

Documentazione Homer



Gruppo BGN:

- **856201 Nason Ulisse (Referente)**
 - **844861 Bergamin Samuele**
 - **808826 Guastoni Simone**

-Introduzione-

Homer è un'applicazione per la ricerca e la lettura di notizie provenienti da tutte le parti del mondo, permette di ricevere informazioni personalizzate in base ai diversi topic che l'utente può selezionare al momento della registrazione.

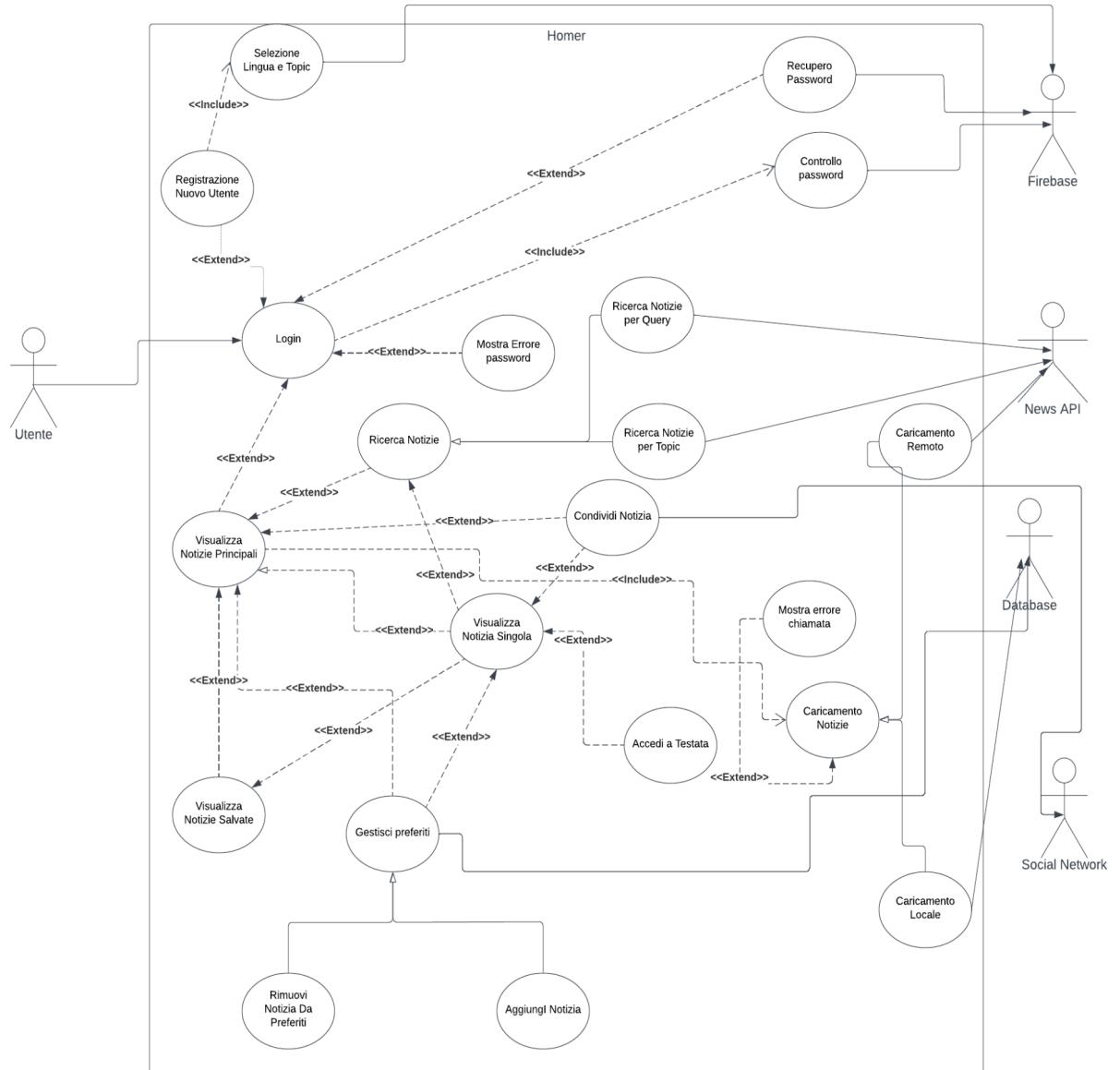
Il paese di default per la ricerca delle notizie è l'Italia ma è possibile selezionarne uno differente durante la registrazione optando per Italia, Regno Unito o Francia (Nel contesto del suo sviluppo è stato deciso di limitarsi a questi tre paesi d'esempio).

