps:zxing-android-embedded:4.3.0'
```

;

SVILUPPO FRONTEND

Lo sviluppo frontend è stato realizzato da Giuseppe Mele si è occupato dei layout relative alle funzionalità di registrazione, di login e di visualizzazione dei diversi profili utenti mentre Giuseppe Lopez si è occupato della realizzazione dei layout di tutte le altre funzionalità. Di comune accordo abbiamo deciso di sviluppare seguendo le line guida del Material Design versione 2. Questo perché al momento di inizio di sviluppo dell'applicazione il Material Design 3 era stato rilasciato da poco e volevamo offrire

all'utente un'esperienza più morbida e intuitiva. Dopo la scelta della versione di Material Design da utilizzare è stata effettuata una scelta relativa ai colori che l'applicazione dovesse avere. Seguendo come riferimento la color emotion guide, i due Giuseppe hanno deciso che, l'applicazione dovendo essere uno strumento di semplificazione di gestione e creazione delle tesi sia per i relatori che per gli studenti, dovesse suscitare negli utenti sicurezza e affidabilità e allo stesso tempo anche creatività. Quindi è stato deciso che il colore primario dovesse trovarsi tra il blu e il viola. Sono stati fatti quindi dei prototipi con il color tool del material design 2 e tutti i membri del gruppo di comune accordo hanno scelto utilizzare la seguente Material Palette:



Dopodiché è stata effettuata la progettazione dell'icona, il cui approfondimento è fatto a parte in un altro documento. È stata creata poi l'homepage da parte di Giuseppe Lopez, il layout è molto semplice prevede come background lo stesso tipo di gradiente utilizzato all'interno dell'icona e prevede il logo dell'applicazione, e tre bottoni: il primo per accedere, il secondo per iscriversi ed il terzo per accedere senza essere registrati. Abbiamo deciso poi di creare 3 activity principali, una per il login, una per la registrazione ed una per la visualizzazione delle varie funzionalità dell'app. Abbiamo deciso di sviluppare l'app in questa maniera perché dobbiamo gestire 3 tipi di utenti diversi che svolgono però le stesse funzionalità con tipologie di azioni diverse a seconda del tipo di utente. Abbiamo deciso quindi di aiutarci con i fragment per favorire il riutilizzo dei layout. Per il login è stata implementata un'interfaccia di base molto semplice che prevede l'inserimento dei campi e-mail e password, un bottone di recupero password e il bottone di login. L'interfaccia di recupero password prevede l'inserimento due volte dell'e-mail e della password. Il layout dell'activity di

registrazione invece prevede semplicemente un FrameLayout, al suo interno verranno poi caricati per primo il layout utente generico dove verrano inseriti i campi in comune per ogni utente e poi a seconda della tipologia di utente verrà caricato il fragment per l'inserimento dei campi mancanti. L'activity per l'utente loggato si sviluppa alla stessa maniera, prevede un FrameLayout in cui verranno caricati i layout delle varie funzionalità e poi una BottomNavigationview.

Le funzionalità previste per ogni utente registrato sono le seguenti:

Home: un layout in cui è presente un calendario settimanale (che al momento dell'accesso è impostata alla data del giorno corrente) ed in base alla data selezionata nella ListView presente sotto il calendario vengono visualizzati gli eventi che l'utente ha quel giorno.

Ricevimenti: il layout per la creazione dei ricevimenti prevede tre EditText: uno per l'inserimento di un messaggio inserito da parte dello studente e due che riguardano data e ora. L'input di questi due EditText sono disabilitati in maniera tale che al click sugli EditText vengono aperti DatePickerDialog per la data e il TimePickerDialog per l'ora e al click sull'ok dei relativi dialog vengono inseriti negli EditText i valori scelti. La proposta di ricevimento verrà quindi inviata al relatore e nel layout della visualizzazione della proposta oltre a vedere il messaggio e data e ora del ricevimento avrà la possibilità di accettare e rifiutare il ricevimento (sono presenti due buttoni) oppure di proporre allo studente un cambio di data e/o ora, il fragment del reschedule del ricevimento è lo stesso della sua creazione quindi conterrà sempre un messaggio (in questo caso sarebbe il messaggio di risposta) e i due campi data e ora.

