Università degli Studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Sviluppo mobile con supporto di LLM

Relatore: Prof. Daniela Micucci
Correlatore: Dott. Maria Teresa Rossi

Relazione della prova finale di:
Marco Napolitano
Matricola 886187

Anno Accademico 2023-2024

Sommario

1 Introduzione

Negli ultimi anni, l'evoluzione esponenziale dei Large Language Model (LLM) ha aperto nuove prospettive in ambito di sviluppo software. Questa tesi esplora il potenziale degli LLM, con particolare attenzione a ChatGPT 3.5, e indaga la sua efficacia e utilità nel processo di sviluppo di un'applicazione mobile nel campo delle iniziative benefiche. Le motivazioni di questa tesi risiedono nella valutazione degli LLM come strumento di supporto e nella crescente importanza delle piattaforme digitali per promuovere e facilitare le donazioni e il volontariato.

CharityHub si propone come un hub centralizzato che raccoglie progetti benefici, mostra informazioni dettagliate e permette agli utenti di filtrarli per interesse o nazione. Durante lo sviluppo di CharityHub, ChatGPT si è rivelato particolarmente utile in diverse fasi del processo, soprattutto durante l'implementazione e il testing. È stato utilizzato per richieste specifiche e problemi confinati, fornendo suggerimenti e idee che hanno accelerato il lavoro.

Il codice fornito da ChatGPT non era sempre immediatamente utile o aggiornato, ma ha comunque offerto spunti preziosi per lo sviluppo, velocizzandone il processo. I test effettuati hanno mostrato che, pur richiedendo diverse modifiche, il contributo di ChatGPT ha avuto un impatto positivo sullo sviluppo di CharityHub. L'assistenza fornita ha permesso di risolvere problemi tecnici e di implementazione in modo più rapido, sebbene con la necessità di adattare e correggere il codice generato.

Nel capitolo dedicato all'utilizzo di ChatGPT verrà mostrato che al fine di ottenere i risultati migliori è necessario porre domande specifiche offrendo un contesto limitato e inerente. Si discuterà con dettaglio la sua effettiva utilità in ogni fase, partendo dalla stesura dei requisiti, passando per l'UI, implementazione e infine testing. Verranno inoltre discusse le metodologie per ottenere risultati più appropriati dopo aver ottenuto una risposta non soddisfacente come, rigenerare la risposta, continuare la conversazione o ricominciarla da capo cambiando il contesto.

Per tenere traccia dell'uso di ChatGPT, è stato utilizzato un processo sistematico. Ogni interazione con ChatGPT è stata documentata, annotando la richiesta effettuata, la risposta ricevuta, utilità dell'output, rielaborazione necessarie e numero di richieste. Questo ha permesso non solo di valutare l'efficacia di ChatGPT, ma anche di identificare pattern ricorrenti nei tipi di problemi risolti e nelle soluzioni proposte.

Il lavoro è stato svolto in ambiente Android, utilizzando Android Studio come ambiente di sviluppo e Java come linguaggio di programmazione principale.
Per il progetto di testing sono stati utilizzati tre framework: Espresso e Roboelectric per gli UI test e Mockito per gli unit test delle view models e repositories.

La relazione è organizzata come segue:

1. **Introduzione**: Presentazione del contesto e delle motivazioni della tesi.
2. **Analisi dei requisiti**: Analisi dei requisiti funzionali e non funzionali dell'app
3. **Implementazione**: Descrizione dettagliata dell'implementazione
4. **Progetto di testing**: Descrizione della test suite prodotta
5. **Utilizzo di ChatGPT**: Descrizione e analisi sull'utilizzo di ChatGPT
6. **Conclusioni:** Sintesi dei principali risultati della tesi.

In sintesi, questa tesi dimostra come l'integrazione dei Large Language Model, in particolare ChatGPT 3.5, possa apportare significativi vantaggi nel processo di sviluppo di applicazioni mobile nel settore delle iniziative benefiche. Pur con alcune limitazioni, l'uso di ChatGPT ha accelerato lo sviluppo, migliorando l'efficienza e la risoluzione dei problemi tecnici.

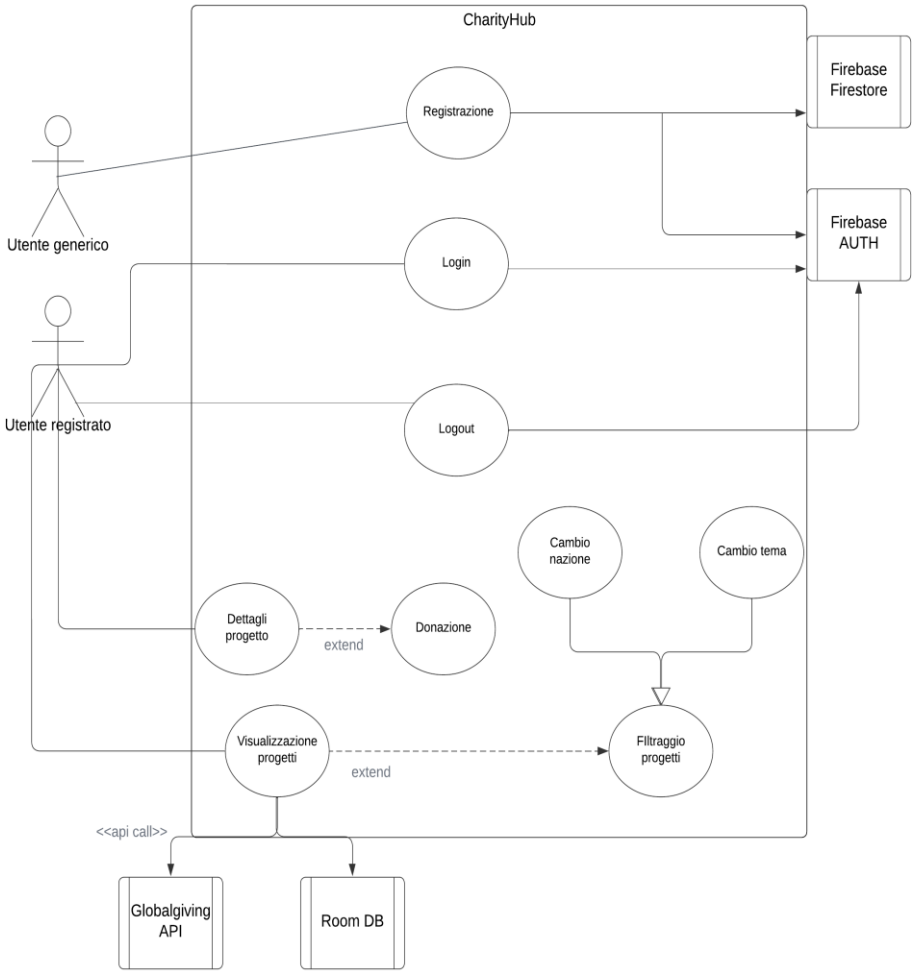
2 Analisi dei requisiti

I requisiti funzionali previsti per permettere il corretto e voluto utilizzo dell'applicazione sono i seguenti:

- Registrazione
- Login
- Logout
- Visualizzazione progetti
- Filtraggio dei progetti (per tema o nazione)
- Visualizzazione dettagli di un progetto
- Donazione

I requisiti non funzionali sono i seguenti:

- L'app deve garantire tempi di risposta rapidi
- Autenticazione e autorizzazione
- Il codice deve essere scritto in modo modulare per facilitare la manutenzione
- I test devono garantire il funzionamento delle classi critiche
- UI intuitiva e facile da usare
- L'applicazione deve essere offline first



2.1 Caso d'uso Registrazione

	Caso d'uso Registrazione
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente non registrato
Parti interessate	Utente non registrato: vuole registrarsi
Pre-condizioni	Nessuna
Garanzia di successo	L'utente è registrato e viene salvato nel DB remoto di Firestore
Scenario principale di successo	1. L'utente non registrato inserisce il nome 2. L'utente non registrato inserisce l'e-mail 3. L'utente non registrato inserisce la password 4. L'utente non registrato inserisce la nazione di interesse 5. L'utente non registrato seleziona "Registrati" 6. Il sistema conferma la registrazione e cambia Activity
Estensioni	2a) L'email inserita non è valida 2a.1 Il sistema avverte l'utente che il formato dell'e-mail non è corretto 2a.2 L'utente reinserisce l'e-mail 2b) L'email inserita è già esistente 2b.1 Il sistema avverte l'utente che l'e-mail inserita è già registrata 2b.2 L'utente cambia e-mail 3a) La password inserita non rispetta i requisiti minimi 3a.1 Il sistema avverte l'utente che la password inserita non rispetta i requisiti minimi 3a.2 L'utente reinserisce la password
Requisiti speciali	nessuno
Elenco delle varianti tecnologiche e dei dati	nessuno
Frequenza di ripetizione	Ogni volta che un utente registrato vuole fare un nuovo account