E' stato deciso di gestire le interfacce ed eventuali messaggi di risposta alle operazioni svolte dall'utente utilizzando la lingua Inglese.

Homer prende il suo nome ispirandosi ai piccioni viaggiatori che tramite una velocità media di 80Km/h venivano utilizzati sin dai tempi degli Egizi e dei Persiani per comunicare importanti informazioni, ricoprendo distanze fino a 1800 Km.

Nel corso della storia tali volatili sono stati utilizzati sempre meno e sostituiti da tecnologie di comunicazione più efficienti, al giorno d'oggi i loro discendenti prendono parte a delle gare di velocità. Questa razza di piccioni viene identificata con il nome di "Homing pigeon" abbreviato in "Homer".

-Casi d'uso dell'applicazione-



-Login:

- **Scenario di successo:**
 - L'utente entra nella schermata di login, inserisce le credenziali e preme sulla conferma.
 - Lo scenario di successo prevede che le credenziali siano corrette e l'utente possa accedere al suo account.
- **Pre-condizioni:**
 - Il dispositivo deve essere connesso a internet.
 - L'utente deve già essere registrato.
- **Scenario alternativo:**
 - Connessione internet assente:
 - Messaggio di errore.
 - L'utente tenta di accedere con una password errata:
 - Viene avviato il caso d'uso recupero password.
 - L'utente non risulta essere registrato:
 - Messaggio di errore.
 - Necessario passare alla registrazione.
- **Estensioni:**
 - Registrazione, se infatti l'utente non possiede un account o desidera creare uno nuovo può passare al caso d'uso della registrazione.
 - Recupero password: questo caso d'uso prevede che l'utente recuperi la password che non ricorda più.
 - Mostra errore password:
 - Se il sistema rileva un problema all'interno della password inserita dall'utente viene notificato un errore.
- **Inclusioni:**
 - Controllo password.

-Controllo password:

- **Scenario di Successo:**
 - Il sistema esegue il controllo della password verificando la correttezza di quest'ultima, in tal caso permette l'accesso all'utente.
- **Pre-condizioni:**
 - L'utente è connesso a internet.
 - Esiste un account riferito all'utente.
- **Scenario alternativo:**
 - Non esiste alcun account o la password non è corretta:
 - Viene notificato un errore al sistema.

-Mostra Errore Password:

- **Scenario di Successo:**
 - Viene mostrato un errore a schermo relativo al fatto che l'utente ha inserito una password errata.
- **Pre-condizioni:**
 - L'utente deve aver inserito una password errata e aver tentato il login.
- **Scenario alternativo:**
 - L'utente ha inserito una password corretta e ha eseguito l'accesso.

-Recupero Password:

- **Scenario di Successo:**
 - L'utente preme sul bottone per recuperare la password e viene rimandato a un fragment che permette di resettarla, l'utente inserisce il suo indirizzo email e il sistema provvede a inviare una mail di recupero password.
- **Pre-condizioni:**
 - L'utente deve essere connesso a internet.
- **Scenario alternativo:**
 - L'utente entra nel fragment ma il servizio non funziona per mancanza di rete.