Proposte di tesi: In questo layout è presente un campo per la ricerca del titolo, un tasto filtro in cui è possibile filtrare le tesi in base ai vari campi di ricerca e una semplice ListView in cui vengono visualizzate le proposte di tesi effettuate dai relatori. L'item nella lista ha un titolo, la descrizione e lo stato della tesi e due buttoni, il primo che se cliccato apre un bottomsheet che mostra i dettagli della proposta di tesi e il suo codice qr e da qui per tutti gli utenti sarà possibile creare una segnalazione per quanto riguarda la tesi. Il secondo bottone invece è visibile solo al relatore che ha creato la tesi e permette di modificare i dettagli di quella che viene da noi chiamata proposta di tesi.

Segnalazione: in questo layout è presente una ListView in cui vengono elencate l'elenco delle segnalazioni aperte. Gli item nella lista sono cliccabili e una volta cliccata viene aperta la pagina relativa alla segnalazione scelta che viene trattata come una chat e quindi sulla sinistra verranno visualizzati i messaggi inviati da altri utenti e sulla

destra i messaggi inviati dall'utente loggato. I messaggi hanno come sfondo un drawable realizzato sempre da Lopez che contiene una layer-list in cui sono presenti due item: il primo è un elemento rettangolare ruotato di 45° (il punto di rotazione cambia a seconda se il messaggio è ricevuto o inviato) e il secondo elemento che viene posizionato a 16dp a sinistra/destra (dipende se è messaggio ricevuto o inviato) rispetto al primo elemento. Il colore degli item è impostato su primaryColor per i messaggi inviati e primaryColorLight per i messaggi ricevuti. Sul fondo del layout è presente un editText con un background realizzato tramite un drawable creato sempre da Lopez che prevede un rettangolo con gli angoli arrotondati e di colore grigio (impostato come searchColor nei colori ed è lo stesso drawable utilizzato anche per la ricerca delle tesi) e accanto all'EditText è presente un bottone di invio realizzato con un ImageButton che ha come background un drawable che rappresenta un cerchio con il primaryColor come colore di sfondo e come immagine un drawable con l'icona del messaggio.

Le mie tesi: Per relatore e corelatore prevede due tab: una per le richieste delle tesi e fa vedere la lista delle richieste di tesi effettuate l'altro in c'è una lista con tutti i tesisti. Al click su un tesista si apre una schermata in cui è possibile visualizzare i vari dettagli della tesi in corso, scaricare e caricare le tesi e creare e visualizzare task. Per lo studente invece viene visualizzato, quando non ha ancora una tesi una pagina che fa vedere l'elenco e lo stato delle richieste effettuate. Se una sua proposta di tesi viene accettata a quel punto verrà invece visualizzata la pagina della tesi con la possibilità di inserire l'abstract, di caricare e scaricare la tesi e di visualizzare i task assegnati da relatore e corelatore.

Task: La creazione del task prevede EditText per i campi Nome, Descrizione e Data di consegna del task (che funziona come per le date dei ricevimenti), uno slider per indicare lo stato del task (che di default è settato a 0) e un campo materiale in cui è possibile caricare del materiale per il task. Mentre il layout per la visualizzazione del fragment oltre ad avere gli stessi campi (solo che al posto di utilizzare degli EditText vengono utilizzate delle TextView) e poi oltre allo slider e alla possibilità di caricare e scaricare materiale (i bottoni vengono visualizzati in base al tipo di utente loggato), per lo studente è possibile richiedere un ricevimento.