2.2 Caso d'uso Login

	Caso d'uso Login
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente registrato
Parti interessate	Utente registrato: vuole fare login
Pre-condizioni	L'utente deve essere registrato
Garanzia di successo	L'utente viene autenticato ed entra nell'applicazione
Scenario principale di successo	1. L'utente inserisce la mail 2. L'utente inserisce la password 3. L'utente clicca "Login" 4. Il sistema autentica l'utente e cambia Activity
Estensioni	1a) L'email inserita non è valida 1a.1 Il sistema avverte l'utente che il formato dell'e-mail non è corretto 1a.2 L'utente reinserisce l'e-mail 1b) L'email inserita non è esistente o la password è sbagliata 1b.1 Il sistema avverte l'utente che e-mail o password non sono corrette 1b.2 L'utente corregge i dati 2a) L'email inserita non è esistente o la password è sbagliata 2a.1 Il sistema avverte l'utente che e-mail o password non sono corrette 2a.2 L'utente corregge i dati
Requisiti speciali	nessuno
Elenco delle varianti tecnologiche e dei dati	nessuno
Frequenza di ripetizione	Ogni volta che un utente registrato vuole entrare nell'applicazione

2.3 Caso d'uso Logout

	Caso d'uso Logout
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente registrato
Parti interessate	Utente registrato: vuole fare logout
Pre-condizioni	L'utente deve essere autenticato
Garanzia di successo	L'utente effettua il logout ed esce dall'applicazione
Scenario principale di successo	1. L'utente clicca il pulsante di logout 2. Il sistema rimuove la nazione di preferenza salvata in locale 3. Il sistema rimuove i progetti salvati in locale 4. Il sistema torna alla WelcomeActivity
Estensioni	Nessuna
Requisiti speciali	Nessuno
Elenco delle varianti tecnologiche e dei dati	Nessuno
Frequenza di ripetizione	Ogni volta che un utente autenticato vuole uscire dall'applicazione

Registrazione, Login e Logout sono funzionalità ad alta priorità e necessarie per permettere non solo la possibilità di filtrare i progetti ma anche per facilitare future estensioni (quali salvataggio di pagamento, progetti preferiti, ecc…). Sono state implementate correttamente.

2.4 Caso d'uso Visualizzazione Progetti

	Caso d'uso Visualizzazione Progetti
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente registrato
Parti interessate	Utente registrato: vuole vedere i progetti
Pre-condizioni	L'utente deve essere registrato
Garanzia di successo	Il sistema mostra i progetti disponibili
Scenario principale di successo	1. L'utente entra nell'applicazione 2. I progetti vengono visualizzati 3. L'utente scrolla la pagina
Estensioni	2a) L'utente non ha progetti salvati in locale 2a.1 Il sistema fa una chiamata a GlobalGivingAPI con la nazione di interesse selezionata 2a.2 Il sistema scarica in locale i progetti recuperati dalla chiamata API 2b) L'utente ha progetti salvati in locale con la nazione scelta 2b.1 Il sistema mostra i progetti salvati in locale 3a) L'utente arriva alla fine dei progetti scaricati 3a.1 Il sistema fa una nuova chiamata a GlobalGivingAPI 3.a.1.1 Il sistema mostra i nuovi progetti 3.a.2.1 I progetti disponibili sono terminati 3.a.2.2 Il sistema mostra un messaggio all'utente
Requisiti speciali	nessuno
Elenco delle varianti tecnologiche e dei dati	nessuno
Frequenza di ripetizione	Sempre

Questa funzionalità ha priorità massima ed è stata implementata completamente.

2.5 Caso d'uso Filtraggio Progetti

	Caso d'uso Filtraggio Progetti
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente registrato
Parti interessate	Utente registrato: vuole filtrare i progetti
Pre-condizioni	L'utente deve essere registrato
Garanzia di successo	Il sistema mostra i progetti disponibili con i filtri corretti
Scenario principale di successo	1. L'utente entra nell'applicazione 2. L'utente seleziona un tema o nazione di interesse 3. I progetti vengono visualizzati 4. L'utente scrolla la pagina
Estensioni	3a) L'utente non ha progetti salvati in locale 2a.1 Il sistema fa una chiamata a GlobalGivingAPI con la nazione di interesse selezionata e tema scelto 2a.2 Il sistema scarica in locale i progetti recuperati dalla chiamata API 3b) L'utente ha progetti salvati in locale con la nazione scelta e tema selezionato 3b.1 Il sistema mostra i progetti salvati in locale 4a) L'utente arriva alla fine dei progetti scaricati 4a.1 Il sistema fa una nuova chiamata a GlobalGivingAPI con nazione e tema selezionati 4.a.1.1 Il sistema mostra i nuovi progetti 4.a.2.1 I progetti disponibili sono terminati 4.a.2.2 Il sistema mostra un messaggio all'utente
Requisiti speciali	nessuno
Elenco delle varianti tecnologiche e dei dati	nessuno
Frequenza di ripetizione	Sempre

La possibilità di filtrare progetti, seppur non obbligatoria è di particolare importanza per fornire un'esperienza utente piacevole, la funzionalità è stata implementata completamente.

2.6 Caso d'uso Visualizzazione Dettagli Progetto

	Caso d'uso Visualizzazione Dettagli Progetto
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente registrato
Parti interessate	Utente registrato: vuole visualizzare i dettagli di un progetto
Pre-condizioni	L'utente deve essere registrato
Garanzia di successo	Il sistema mostra le informazioni di un progetto
Scenario principale di successo	1. L'utente visualizza i progetti 2. L'utente clicca su "Mostra dettagli" 3. Il sistema scarica le immagini di un progetto 4. Il sistema mostra le informazioni di un progetto
Estensioni	3a) Il sistema non trova immagini disponibili del progetto 3a.1 Viene visualizzata un'immagine placeholder
Requisiti speciali	Nessuno
Elenco delle varianti tecnologiche e dei dati	nessuno
Frequenza di ripetizione	Sempre

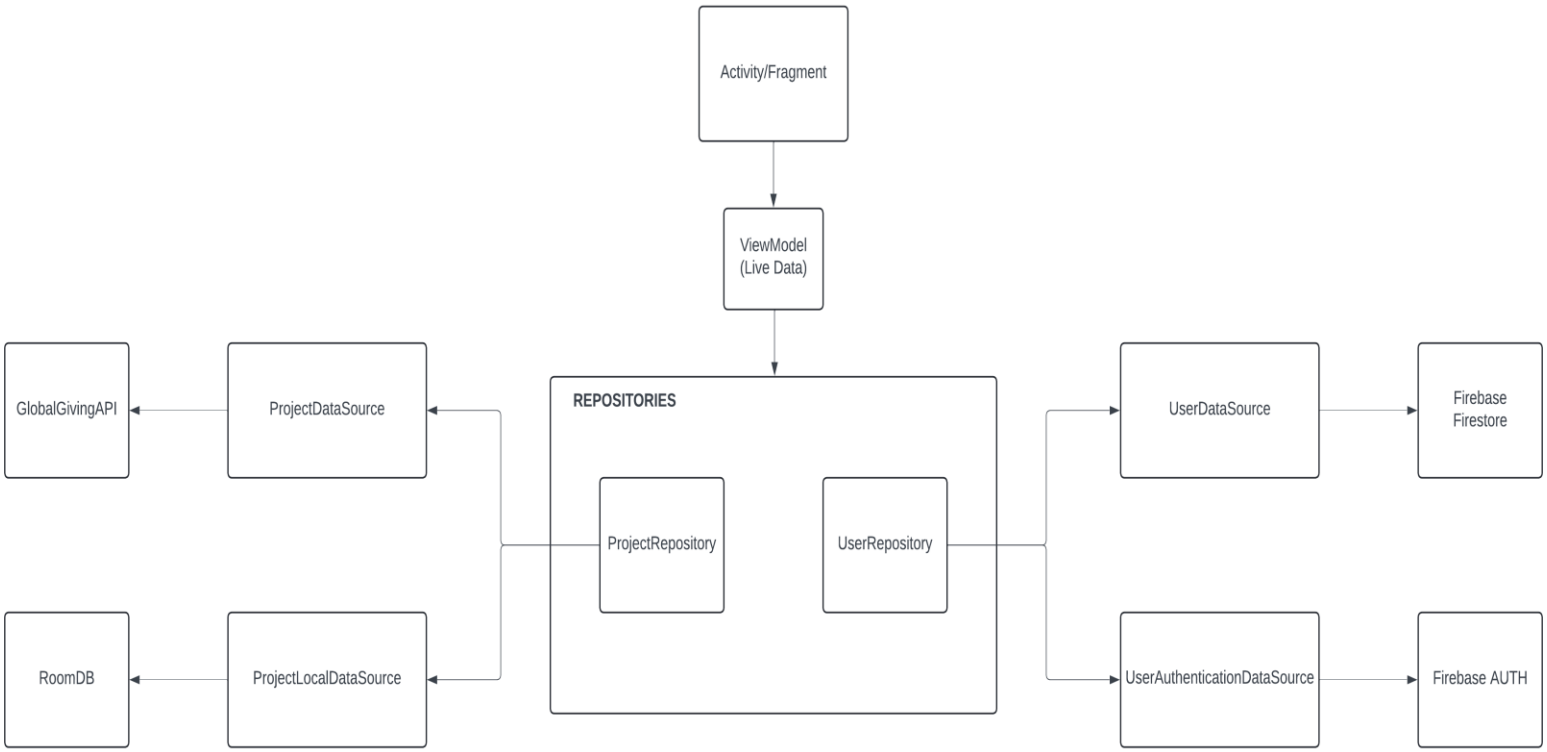
Anche questo caso d'uso come la visualizzazione dei progetti ha priorità massima ed è stato implementato correttamente.