-Caso D'uso Registrazione:

- **Scenario di successo:**
 - L'utente entra nella schermata di Registrazione, inserisce le credenziali, sceglie uno username e si registra.
 - Lo scenario di successo prevede che l'utente registri con successo un nuovo account.
- **Pre-condizioni:**
 - Il dispositivo deve essere connesso a internet.
- **Scenario alternativo:**
 - Connessione internet assente:
 - messaggio di errore.
 - L'utente inserisce una password non idonea:
 - messaggio di password non sufficientemente sicura.
- **Inclusione**
 - Il caso d'uso registrazione include il caso d'uso selezione Lingua e Topic, in questo caso d'uso l'utente sceglie la lingua delle notizie che vuole ricevere i topic per i quali ha interesse nel leggere le notizie.

-Visualizza Notizie Principali:

- **Scenario di Successo:**
 - Dopo aver effettuato il login l'utente accede alla schermata delle notizie principali.
 - Le notizie vengono caricate e mostrate correttamente .
- **Pre-condizioni:**
 - l'utente è connesso a internet o esistono notizie già salvate.
- **Scenario alternativo:**
 - Il caso d'uso "caricamento notizie" fallisce:
 - Viene mostrato un errore a schermo.
- **Estensioni:**
 - Visualizza singola Notizia.
 - Condividi Notizia.
 - Ricerca Notizie.
 - Gestisci preferiti.
 - Visualizza notizie Salvate.
- **Inclusione**
 - Questo caso d'uso include il caso d'uso caricamento notizie il quale è necessario al successo di questa componente.

-Caricamento Notizie:

- **Scenario di successo:**
 - Le notizie vengono caricate, o dal database locale o da quello remoto, in base a quanto tempo è passato dall'ultima richiesta al database remoto.
- **Pre-condizioni:**
 - Connessione internet presente.
 - Servizio remoto raggiungibile.
- **Scenario alternativo:**
 - Il servizio remoto non risulta raggiungibile:
 - Viene mostrato un errore a schermo e le notizie più recenti non vengono caricate.
- **Generalizzazioni:**

Abbiamo due generalizzazioni, infatti il caricamento delle notizie avviene in due modi distinti:

 - Caricamento locale: se è già stato effettuato un download delle notizie da un tempo limitato, allora non si rende necessario scaricare nuovamente le notizie, quindi verranno visualizzate le notizie salvate nel database locale.

- Caricamento da remoto: se è passata una ragionevole quantità di tempo dall'ultimo caricamento remoto, allora verrà effettuato un download delle notizie dal servizio.

-Visualizza singola Notizia:

- **Scenario di successo:**
 - Una volta cliccato sulla preview della notizia viene mostrata a schermo intero la card della notizia, con un link alla testata.
- **Pre-condizioni:**
 - Visualizza notizie principali ha avuto successo.
 - Tutte le notizie sono state scaricate e salvate nel database locale.
- **Scenario alternativo:**
 - Nessuno.
- **Estensioni:**
 - Gestione preferiti.
 - Condividi notizia.
 - Accedi alla relativa testata.

-Ricerca Notizie:

- **Scenario di successo:**
 - L'utente inserisce una stringa o seleziona un topic, a quel punto preme invio per effettuare una ricerca, l'applicazione esegue una richiesta al servizio e viene restituita una lista di notizie coerente a ciò che l'utente ha richiesto.
- **Pre-condizioni:**
 - Il dispositivo mobile è connesso a internet.
 - L'utente ha effettuato il login.
- **Scenario alternativo:**
 - Se manca la connessione la ricerca fallisce.
- **Estensioni:**
 - Visualizza Singola Notizia:
 - L'utente può cliccare su uno dei risultati della ricerca e verrà rimandato alla card della notizia singola.
- Generalizzazioni:
 - Ricerca Notizie per Query
 - esegue una ricerca in base a una stringa inserita
 - Ricerca Notizie per Topic
 - esegue una ricerca in base al topic selezionato

-Visualizza Notizie Salvate:

- **Scenario di successo:**
 - L'utente preme il bottone per recarsi nella schermata delle notizie salvate e l'applicazione apre l'apposita area riservata ad esse.
- **Pre-condizioni:**
 - L'utente ha effettuato il login.
- **Scenario alternativo:**
 - Se l'utente non ha precedentemente selezionato notizie da salvare, la pagina lo segnala.
- **Estensioni:**
 - Visualizza Notizia Singola: si attiva quando l'utente preme su una delle notizie salvate.