Scan QR: la sezione scan qr al click apre la fotocamera e permette la scansione di un codice qr. Se il codice qr è valido verrà reindirizzato alla pagina della tesi a cui si riferisce il codice. L'interfaccia dello scanner qr è delegata alla libreria esterna Zxing

che si occupa appunto di tutto il lato relativo alla creazione e scannerizzazione dei codici qr e che è stata documentata nella sezione apposita.

Filtri: Nell'app sono stati creati diversi filtri per le varie ricerche. Questi sono tutti stati implementati come bottomsheet che si aprono al click della scritta "Filtri" e al loro interno contengono semplicemente i vari campi per cui è possibile filtrare gli item nelle liste.

Sviluppo Backend

Lo sviluppo backend è stato realizzato da Lotito Vito che si è occupato dei refactor e della maggior parte dei casi d'uso, Panza Francesco che si è occupato delle liste, della visualizzazione di esse e della creazione e modifiche della struttura del database e Mancino Andrea che si è occupato di casi d'uso di visualizzazione e modifica.

Lo sviluppo backend è andato di pari passo con quello frontend, sono state create classi per ogni entità del model, classi database che si occupano della gestione di operazioni con il database per ogni classe del model, classi lista per alcune classi del model, classi activity e fragment per la gestione del flusso dell'app e una classe Utility con le funzioni e le variabili più utili e riutilizzate.

Riportiamo una descrizione dei casi d'uso nello specifico:

- 1) UTENTE GENERALE (Tesista, Relatore, Corelatore)
 - a) Registrazione: l'app permette la registrazione di un account del tipo selezionato; in base al tipo di account si avranno diversi campi obbligatori da compilare oltre a quelli in comune.
 - b) Login: l'app permette di accedere con le proprie credenziali.
 - c) Logout: l'app permette di uscire dal proprio account
 - d) RecuperoPassword: l'app permette di recuperare la password del proprio account nel caso di problemi.
 - e) ModificaProfilo: l'app permette di modificare le informazioni del proprio account nel caso ci fosse il bisogno.
 - f) VisualizzaProfilo: l'app permette di visualizzare le informazioni del proprio profilo.
 - g) CambioLingua: l'app permette di cambiare la lingua in italiano o inglese.
 - h) CambioTema: l'app permette di cambiare il tema in scuro o chiaro.

- i) [ListaTesi](#): l'app permette di visualizzare una lista di proposte di tesi caricate dai relatori; la lista può essere filtrata e ordinata secondo diversi campi.
 - j) [ScanQR](#): l'app permette di scannerizzare i QRCode presenti nella visualizzazione della tesi; se il QRCode corrisponde ad una tesi, l'app chiederà all'utente se vuole visualizzare la tesi; se l'utente conferma gli verrà mostrata la pagina di visualizzazione di quella tesi.
 - k) [Condivisione](#): l'app permette di condividere le informazioni della tesi visualizzata con le applicazioni suggerite o di copiare il messaggio.
- 2) [ListaRicevimenti](#): l'app permette di visualizzare la lista dei ricevimenti dell'utente che ha effettuato l'accesso; la lista è organizzata a calendario e mostrerà i ricevimenti esistenti della data selezionata o tutti i ricevimenti dalla data odierna in poi se nessuna data è selezionata.
- 3) [VisualizzaRicevimento](#): l'app permette di visualizzare le informazioni del ricevimento selezionato.
- 4) [CreaSegnalazione](#): l'app permette di aprire una chat di segnalazione con il relatore della tesi visualizzata; le segnalazioni fanno riferimento alle tesi; al momento della creazione della segnalazione viene già inviato un messaggio richiesto all'utente.
- 5) [ListaSegnalazioni](#): l'app permette di mostrare la lista delle chat di segnalazione dell'utente che ha effettuato l'accesso ordinate dalla più recente alla più vecchia.
- 6) [RispondiSegnalazione](#): l'app permette di rispondere nelle chat di segnalazione con nuovi messaggi permettendo il dialogo tra le due parti.