2.7 Caso d'uso Donazione

	Caso d'uso Donazione
Portata	CharityHub
Livello	Obiettivo utente
Attore primario	Utente registrato
Parti interessate	Utente registrato: vuole effettuare una donazione
Pre-condizioni	L'utente deve essere registrato
Garanzia di successo	L'utente effettua con successo una donazione
Scenario principale di successo	1. L'utente visualizza i dettagli di un progetto 2. L'utente seleziona "Supporta il progetto" 3. Il sistema mostra i pacchetti di donazioni 4. L'utente seleziona un pacchetto 5. L'utente inserisce i dati di pagamento 6. Il sistema riceve il pagamento 7. Il sistema invia all'utente la conferma di pagamento
Estensioni	6a) Il pagamento non va a buon fine 6a.1 Il sistema mostra un messaggio di errore all'utente 6a.2 L'utente inserisce di nuovo i dati di pagamento
Requisiti speciali	Nessuno
Elenco delle varianti tecnologiche e dei dati	nessuno
Frequenza di ripetizione	Sempre

Il requisito funzionale donazione non è stato effettivamente implementato a causa della necessità di certificati e processi legali; tuttavia, questo scenario dovrebbe rispecchiare la realtà in modo accurato.

3 Implementazione



L'architettura è composta da:

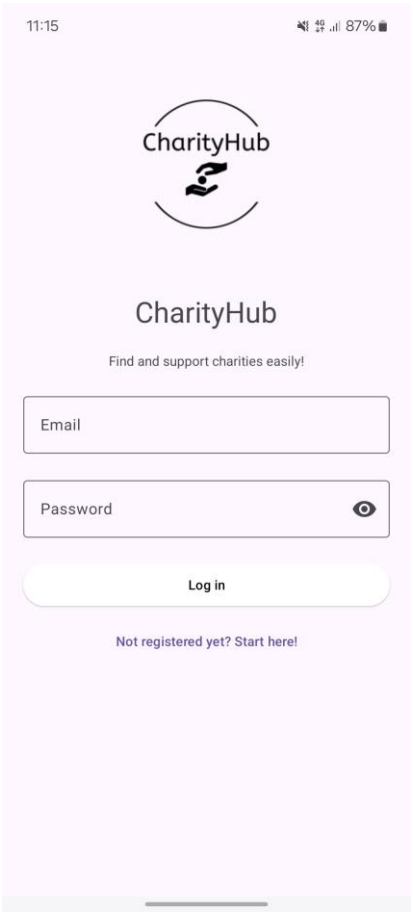
- Activity/Fragment, compongono l'UI
- ViewModel, dove vivono i live data, fanno da intermediari tra UI e Repository
- Repository, ulteriore livello di astrazione che comunicano con i data source, facendo da intermediari con le view model
- Data sources, prelevano i dati da una fonte

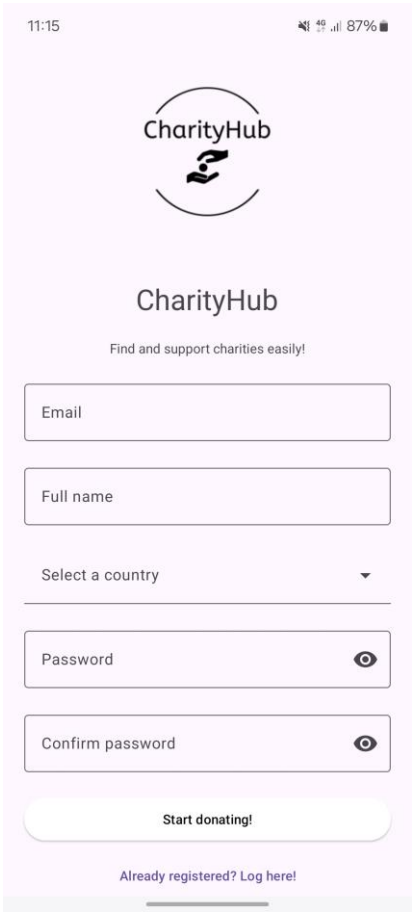
3.1 UI

L'interfaccia utente viene realizzata secondo gli standard di Material3 e utilizzando le view da esso fornite. Lo stile dell'app è semplicistico e i colori utilizzati vogliono trasmettere tranquillità e professionalità all'utente, che può godere di una navigazione chiara e diretta.

3.1.1 Welcome Activity

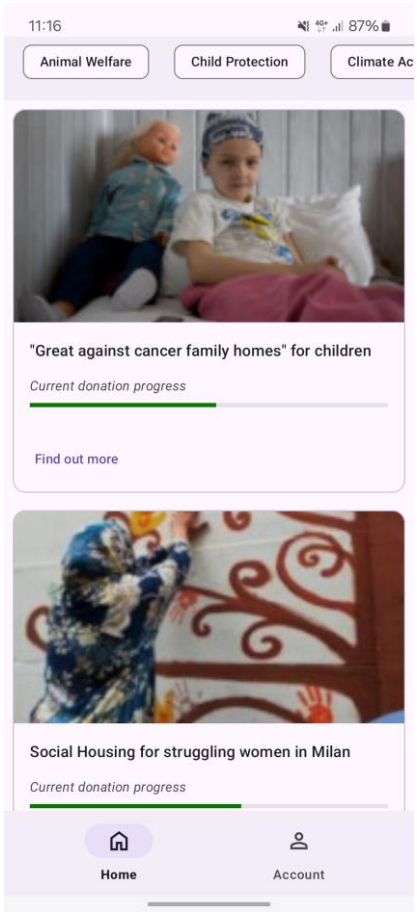
Il LoginFragment e il RegistrationFragment sono le prime interfacce visualizzate quando l'utente apre l'applicazione per la prima volta o dopo un logout. L'applicazione richiede l'inserimento dei dati per l'autenticazione o la registrazione. Durante la registrazione, è necessario inserire il nome e la nazione di interesse, così da poter mostrare i progetti nella località scelta. Questi fragment comunicano con il UserViewModel, che a sua volta interagisce con il UserRepository per fornire il sistema di autenticazione (Firebase AUTH) e per salvare o interrogare il database remoto (Firebase Firestore).