-Condividi notizia:

- **Scenario di successo:**
 - L'utente preme sul bottone apposito per condividere la notizia selezionata e viene rimandato al servizio del sistema operativo per la condivisione del link della notizia .
- **Pre-condizioni:**
 - L'utente ha effettuato il login.
 - L'utente è connesso a Internet.
 - Sono state caricate correttamente le notizie.

-Gestisci preferiti:

- **Scenario di Successo:**
 - L'utente preme sul bottone e lo stato della notizia viene modificato.
- **Pre-condizioni:**
 - L'utente ha eseguito l'accesso.
 - Le notizie sono state caricate .
- **Generalizzazioni**
 - Rimuovi Notizia:
 - In questa generalizzazione la notizia viene rimossa dai preferiti.
 - Aggiungi Notizia:
 - La notizia viene aggiunta ai preferiti.

-Tecnologie e linguaggio Utilizzati-

-Firebase:

Firebase è una piattaforma con lo scopo di fornire servizi per lo sviluppo di applicazioni mobili.

Nel nostro progetto firebase è stato utilizzato per la gestione degli account, ovvero permettere di effettuare registrazione/login e per salvare informazioni relative all'account, quali topic e lingua di interesse di ciascun utente.

-News API:

News API è una libreria che permette di accedere a una pletora di fonti di notizie, è stata usata quindi per aggregare le notizie che l'applicazione va poi a mostrare.

Questo servizio fornisce due diversi approcci di ricerca e recupero delle notizie:

- **Top HeadLines:** questa modalità ci permette di visualizzare le notizie più importanti e recenti per ogni topic, è stata quindi utilizzata nella schermata principale, la Home
- **EveryThing:** questa modalità ci permette invece di ricercare molte più notizie rispetto a top headLines e di farlo usando una query di ricerca, permettendo quindi una ricerca più accurata

-Retrofit:

Retrofit è una libreria che permette di semplificare la creazione di richieste http.

Durante lo sviluppo dell'applicazione ha avuto un ruolo cruciale rappresentato dalla necessità di effettuare il parsing delle informazioni ricevute da NewsAPI le quali devono essere poi manipolate per permettere la visualizzazione a schermo.

-Room:

Room è una libreria per android che fornisce una soluzione facile e semplice per l'utilizzo di database SQL.

Tale libreria semplifica molto il lavoro di gestione query e dei dati locali, oltre ovviamente alla gestione del database vero e proprio.