- 7) [VisualizzaTesiScelta](#): l'app permette di visualizzare la tesi in svolgimento selezionata.
- 8) [DownloadTesi](#): l'app permette di scaricare i file della tesi completata selezionata.
- 9) [ListaTask](#): l'app permette di visualizzare la lista di task relativi alla tesi in svolgimento selezionata.
- 10) [DettagliTask](#): l'app permette di visualizzare i dettagli di un task selezionata.
- 11) [ModificaTask](#): l'app permette di modificare un task selezionato; relatori e corelatori possono modificare titolo, argomento, stato e file caricati; i tesisti possono modificare i file caricati e lo stato.
- 12) [DownloadTask](#): l'app permette di scaricare il file caricato di un task selezionato.
- 12) [UploadTask](#): l'app permette di caricare un file per un task selezionato.
- 13) TESISTA
- 14) [CreaRicevimento](#): l'app permette di creare un ricevimento con il relatore della tesi in svolgimento del tesista; i ricevimenti fanno riferimento ai singoli task; i ricevimenti sono visibili anche ai corelatori coinvolti.

- 15) RispondiRicevimento: l'app permette di rispondere a una controrisposta del relatore per data e ora del ricevimento; il tesista potrà solo accettare o rifiutare.
- 16) RichiediTesi: l'app permette di richiedere l'assegnazione di una proposta di tesi selezionata; il tesista può avere una richiesta in attesa per volta; se il tesista ha già una tesi assegnata non potrà fare richiesta e non potrà più visualizzare la lista di richieste di tesi; il tesista può fare richiesta anche per tesi per cui non soddisfa i requisiti, ma solo per le tesi della propria università e corso.
- 17) VisualizzareRichiesteTesi: l'app permette di visualizzare una richiesta di tesi selezionata.
- UploadTesi: l'app permette di caricare un file relativo alla tesi completata assegnata al tesista.
 - ConsegnareTesi: l'app permette di consegnare la tesi ormai completata con la scrittura di un abstract della tesi e il caricamento di file relativi.
- 18) RELATORE
- RispondiRicevimento: l'app permette di rispondere a una richiesta di ricevimento; il relatore può accettare, rifiutare o creare una controrisposta; nella cotorrisposta il relatore dovrà indicare data e orario che preferisce e attenderà la conferma del tesista.
 - RegistrazioneTesi: l'app permette di registrare una proposta di tesi; nella registrazione saranno da inserire anche i vincoli per l'accesso ad essa; nella registrazione si può settare la disponibilità della tesi e il corso a cui fa riferimento; i corsi selezionabili sono quelli in cui il relatore insegna.
 - ModificaTesi: l'app permette al relatore di modificare i campi, i vincoli e le disponibilità delle proprie tesi.
 - AccettaRichiestaTesi: l'app permette al relatore di accettare o rifiutare le proposte di tesi fatte dai tesisti; il relatore può visualizzare il paragone tra i vincoli della tesi e le capacità, media, ecc. del tesista; il relatore può accettare un tesista anche se quest'ultimo non rispetta i vincoli della tesi.
 - ListaTesiScelte: l'app permette al relatore di visualizzare la lista delle proprie tesi assegnate e in svolgimento.
 - UploadTesi: l'app permette al relatore di caricare file per le tesi completate.
 - AggiungiCorelatoreTesiScelta: l'app permette al relatore di inviare una richiesta di partecipazione ad un corelatore; la richiesta di partecipazione viene fatta inserendo l'e-mail del corelatore da aggiungere.

- h) RimuoviCorelatoreTesiScelta: l'app permette al relatore di rimuovere un corelatore da una tesi in svolgimento; può essere rimossa anche una richiesta di collaborazione.
 - i) CreaTask: l'app permette al relatore di creare dei task da svolgere per il completamento della tesi in svolgimento; il relatore può caricare file utili allo svolgimento del task; il task ha una data di consegna.