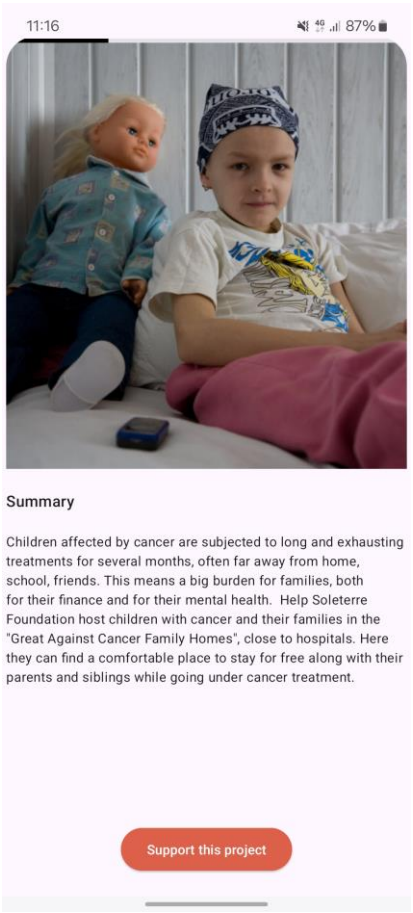


3.1.2 Main Activity

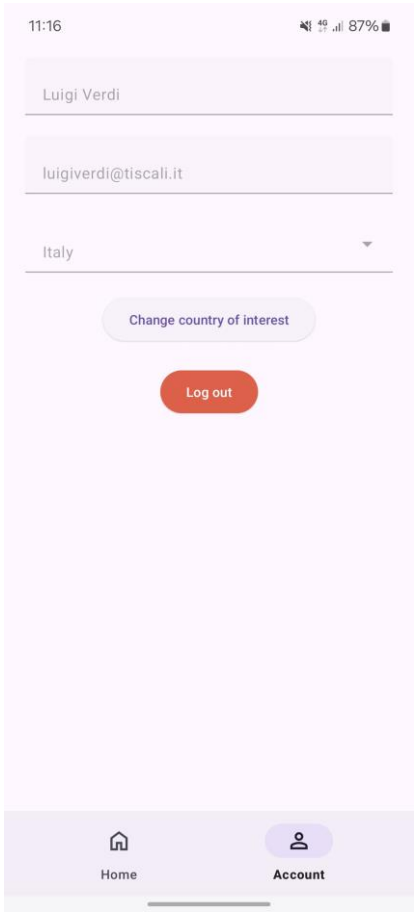
HomeFragment è la pagina principale dell'applicazione, dove vengono mostrati i progetti della nazione selezionata. Gli utenti possono filtrare i progetti per tema utilizzando le chips situate nella parte superiore dello schermo. I dati dei progetti provengono da due fonti diverse: una locale e una remota. Se l'utente non ha progetti salvati localmente, o se questi sono già stati visualizzati nella RecyclerView del HomeFragment, verrà effettuata una chiamata alla GlobalGivingAPI con i filtri appropriati. Una volta ricevuti, i progetti saranno salvati nel database locale per le chiamate future. Il fragment è composto da una RecyclerView con un adapter personalizzato per i progetti. Per la navigazione verso il fragment account, è presente una bottom navigation bar.



Il ProjectDetailsFragment e il DonationFragment mostrano rispettivamente le informazioni dettagliate di un progetto, incluse immagini aggiuntive, e le opzioni di donazione per supportare il progetto selezionato. Il ProjectDetailsFragment comunica con il ProjectDetailsViewModel per scaricare le immagini di un progetto utilizzando il suo ID. Se le immagini non sono disponibili, o se l'utente non è collegato alla rete, verrà mostrata un'immagine placeholder con il logo dell'applicazione. Per visualizzare le varie informazioni di un progetto, viene utilizzato un FullScreenCarousel a scorrimento orizzontale, con ciascuna pagina contenente un'immagine e un capitolo. Cliccando su "Support this project", si accede al DonationFragment, che contiene le opzioni di donazione scaricate in precedenza. Le opzioni di donazione sono visualizzate in una RecyclerView con un adapter personalizzato, contenente card selezionabili.



Infine, l'AccountFragment permette all'utente di cambiare la nazione di suo interesse ed effettuare il logout. Quando viene cambiato lo stato di preferenza, viene aggiornato sia il campo "countryOfInterest" in Firebase Firestore, sia il file locale salvato con SharedPreferencesUtil. Al ritorno nell'HomeFragment, i progetti verranno aggiornati automaticamente. Il pulsante "Change country of interest" abilita l'AutoCompleteTextView per la selezione delle nazioni, mostrando anche i pulsanti "Undo" per annullare la modifica e "Confirm" per confermarla. Se la nazione selezionata è identica a quella precedentemente salvata, non verrà eseguita alcuna query aggiuntiva. Il pulsante "Log out" permette di uscire dall'applicazione, eliminando tutti i progetti salvati in locale, e riportando l'utente al LoginFragment.



3.2 Classi Java

A seguire un elenco con una breve descrizione delle classi Java utilizzate all'interno del progetto:

Adapter:

- Donation Adapter: adapter per la recycler view del DonationFragment, contiene le card per i pacchetti di donazione
- Image Adapter: adapter per la recycler view del ProjectDetailsFragment, contiene le immagini di un progetto da mostrare nel carosello
- Project Adapter: adapter per la recycler view del HomeFragment, contiene le card per le informazioni di un progetto

Data:

- Database:
  - ProjectDAO: interfaccia per eseguire query al database locale
  - ProjectsRoomDatabase: implementazione del database locale di Room
- Repositories:
  - Project:
    - ProjectRepository: repository per ottenimento dei progetti da ProjectDataSource e ProjectLocalDataSource
  - User:
    - UserRepository: repository per interfacciarsi con UserDataSource e UserAuthenticationDataSource
- Service:
  - ProjectApiService: interfaccia per eseguire chiamate API utilizzando Retrofit
- Source:
  - Projects:
    - ProjectDataSource: classe che utilizza ProjectApiService per eseguire chiamate a GlobalGivingAPI
    - ProjectLocalDataSource: classe che salva e preleva progetti dal database locale
    - ProjectCallBack: interfaccia implementata da ProjectRepository e utilizzata dalle DataSource per chiamare i callback
  - User:

        - UserAuthenticationDataSource: classe che comunica con Firebase AUTH per permettere le operazioni di login, registrazione e logout
        - UserDataSource: classe che comunica con Firebase Firestore per permettere di salvare e interrogare il database remoto contenente informazioni sugli utenti

Model:

- Countries:
    - Country: classe che definisce la struttura di una nazione (codice, nome)
- Projects:
    - DonationOptions: classe che definisce la struttura della risposta json quando vengono richieste le opzioni di donazione di un progetto
    - Image: classe che definisce la struttura di un'immagine (url, dimensione) per la risposta json di ImagesApiResponse
    - ImagesApiResponse: classe che definisce la struttura della risposta json quando vengono richieste le immagini relative a un progetto
    - Project: classe che definisce la struttura di un progetto per la risposta json di ProjectApiResponse
    - ProjectApiResponse: classe che definisce la struttura della risposta json quando viene richiesto un progetto
    - Theme: classe che definisce la struttura di un tema (codice, nome) per la risposta json di ThemesApiResponse
    - ThemesApiResponse: classe che definisce la struttura della risposta json quando vengono richieste i temi
- Result: classe utilizzata durante l'utilizzo dei MutableLiveData per definire se un'operazione relativa al prelevamento di dati sia andata a buon fine
- User: classe che definisce la struttura di un utente

Utils:

- Constants: classe contenitore per le varie costanti utilizzate
- Converters: classe che converte le opzioni di donazione di un progetto in un formato accettabile per essere salvato in locale come attributo
- SharedPreferencesUtil: classe utilizzata per salvare il codice della nazione di un utente in locale

3.3 Flusso di dati

Per permettere di gestire il contenuto dell'applicazione, ovvero lettura e scrittura di progetti benefici e informazioni a loro associate e nazione di interesse vengono utilizzati questi tre elementi fondamentali:

3.3.1 GlobalGiving API

Come già menzionato, per ricevere i progetti vengono utilizzate le API fornite da Global Giving. Il sito fornisce non solo una lista di iniziative benefiche, ma anche informazioni ad esso associate, quali obbiettivi di un progetto, spiegazioni su come verranno utilizzati le eventuali donazioni, progressi attuali e immagini associate. La chiamata API viene effettuata in una sola occasione, ovvero all'interno dell'HomeFragment, dove vengono scaricate tutte le informazioni necessarie per poi essere visualizzate nella sezione dettagliata di un progetto, permettendo all'utente una navigazione veloce.