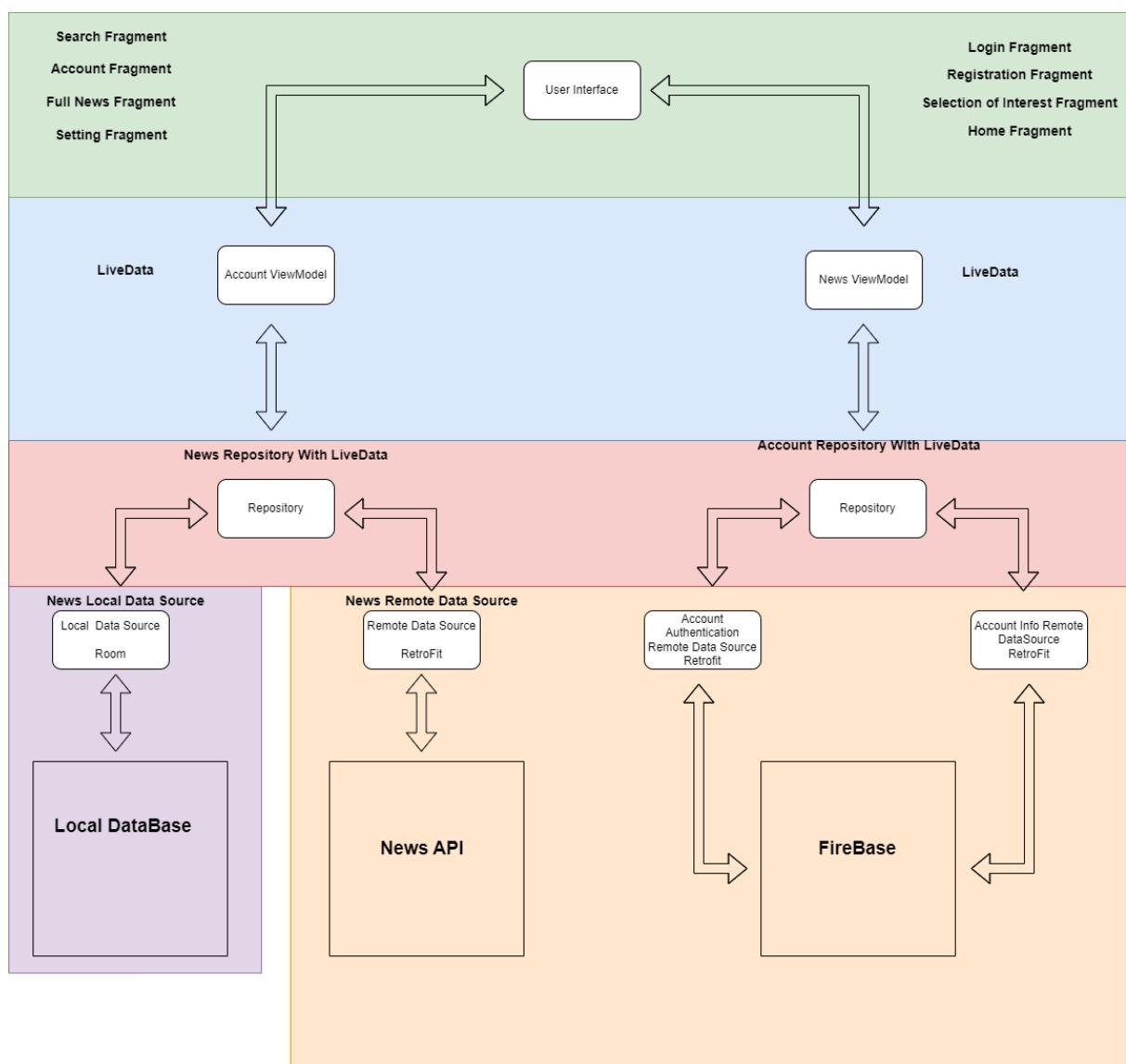
Nella nostra applicazione è stata utilizzata per gestire il salvataggio delle notizie in locale, dopo aver infatti interrogato NewsAPI e aver effettuato il parsing delle notizie, esse vengono salvate in un database apposito da dove possono essere facilmente recuperate dal ViewModel.

-Picasso:

Picasso è una libreria open-source che semplifica il lavoro di caricamento e visualizzazione di immagini associate alle notizie.

Nella nostra applicazione è stata utilizzata per recuperare e permettere la corretta visualizzazione delle immagini associate alle diverse notizie.

-Architettura-



Per permettere uno sviluppo ordinato e pulito è stato utilizzato il principio della separation of concern, ovvero il principio secondo cui la logica delle view viene separata dalla logica della gestione dei dati.