- 19) CORELATORE
- a) AccettaPartecipazioneTesi: l'app permette al corelatore di accettare o rifiutare la collaborazione per una tesi in svolgimento.
 - b) ListaTesiScelte: l'app permette al corelatore di mostrare la lista di tesi in svolgimento per cui ha accettato la collaborazione.
 - c) CreaTask: l'app permette al corelatore di creare dei task da svolgere per il completamento della tesi in svolgimento; il corelatore può caricare file utili allo svolgimento del task; il task ha una data di consegna.
- 20) GUEST
- a) ListaTesiCompletate: l'app permette all'utente guest di visualizzare liste di tesi completate; le liste di tesi possono essere filtrate e ordinate secondo diversi campi.
 - b) VisualizzaTesiCompletata: l'app permette all'utente guest di visualizzare una tesi completata scelta; le liste di tesi possono essere filtrate e ordinate secondo diversi campi.

L'applicazione chiederà all'utente una conferma per le operazioni più importanti attraverso un dialog.

Durante lo sviluppo backend non sono stati trovati particolari problemi se non i già menzionati sulla gestione dei file.

LIMITI DELL'APP E SVILUPPI FUTURI

I limiti dell'applicazione riguardano la scelta dell'utilizzo di SQLite, che comporta diverse limitazioni quali:

- Il salvataggio dei file, abbiamo dovuto far ricorso ad un altro strumento, Firebase, cosa che non sarebbe stata necessaria nel caso non avessimo usato SQLite;
- I dati sono salvati in locale, questo vuol dire che due dispositivi diversi non avranno lo stesso database in comune e in più non possono comunicare;

- La funzione share delle tesi non permette la condivisione di un link per la tesi che si vuole condividere ma solo un messaggio testuale con le informazioni relative alla tesi condivisa;
- Problemi di sincronizzazione in tempo reale delle segnalazioni, il refresh della chat avviene uscendo e rientrando nella chat nel caso di un nuovo messaggio;

Queste limitazioni sono però da associare alla natura di demo dell'applicazione, in uno sviluppo futuro si potrebbe usare un database online oltre a SQLite che salverebbe le informazioni più importanti per l'utente, in maniera tale da non necessitare a forza di una connessione internet nel caso di visualizzazioni di alcune informazioni, ad esempio un tesista che vuole consultare la propria tesi in svolgimento.

Altre limitazioni derivano da problemi minori di comunicazione tra alcuni tipi di utenti come il corelatore che non può comunicare con il tesista direttamente, o il tesista non può comunicare con il corelatore nel momento di richiesta di collaborazione. Un obiettivo futuro può essere quello di creare chat di qualsiasi tipo per favorire la comunicazione e collaborazione delle parti e anche l'implementazione di chat con membri multipli

Altre limitazioni derivano dalla quantità di file caricabili e la visualizzazione di essi, per natura di demo ci siamo limitati a permettere il caricamento di un solo file per tesi/task, ma in un futuro sviluppo si può allargare il numero di file caricabili e creare anche un'organizzazione di essi.

Altre limitazioni derivano dai filtri sviluppati al momento solo per le tesi, in un futuro sviluppo si potrebbero sviluppare filtri anche per chat e altri tipi di liste.

Altre limitazioni derivano dall'assenza di verifica sulla veridicità dei dati, in un futuro sviluppo si potrebbe creare un sistema di verifica per la creazione di account certificati e dati veritieri per garantire la sicurezza.

CONCLUSIONI SULL'APP

L'app sviluppata ha diverse limitazioni derivanti da scelte basate sulla sua natura di demo come l'utilizzo di SQLite che è stato molto utile per lo sviluppo, tutta via mostra chiaramente quali sono le funzioni di base dell'applicazione con diversi controlli sugli input, viste chiare e intuitive e funzioni semplici da eseguire.