3.3.2 RoomDB

Room è il database locale fornito da Android, il suo utilizzo nell'applicazione è fondamentale per garantire il requisito non funzionale "Offline First". Quando il sistema effettua una chiamata a GlobalGiving API, dopo che il prelevamento dei progetti è andato a buon fine, questi vengono salvati su Room. Se un utente all'apertura dell'HomeFragment possiede dei progetti scaricati con i filtri appropriati, questi verranno visualizzati per prima, e se, allo scrollare della recycler view arriva in fondo ai progetti disponibili, viene effettuata una chiamata API. Dopo un logout il database viene pulito completamente.

3.3.3 Shared Preferences

Shared Preferences è un ulteriore API fornita da Android per il salvataggio in locale di valori più semplici, in questo caso una stringa. Alla registrazione di un nuovo account viene creato il file dove viene salvato il codice della nazione selezionata, che verrà poi letta nell'HomeFragment. Dopo il login invece, sul fine viene scritta la nazione di preferenza salvata sul DB remoto, che viene scritta e modificata sempre in contemporanea con quella locale, garantendo che il valore della stringa salvata in locale sia sempre lo stesso di quella in remoto

3.4 Strumenti utilizzati

Segue un elenco dei principali strumenti utilizzati per lo sviluppo di CharityHub:

- Android Studio, IDE per le applicazioni Android
- Android Developers, documentazione per sviluppo Android
- Firebase Firestore
- Firebase AUTH
- GitHub
- Stack Overflow
- Material Design 3, come punto di riferimento per le scelte della UI
- ChatGPT 3.5

4 Progetto di testing

4.1 Framework utilizzati

L'architettura ben strutturata di un'applicazione facilita notevolmente il processo di testing, rendendo più semplice garantire la qualità e il corretto funzionamento del software. Per assicurare che CharityHub funzioni correttamente, sono state sottoposte a test approfonditi le classi più critiche e importanti, nonché le interfacce utente per imitare il comportamento di un utente reale. I principali framework utilizzati in questo processo sono Roboelectric, Espresso e Mockito.

Roboelectric è un potente framework che fornisce un dispositivo simulato, consentendo di eseguire test in un ambiente più rapido rispetto a un emulatore Android standard. Questo strumento è particolarmente utile per eseguire test UI e verificare il comportamento dei fragment, permettendo di testare i componenti Android in un ambiente isolato. Grazie a Roboelectric, è possibile eseguire test di integrazione e unitari su classi che altrimenti richiederebbero l'uso di un dispositivo fisico o di un emulatore, migliorando così l'efficienza e la velocità del ciclo di sviluppo.

Espresso è un framework specificamente progettato per il testing delle interfacce utente. Fornisce un ampio set di API che permettono di simulare interazioni dell'utente, come click, swipe e inserimento di testo, e di fare assertions sulle view di Android. Questo consente di verificare che l'interfaccia utente reagisca correttamente alle interazioni, assicurando un'esperienza utente fluida e priva di errori.

Mockito è uno strumento essenziale per la creazione di oggetti simulati (mock) che riproducono il comportamento di oggetti reali. È utilizzato prevalentemente per testare repositories e view model, rendendo il framework particolarmente utile per gli unit test. Mockito permette di isolare una classe e assicurarsi che i test si concentrino esclusivamente sul comportamento della classe stessa, senza interferenze esterne. Questo approccio consente di individuare rapidamente bug e difetti, migliorando la stabilità del codice e riducendo il rischio di regressioni.

Features di Roboelectric:

- Utilizza la JVM, evita il bisogno di utilizzare un emulatore per testare
- Velocità, i test sono molto più veloci
- Accesso al framework di Android, per permettere di testare Activities

- Integrazione con JUnit, completa compatibilità con JUnit

Features di Espresso:

- Sincronizzazione automatica tra il codice di testing e il thread dell'UI
- Inclusione di metodi per testare la visualizzazione e interazione con view all'interno dell'applicazione
- Integrazione con AndroidJUnitRunner

Features di Mockito:

- Creazione di oggetti simulati per replicare il comportamento di oggetti reali in modo controllato
- Possibilità di definire il comportamento degli oggetti simulati, specificando cosa dovrebbero ritornare o come dovrebbero agire
- Possibilità di verificare interazioni con gli oggetti simulati, assicurando che metodi specifici vengano chiamati con gli argomenti giusti

4.2 Test eseguiti

4.2.1 Project Repository Test

La prima classe ad essere testata è la ProjectRepository, fondamentale per il funzionamento dell'applicazione. Come già accennato, ProjectRepository gestisce la ricezione dei dati da due Data Source. Gli unit test per questa classe verificano che vengano chiamati i metodi corretti e che le interazioni con ProjectDataSource e ProjectLocalDataSource avvengano come previsto.
Dettagli dell'Implementazione:

- Setup:
    - Prima dell'esecuzione di ogni test, il metodo setUp inizializza i mock per BaseProjectDataSource e BaseProjectLocalDataSource, e crea un'istanza della ProjectRepository.
    - Vengono anche inizializzate le variabili projectsLiveData, themesLiveData, e imagesLiveData, che rappresentano le LiveData utilizzate per aggiornare l'interfaccia utente con i risultati delle operazioni di caricamento dati.
- Test dei Metodi:
    - searchForProjects: Verifica che il metodo searchForProjects invochi correttamente il metodo getProjectsByCountry del projectLocalDataSource.

- getProjectsByTheme: Controlla che il metodo getProjectsByTheme richiami getProjectsByTheme del projectLocalDataSource.
- getProjectsByCountry: Assicura che getProjectsByCountry richiami getProjectsByCountry del projectLocalDataSource.
- searchProjects: Verifica che searchProjects invochi searchForProjects del projectDataSource.
- getDownloadedProjects: Controlla che getDownloadedProjects richiami getProjects del projectLocalDataSource.
- getThemesLiveData: Assicura che getThemesLiveData invochi getThemes del projectDataSource.
- getImagesLiveData: Verifica che getImagesLiveData richiami getImages del projectDataSource.
- onProjectsLoaded: Verifica che, una volta caricati i progetti, il metodo insertProjects del projectLocalDataSource venga chiamato correttamente.
- onFailureFromRemote: Assicura che, in caso di fallimento remoto, venga postato un errore su projectsLiveData.
- onThemesLoaded: Verifica che, una volta caricate le tematiche, un successo venga postato su themesLiveData.
- onFailureThemesLoaded: Controlla che in caso di fallimento del caricamento delle tematiche venga postato un errore su themesLiveData.
- onSuccessImagesLoaded: Assicura che, una volta caricate le immagini, un successo venga postato su imagesLiveData.
- onFailureImagesLoaded: Verifica che in caso di fallimento del caricamento delle immagini venga postato un errore su imagesLiveData.
- onLocalSuccess: Verifica che, in caso di successo locale, un successo venga postato su projectsLiveData.
- onProjectsLocalSuccess: Assicura che, in caso di successo nel caricamento locale dei progetti, un successo venga postato su projectsLiveData

Questa test suite copre tutti i metodi della ProjectRepository, assicurandone il corretto funzionamento.

4.2.2 User Repository Test

Similmente ai test per ProjectRepository, questa test suite si occupa di verificare le interazioni e comportamenti con UserDataSource e UserAuthenticationDataSource.
Dettagli dell'Implementazione:

- Setup:
  - Il metodo setUp inizializza i mock per BaseUserAuthenticationDataSource, BaseUserDataRemoteDataSource e BaseProjectLocalDataSource utilizzando MockitoAnnotations.openMocks(this).
  - Viene creata un'istanza di UserRepository e viene iniettata la MutableLiveData<Result> mockata.
- Test dei Metodi:
  - getUserLiveData_RegisteredUser: Verifica che, quando un utente registrato tenta di accedere, il metodo logIn di userAuthenticationDataSource venga chiamato con le credenziali appropriate.
  - getUserLiveData_NewUser: Controlla che, quando un nuovo utente tenta di registrarsi, il metodo signIn di userAuthenticationDataSource venga chiamato con le informazioni appropriate.
  - getLoggedUser: Assicura che il metodo getLoggedUser restituisca l'utente attualmente loggato, verificando che il risultato sia uguale all'utente atteso.
  - logOut: Verifica che il metodo logOut di userAuthenticationDataSource venga chiamato correttamente.
  - changeUserCountry: Controlla che il metodo changeUserCountry di userDataRemoteDataSource venga chiamato con l'utente fornito.
  - onSuccessAuthentication: Verifica che, in caso di autenticazione avvenuta con successo, il metodo getUserCountry di userDataRemoteDataSource venga chiamato con l'utente autenticato.
  - onFailureAuthentication: Assicura che, in caso di fallimento dell'autenticazione, venga postato un errore su userLiveData.
  - onSuccessLogout: Verifica che, in caso di logout avvenuto con successo, tutti i dati dei progetti locali vengano cancellati e un successo venga postato su userLiveData con l'utente impostato su null.
  - onSuccessUserSaved: Controlla che, una volta salvato con successo un utente, un successo venga postato su userLiveData con l'utente salvato.
  - onSuccessRegistration: Verifica che, in caso di registrazione avvenuta con successo, il metodo saveUserData di userDataRemoteDataSource venga chiamato con l'utente registrato.