Per questo motivo è stata applicata una struttura architettonale ben precisa, detta Model-View-ViewModel, di seguito le componenti in dettaglio:

-Interfaccia utente (UI):

- **User Access Activity:** Comprende tutte le Activity e Fragment rappresentanti l'interfaccia utente per l'attività di login e registrazione.
 - **Login Fragment.**
 - **Registration Fragment.**
 - **Selection of interest Fragment.**
- **Main Activity:** Comprende tutte le Activity e Fragment rappresentanti l'interfaccia utente per l'attività principale dell'applicazione, manipolazione delle notizie, modifica dei dati, salvataggio, ricerca e lettura di notizie.
 - **Home Fragment.**
 - **Favorites Fragment.**
 - **Search Fragment.**
 - **Account Fragment.**
 - **Full News Fragment.**
 - **Setting Fragment.**

-ViewModel: Il ViewModel ha il compito di far comunicare i fragment con il livello architettonale superiore rappresentato dalle Repository, in questo modo all'interno dei fragment legati alle operazioni principali o al login vengono semplicemente chiamate tutte una serie di operazioni che il ViewModel può eseguire per comunicare con dati locali o esterni. Il ViewModel è solamente una classe di passaggio che non compie alcuna attività specifica ma è il ponte di collegamento tra i due diversi livelli architettonici.

- **News ViewModel:** Questa classe comprende al suo interno tutti i metodi che comunicano alle altre classi quali operazioni vuole svolgere l'utente che interagisce con l'interfaccia, principalmente sono contenute al suo interno richieste di fetch di notizie (In locale o in remoto) comprendenti uno o più topic ed eventuali query, richiesta di modifica di attributi di notizie salvate localmente e pulizia del database locale. Il suo metodo principale “getNews” permette di effettuare il recupero delle notizie per il fragment “Home Fragment”, viene effettuato un controllo specifico del tempo attraverso il quale si richiede una

chiamata all'API esterna solo se è passato un certo quantitativo di tempo dall'ultima operazione di fetch. Se le condizioni di tempo lo permettono prima di eseguire una chiamata viene utilizzato il metodo "clearDatabase" che come suggerisce il nome ha il compito di ripulire il database da tutte le notizie al di fuori di quelle identificate come preferite dall'utente, in modo tale da avere sempre lo spazio necessario per recuperare le nuove informazioni. Nel caso in cui le condizioni di tempo non siano rispettate lo stesso metodo ne chiama un secondo "localFetch" che si occupa di effettuare una chiamata locale per recuperare le notizie salvate all'interno del database.

- **News ViewModel Factory:** Questa classe si occupa della creazione del ViewModel, il quale viene creato apposta per essere legato ad una Repository che gli viene passata come argomento, permettendogli di comunicare con essa (La creazione del ViewModel avviene una sola volta durante l'avvio dell'applicazione quando si entra nel fragment principale).
- **Account ViewModel:** Questa classe comprende al suo interno tutti i metodi che comunicano alle altre classi quali operazioni vuole svolgere l'utente che interagisce con l'interfaccia, principalmente sono contenute al suo interno richieste di operazioni di login, logout e registrazione utente, richiesta di specifici dati di un utente in base al valore di ID passato come argomento. Il metodo principale utilizzato dai diversi fragment, facenti parte delle operazioni di login e registrazione, permette di recuperare le informazioni dell'utente connesso in quel momento, esso è chiamato "getAccountData". Tale metodo sfrutta i LiveData per evitare di effettuare più volte il recupero delle informazioni relative all'utente con lo spostarsi tra i fragment, infatti se essi sono già popolati con le informazioni dell'utente non vengono recuperate nuovamente.
 - **Account ViewModel Factory:** Questa classe si occupa della creazione del ViewModel, il quale viene creato apposta per essere legato ad una repository che gli viene passata come argomento, permettendogli di comunicare con essa (La creazione del ViewModel avviene una sola volta durante l'avvio dell'applicazione quando si entra nel fragment principale).

-Repository: La Repository ha il compito di manipolare il flusso di dati richiesto dal ViewModel e girare la richiesta di lavoro a classi che lavorano in locale (classi che si occupano della manipolazione locale dei dati) o in remoto (classi che comunicano con API o servizi esterni). La Repository ha anche il compito di gestire il risultato di queste chiamate in locale o remoto presentando dei casi di successo o di fallimento. Vengono sfruttati i LiveData come flusso di dati costante che rimane “vivo” fin tanto che l’applicazione rimane aperta e non venga distrutta.

- **News Repository with live data:** Comprende tutti i metodi che effettuano richieste di notizie da estrarre dal database locale oppure in remoto attraverso l’API oltre che tutti i casi di successo e fallimento.
 - **NewsLocalDataSource:** Rappresenta la classe vera e propria che si occupa di estrarre e manipolare i dati delle notizie salvate localmente, comprende metodi per modificare lo stato di “Favorite” di una notizia salvata localmente, salvare notizie provenienti dall’esterno all’interno del database, svuotare il database nella sua totalità oppure parzialmente mantenendo solamente le notizie preferite, estrarre una lista completa delle notizie salvate o delle notizie indicate come preferite dall’utente. Il flusso del metodo principale “getNews” entra all’interno di questa classe per poter estrarre le notizie salvate localmente nel caso in cui le condizioni di tempo non siano rispettate (Non è passato abbastanza tempo dall’ultima fetch) oppure per andare a salvare localmente tutte le notizie appena estratte dalla chiamata all’API esterna. Il metodo “saveDataInDatabase” permette di salvare le notizie localmente, viene effettuato un controllo tramite il metodo equals per verificare se tale notizia risulti già salvata precedentemente ed eventualmente evitare di sostituirla semplicemente andando a modificare il campo ID di quest’ultima. Un ulteriore metodo più complesso è rappresentato da “insertDataOnDatabase” il quale a differenza del metodo precedente permette di andare a salvare una notizia recuperata da remoto ma che non è stata salvata localmente, viene verificato se essa risulta già salvata all’interno del database, in caso contrario viene semplicemente accodata al database locale.