- onCountryGotSuccess: Assicura che, una volta ottenuto con successo il paese dell'utente, un successo venga postato su userLiveData con l'utente aggiornato.
- onSuccessInfoChanged: Verifica che, una volta modificate con successo le informazioni dell'utente, un successo venga postato su userLiveData con l'utente aggiornato.

Anche questa test suite offre una copertura del 100% dei metodi e verificando la consistenza dei dati durante le comunicazioni con le data source.

4.2.3 Home ViewModel Test

Questa classe di test garantisce che i dati vengano recuperati in modo appropriato e che gli stati della viewmodel vengano aggiornati come previsto.
Dettagli dell'implementazione:

- Setup:
  - Il metodo setUp inizializza i mock per ProjectRepository utilizzando MockitoAnnotations.openMocks(this).
  - Viene creata un'istanza di HomeViewModel e viene associata al ProjectRepository mockato.
- Test dei Metodi:
  - testSearchForProjects: Verifica che il metodo searchForProjects richiami correttamente il metodo corrispondente del ProjectRepository e che i dati ritornati siano quelli attesi.
  - testGetDownloadedProjects: Controlla che il metodo getDownloadedProjects del ProjectRepository venga chiamato correttamente e che i dati dei progetti scaricati siano gestiti correttamente dal ViewModel.
  - testGetThemesLiveData: Verifica che il metodo getThemesLiveData richiami il metodo corrispondente del ProjectRepository e che i dati delle tematiche siano correttamente gestiti e osservati.
  - testSetAndGetCurrentTheme: Assicura che il tema corrente possa essere impostato e recuperato correttamente dal ViewModel.
  - testSetAndIsLoading: Verifica che lo stato di caricamento possa essere impostato e recuperato correttamente dal ViewModel.
  - testSetAndIsFirstLoading: Controlla che lo stato di primo caricamento possa essere impostato e recuperato correttamente dal ViewModel.

- testSetAndIsNoMoreProjects: Assicura che lo stato di "no more projects" possa essere impostato e recuperato correttamente dal ViewModel.
- testSearchProjects: Verifica che il metodo searchProjects del ProjectRepository venga chiamato correttamente quando si cerca un progetto con determinati filtri.
- testGetProjectsByTheme: Controlla che il metodo getProjectsByTheme del ProjectRepository venga chiamato correttamente quando si cercano progetti per tematica.
- testGetProjectsByCountry: Assicura che il metodo getProjectsByCountry del ProjectRepository venga chiamato correttamente quando si cercano progetti per paese.

Per testare correttamente l'HomeViewModel, è fondamentale gestire correttamente i MutableLiveData. I metodi che ricevono live data da ProjectRepository vengono testati nel seguente modo:

1. Vengono creati mock degli oggetti necessari, come ad esempio una lista di progetti vuoti.
2. Viene creato e inizializzato un expectedLiveData.
3. Viene creato il result appropriato utilizzando gli oggetti mock (ad esempio, Result.ProjectResponseSuccess).
4. Il result viene assegnato all'expectedLiveData.
5. Utilizzando il metodo when() di Mockito, si specifica che quando il metodo della repository viene chiamato, deve ritornare l'expectedLiveData creato.
6. Si esegue l'effettiva chiamata con parametri arbitrari.
7. Viene creato un observer che ottiene il risultato effettivo della chiamata e confronta i dati dell'expectedLiveData con quelli reali.
8. Infine, si verifica che il metodo della repository venga chiamato con gli argomenti corretti.

Questa procedura assicura che i MutableLiveData vengano gestiti e testati accuratamente, garantendo che l'HomeViewModel funzioni correttamente con i dati forniti dalla ProjectRepository. Anche questa test suite copre completamente la classe HomeViewModel.

4.2.4 User ViewModel Test

Gli obbiettivi di questi test sono analoghi a quelli della classe HomeViewModelTest.
Dettagli dell'Implementazione

- Setup:

    - Il metodo setUp inizializza i mock per UserRepository e CountryRepository utilizzando MockitoAnnotations.openMocks(this).
    - Viene creata un'istanza di UserViewModel e viene associata ai repository mockati.
- Test dei Metodi:
    - testGetUserLiveData: Verifica che il metodo getUserLiveData richiami correttamente il metodo corrispondente del UserRepository con i parametri corretti e che i dati dell'utente siano gestiti correttamente dal ViewModel.
    - testLogout: Controlla che il metodo logOut del UserRepository venga chiamato correttamente e che lo stato dell'utente venga impostato su null nel LiveData.
    - testGetCountriesLiveData: Verifica che il metodo getCountriesLiveData richiami correttamente il metodo corrispondente del CountryRepository e che i dati dei paesi siano gestiti correttamente dal ViewModel.
    - testChangeUserCountry: Assicura che il metodo changeUserCountry del UserRepository venga chiamato con l'utente fornito e che i dati aggiornati dell'utente siano gestiti correttamente dal ViewModel.
    - testGetLoggedUser: Controlla che il metodo getLoggedUser del UserRepository restituisca l'utente attualmente loggato e che il risultato sia corretto.

Il testing dei livedata è identico a come descritto in precedenza. La copertura è del 100%.

4.2.5 Login Fragment Test

Questa classe testa l'utilizzo del LoginFragment e prova a replicare il comportamento di un utente durante il login. Vengono utilizzati Espresso e Roboelectric.
Dettagli dell'Implementazione

- Setup:
    - Utilizza ActivityScenarioRule per lanciare WelcomeActivity prima di ogni test.
    - Inizializza il contesto dell'applicazione e imposta il tema dell'attività per garantire che i test abbiano l'aspetto e il comportamento corretti.
- Test dei Metodi:
    - testWrongEmailFormat: Verifica che, quando viene inserito un indirizzo e-mail in un formato errato, venga visualizzato un messaggio

di errore appropriato. Esegue azioni di input sul campo e-mail e password, e clicca sul pulsante di login.

- testLoginSuccess: Verifica che, quando vengono inserite credenziali valide, venga mostrato un indicatore di progresso (CircularProgressIndicator). Inserisce un'e-mail e una password valide e clicca sul pulsante di login.
- testLoginFailure: Verifica che, quando vengono inserite credenziali non valide, venga mostrato un messaggio di errore appropriato. Inserisce un'e-mail e una password non valide e clicca sul pulsante di login.
- navigateToRegistrationFragment: Verifica che, quando l'utente clicca sul pulsante per registrarsi, la navigazione passi correttamente al RegistrationFragment. Utilizza un TestNavHostController per testare la navigazione.

4.2.6 Registration Fragment Test

L'obiettivo principale di RegistrationFragmentTest è assicurarsi che le varie operazioni di registrazione, la gestione degli errori e la navigazione nel RegistrationFragment funzionino correttamente e forniscano l'esperienza utente prevista.
Implementazione:

- Setup:
  - Utilizza TestNavHostController per simulare la navigazione all'interno del RegistrationFragment.
  - Inizializza il contesto dell'applicazione e imposta il controller di navigazione per garantire che i test abbiano l'aspetto e il comportamento corretti.
- Test dei Metodi:
  - testRegistrationSuccess: Verifica che, quando vengono inserite credenziali valide, la registrazione avvenga correttamente e la navigazione passi alla MainActivity. Inserisce un'e-mail, un nome completo, seleziona un paese dall'autocomplete, inserisce una password e conferma la password, e clicca sul pulsante di registrazione.
  - testPasswordMismatch: Verifica che, quando le password inserite non corrispondono, venga visualizzato un messaggio di errore appropriato. Inserisce un'e-mail, un nome completo, seleziona un paese, inserisce una password e una password di conferma diversa, e clicca sul pulsante di registrazione.

- testEmptyFields: Verifica che, quando i campi richiesti sono vuoti, vengano visualizzati messaggi di errore appropriati. Clicca sul pulsante di registrazione senza inserire alcun dato e controlla i messaggi di errore visualizzati.
- navigateToLoginFragment: Verifica che, quando l'utente clicca sul pulsante per accedere al login, la navigazione passi correttamente al LoginFragment. Utilizza un TestNavHostController per testare la navigazione.

4.2.7 Home Fragment Test

L'obiettivo principale di HomeFragmentTest è assicurarsi che le varie funzionalità del HomeFragment, come la visualizzazione dei progetti, la navigazione e l'interazione con i componenti dell'interfaccia utente, funzionino correttamente e forniscano l'esperienza utente prevista

Implementazione:

- Setup:
  - Utilizza FirebaseAuth per autenticare un utente di test prima di ogni test.
  - Inizializza il contesto dell'applicazione e configura un TestNavHostController per simulare la navigazione all'interno del HomeFragment.
  - Imposta il controller di navigazione per garantire che i test abbiano l'aspetto e il comportamento corretti.
- Test dei Metodi:
  - testChipGroupIsDisplayed: Verifica che il ChipGroup sia visualizzato correttamente sullo schermo. Utilizza un breve ritardo per assicurarsi che gli elementi dell'interfaccia utente siano caricati.
  - testRecyclerViewIsDisplayed: Verifica che la RecyclerView dei progetti sia visualizzata correttamente sullo schermo.
  - testNavigateToAccount: Verifica che, quando l'utente clicca sull'elemento di navigazione del fondo per l'account, la navigazione passi correttamente al AccountFragment. Include un breve ritardo per garantire che la navigazione sia completata.
  - testClickOnRecyclerViewItem: Simula lo scrolling a una posizione specifica nella RecyclerView e clicca su un pulsante all'interno di una scheda del progetto. Verifica che la navigazione passi correttamente al ProjectDetailsFragment.
- Utility:

- MyViewAction: Classe statica interna utilizzata per eseguire azioni personalizzate su viste specifiche all'interno di una RecyclerView. Fornisce un metodo per cliccare su una vista figlio con un determinato ID.

4.2.8 Account Fragment Test

L'obiettivo principale di AccountFragmentTest è assicurarsi che le varie funzionalità del AccountFragment, come la modifica del paese, la gestione degli stati dei pulsanti e la funzionalità di logout, funzionino correttamente e forniscano l'esperienza utente prevista.
Implementazione:

- Setup:
  - Utilizza FirebaseAuth per autenticare un utente di test prima di ogni test. Disabilita la verifica dell'app per il testing.
  - Inizializza il contesto dell'applicazione e configura un TestNavHostController per simulare la navigazione all'interno del AccountFragment.
  - Imposta il controller di navigazione per garantire che i test abbiano l'aspetto e il comportamento corretti.
- Test dei Metodi:
  - testChangeCountryButton: Verifica che, quando l'utente clicca sul pulsante di modifica (editButton), il campo per il paese (countryEdit) diventi abilitato e i pulsanti di conferma (confirmButton) e annullamento (undoButton) siano visualizzati.
  - testUndoButton: Verifica che, quando l'utente clicca sul pulsante di annullamento (undoButton), il campo per il paese (countryEdit) torni al valore originale. Simula l'inserimento di un nuovo paese e clicca sul pulsante di annullamento per controllare che il valore torni a "Italy".
  - testChangeCountry: Verifica che, quando l'utente cambia il paese e conferma la modifica, il campo per il paese venga aggiornato correttamente e torni ad essere non modificabile. Simula l'inserimento di un nuovo paese, la selezione dalla lista a discesa e la conferma della modifica.
  - logOutButton: Verifica che, quando l'utente clicca sul pulsante di logout (logoutButton), la navigazione torni all'WelcomeActivity. Controlla che l'ID della destinazione corrente del controller di navigazione sia R.id.welcomeActivity.

4.2.9 Shared Preferences Util Test

L'obiettivo principale di SharedPreferencesUtilTest è garantire che i metodi della classe SharedPreferencesUtil funzionino correttamente, interagiscano come previsto con SharedPreferences e SharedPreferences.Editor, e gestiscano i dati in modo appropriato.
Implementazione:

- Setup:
  - Il metodo setUp inizializza i mock per SharedPreferences, SharedPreferences.Editor e Context utilizzando MockitoAnnotations.openMocks(this).
  - Configura il comportamento del contesto mockato per restituire sharedPreferences quando viene chiamato getSharedPreferences.
  - Configura il comportamento di sharedPreferences per restituire editor quando viene chiamato edit.
  - Configura il comportamento di sharedPreferences per restituire una stringa di test ("testValue") quando viene chiamato getString.
  - Inizializza un'istanza di SharedPreferencesUtil utilizzando il contesto mockato.
- Test dei Metodi:
  - testWriteStringData: Verifica che il metodo writeStringData chiami correttamente putString e apply su SharedPreferences.Editor con i parametri forniti. Controlla che putString venga chiamato con la chiave "testKey" e il valore "testValue".
  - testReadStringData: Verifica che il metodo readStringData chiami correttamente getString su SharedPreferences con la chiave fornita e restituisca il valore atteso. Controlla che getString venga chiamato con la chiave "testKey" e il valore di default null.
  - testDeleteAll: Verifica che il metodo deleteAll chiami correttamente clear e apply su SharedPreferences.Editor.

4.3 Risultati dei test

In conclusione, i test eseguiti mirano a garantire il corretto funzionamento delle classi più critiche per l'operatività dell'applicazione. Gli unit test delle view model e dei repository coprono il 100% delle classi, mentre i test UI simulano il comportamento di un utente durante l'utilizzo dell'applicazione. È importante sottolineare che, a causa di problemi di compatibilità con Navigation Component e Material3, il tema utilizzato

durante i test differisce leggermente dall'interfaccia reale dell'applicazione. Tuttavia, questo non dovrebbe compromettere l'affidabilità dei test. Inoltre, per assicurare la correttezza dei test della pagina HomeFragment, i progetti visualizzati di default sono quelli provenienti dall'Italia, poiché non c'è un utente effettivamente collegato con una Nazione di preferenza.

5 Utilizzo di ChatGPT

Come già discusso in precedenza, ChatGPT 3.5 è stato utilizzato come strumento di supporto per lo sviluppo di CharityHub. Verranno discusse in modo dettagliato le tre fasi principali.

5.1 Requisiti

Durante la fase di definizione dei requisiti, l'intelligenza artificiale è stata utilizzata come strumento di supporto per facilitare la stesura degli obiettivi funzionali e non funzionali di CharityHub. In questa fase iniziale, ChatGPT è stato interrogato due volte, fornendo un leggero contesto sull'applicazione e sui suoi obiettivi principali. La risposta data ha prodotto una lista di requisiti che è servita come base di partenza per il progetto, tuttavia, la lista fornita necessitava di ulteriori adattamenti e ridimensionamenti per entrambi i tipi di requisiti. Questo perché ChatGPT, non avendo una conoscenza completa e dettagliata dell'intero contesto e della portata dell'applicazione, ha potuto offrire solo suggerimenti generici. Nello specifico, per quanto riguarda i requisiti funzionali, ChatGPT ha proposto una serie di funzionalità che l'applicazione potrebbe includere, questi suggerimenti hanno fornito un punto di partenza, ma sono stati soggetti a ridimensionamenti. Per i requisiti non funzionali, ha fornito indicazioni su aspetti quali le prestazioni, la sicurezza, l'usabilità e la scalabilità dell'applicazione, ma anche in questo caso, le indicazioni iniziali di ChatGPT hanno rappresentato un utile punto di partenza, richiedendo però ulteriori rifiniture.

5.2 Implementazione

La fase di implementazione è quella in cui ritengo che ChatGPT si sia rivelato più utile, rappresentando uno strumento prezioso per velocizzare il processo di sviluppo. In particolare, ho individuato due situazioni principali in cui l'intelligenza artificiale può essere utilizzata per programmare.

La prima situazione, più semplice e diretta, è quella in cui si richiede una funzionalità specifica o un pezzo di codice. A seconda della richiesta, la risposta può essere più o meno soddisfacente. La dimensione e la specificità della domanda influiscono particolarmente sulla qualità della risposta fornita da ChatGPT. Più una domanda è generale e più ChatGPT fatica a dare risposte utili e precise. Pertanto, l'utilizzo di richieste limitate e semplici si è rivelato più efficace per lo sviluppo. La tecnica migliore in questo contesto è scomporre un problema complesso in sottoproblemi più piccoli e

gestibili. Ad esempio, anziché chiedere come strutturare un'intera classe, è preferibile chiedere come scrivere un singolo metodo. Sebbene il codice generato da ChatGPT debba spesso essere ritoccato per adattarlo all'applicazione specifica, la base e il ragionamento forniti risultano comunque utili. È importante sottolineare che, essendo ChatGPT 3.5 una versione non recente, può capitare che alcuni metodi siano deprecati o che alcuni standard non vengano seguiti. È quindi fondamentale verificare che il codice fornito sia non solo funzionante, ma anche aggiornato.

La seconda situazione riguarda la risoluzione di problemi durante lo sviluppo. In questo caso, non si richiede a ChatGPT di fornire direttamente un codice per risolvere un problema specifico, ma piuttosto si chiedono motivazioni (ed eventualmente soluzioni) per un determinato errore riscontrato durante lo sviluppo. In questa circostanza, è di vitale importanza fornire un contesto quanto più adeguato possibile. Per contesto si intende l'errore riscontrato (sia esso un'eccezione o un comportamento indesiderato) e il codice interessato, non solo quello che genera direttamente l'errore. Tuttavia, fornire un contesto inutile può risultare controproducente: ChatGPT potrebbe segnalare e toccare codice che non influisce sull'errore incontrato. È quindi necessario prestare attenzione alla quantità di codice inviato e assicurarsi che sia pertinente.

Durante l'implementazione, ChatGPT è stato utilizzato sia per lo sviluppo dell'interfaccia utente sia per la progettazione delle funzionalità effettive dell'applicazione. Per quanto riguarda la generazione di codice per l'interfaccia grafica, si può dire che ChatGPT non sia particolarmente utile. Fornire una descrizione dettagliata e ottenere un codice XML corretto può risultare problematico e spesso insoddisfacente. Tuttavia, la situazione cambia quando si tratta di troubleshooting riguardante l'UI: in questi casi, fornendo un contesto adeguato, può offrire un aiuto significativo.

5.3 Testing

Per quanto riguarda l'utilizzo di ChatGPT per le test suite, ci sono da fare delle considerazioni specifiche. L'approccio e l'efficacia nell'uso dell'intelligenza artificiale variano in base al tipo di test che si desidera generare, inoltre quando si richiede la generazione di test, è importante specificare i framework da utilizzare e il tipo di test desiderato.

Per i test dell'interfaccia utente, è fondamentale comprendere i passaggi che un utente potrebbe eseguire all'interno di un fragment dell'applicazione. Di conseguenza, non è pratico inviare l'intera classe del fragment e chiedere la generazione di test completi. Un

approccio più efficace consiste nel suddividere la richiesta in parti più piccole e specifiche, ad esempio, si potrebbe chiedere di generare un test per lo scrolling di una recycler view.

Per quanto riguarda gli unit test delle repository e dei view model, è possibile inviare l'intera classe che si desidera testare poiché ChatGPT genera un test per ogni metodo trovato all'interno della classe. Tuttavia, gli unit test sono generalmente chiusi e limitati, e non si può inviare l'intera applicazione, di conseguenza, i test generati richiedono spesso pesanti modifiche, poiché è necessario verificare gli oggetti, i loro tipi e conoscere i risultati attesi. In particolare, l'intelligenza artificiale è risultata utile non tanto per la generazione delle effettive asserzioni dei test, ma più per la tecnica da utilizzare per testare i LiveData.

Ho osservato che la generazione di test è l'ambito in cui compaiono più frequentemente metodi deprecati forniti da ChatGPT, richiedendo un ulteriore livello di verifica e aggiornamento del codice generato. È quindi essenziale valutare ogni suggerimento fornito dall'intelligenza artificiale, assicurandosi che i metodi utilizzati siano attuali e conformi agli standard di sviluppo più recenti.

5.4 Statistiche

Come spiegato nell'introduzione, sono state documentate le varie richieste a ChatGPT, indicando il numero di richieste, l'utilità e la necessità di rielaborazione. Questo approccio consente di definire statistiche specifiche sull'utilità e sul numero di richieste.

- Fase di requisiti: Sono state effettuate solo due domande; quindi, non è rilevante fornire dati a riguardo.
- Fase di implementazione: In questa fase, sono state fatte complessivamente venti richieste. Di queste, il 60% si sono rivelate utili, il 20% inutili, mentre il restante 20% si suddivide equamente tra abbastanza utili e parzialmente utili.
- Fase di testing: Durante la fase di testing, sono state fatte quindici richieste. Circa il 53% di queste sono risultate utili, il 40% abbastanza utili e il restante 7% inutili.

6 Conclusioni

Da questo esperimento con ChatGPT si possono trarre diverse conclusioni significative. Nella mia esperienza, ritengo che ChatGPT possa essere utilizzato sia da programmatori più esperti che da persone alle prime armi, con utilizzi differenti e personalizzati. Nel mio caso specifico, non ho utilizzato ChatGPT per definire l'architettura dell'applicazione, ad esempio, ma piuttosto come strumento di risoluzione dei problemi o addirittura per automatizzare alcuni compiti.

Tradizionalmente, i programmatori hanno sempre fatto affidamento sui motori di ricerca come supporto per l'implementazione di funzionalità o la risoluzione di problemi specifici, e l'avvento dei modelli linguistici di grandi dimensioni non differisce significativamente da questa attività. Tuttavia, con l'avanzamento di questi modelli, si aprono nuove opportunità per sviluppatori di ogni livello. Una persona alle prime armi può utilizzare ChatGPT per imparare nuovi linguaggi di programmazione o per implementare funzioni di base, acquisendo così competenze fondamentali in modo più rapido e assistito.

Nonostante ciò, è importante sottolineare che l'uso di ChatGPT non esclude la necessità di sfruttare tutte le risorse disponibili. Durante lo sviluppo del mio progetto, infatti, ho fatto ampio uso delle mie conoscenze pregresse su Java e su come strutturare l'architettura dell'applicazione. La mia esperienza con determinati tipi di errori o situazioni particolari si è rivelata indispensabile. Ho inoltre utilizzato motori di ricerca e siti web specializzati per risolvere situazioni specifiche, e ovviamente ho consultato le documentazioni ufficiali fornite da Android.

Di conseguenza, come già ripetuto più volte, ChatGPT rappresenta niente meno che uno strumento aggiuntivo che aiuta e velocizza il processo di sviluppo. Non preclude l'utilizzo di risorse tradizionali come documentazioni ufficiali, forum di discussione, e tutorial online. Non ci si può aspettare che ChatGPT risolva problemi complessi in modo completo e autonomo, che il codice proposto sia sempre corretto o privo di imperfezioni. È quindi fondamentale avere una comprensione estesa dei risultati che si vogliono ottenere, delle migliori pratiche del linguaggio utilizzato e degli standard del settore.

In conclusione, mentre ChatGPT può migliorare notevolmente l'efficienza e la produttività durante lo sviluppo software, rimane essenziale combinare il suo utilizzo con una solida base di conoscenze tecniche, esperienze pratiche e altre risorse di supporto disponibili. Solo così si può garantire uno sviluppo di alta qualità, conforme agli standard del settore e capace di affrontare le sfide più complesse.

7 Bibliografia

[1] Google, «Android Developers,» [Online]. Available: https://developer.android.com/.

[2] Google, «Material Design,» [Online]. Available: https://m3.material.io/.

[3] OpenAI, «ChatGPT,» [Online]. Available: https://chatgpt.com/.

[4] Stack Exchange, «Stack Overflow,» [Online]. Available: https://stackoverflow.co/.

[5] GitHub, «GitHub,» [Online]. Available: https://github.com/.

[6] Google, «Firebase,» [Online]. Available: https://firebase.google.com/.

[7] S. Faber, «Mockito,» [Online]. Available: https://site.mockito.org/.

Ringrazio la Prof. Micucci e la Dott. Rossi per il prezioso aiuto fornito nella stesura di questa tesi.

Ringrazio la mia famiglia che mi ha sempre supportato durante il mio percorso.

Ringrazio i miei amici che conosco da una vita e sono come una seconda famiglia.

Ringrazio i miei colleghi e prima di tutto amici che mi hanno accompagnato in questi anni di università e che hanno reso questa esperienza speciale.