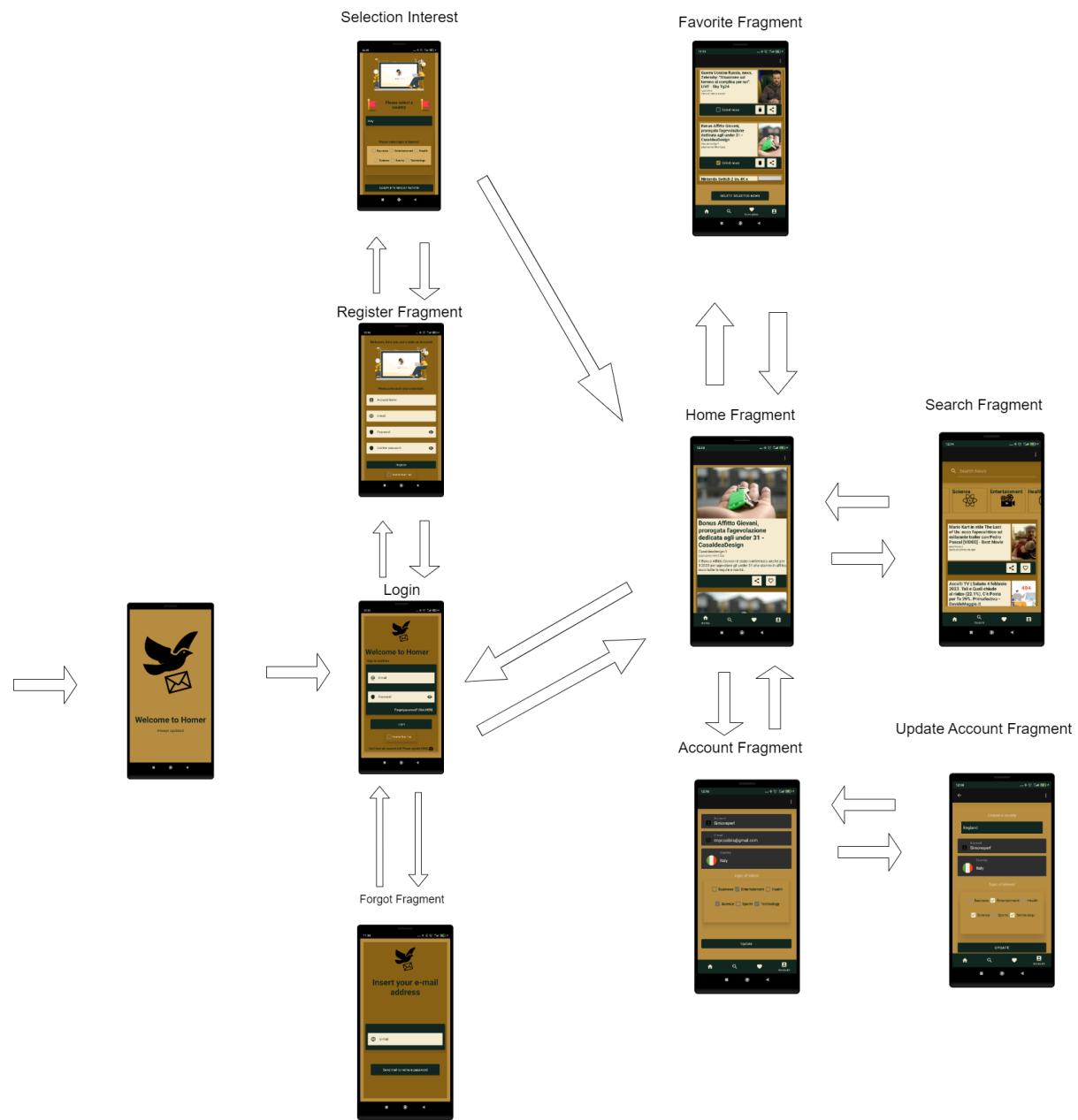
- **NewsRemoteDataSource:** Rappresenta la classe vera e propria che si occupa di effettuare le diverse chiamate alla News API e che permettono il recupero delle notizie. Questa classe presenta tre diverse tipologie di chiamata una per la HomePage che permette una call multipla sui diversi topic preferiti dall'utente, questa fetch viene eseguita seguendo il flusso del metodo principale “getNews”. Una seconda chiamata viene effettuata su di un topic specifico scelto dall'utente ed una terza che sfrutta la modalità “Everything” di NewsAPI per effettuare una ricerca approfondita in base ad una query specifica inserita sempre dall'utente.
- **Account Repository with live data:** Comprende tutti i metodi che effettuano richieste di comunicazione con Firebase, sia per l'autenticazione che per il salvataggio delle informazioni dell'utente su di un Real Time Database. Comprende anche la gestione di tutti i casi di successo e fallimento.
 - **AccountAuthenticationRemoteDataSource:** Comprende tutte le operazioni di login, logout registrazione e recupero dell'utente loggato in quel momento, operazioni che Firebase implementa in maniera autonoma. Il metodo che presenta un flusso più articolato è “signUp” infatti quest'ultimo, mediante la gestione del caso di successo e il seguente appoggio alla Repository, permette di andare ad effettuare una richiesta per salvare i dati dell'utente appena registrato.
 - **AccountInfoRemoteDataSource:** Comprende tutte le operazioni fornite da Firebase che permettono il salvataggio di informazioni al di fuori di quelle legate all'identificazione dell'utente, recupero ed eventuale modifica di queste informazioni. Tali informazioni vengono inserite all'interno di un RealTimeDatabase di Firebase, in questo modo è possibile salvare una serie di informazioni customizzate per tutti gli utenti. Il flusso del metodo “signUp” permette tramite la Repository di entrare in questa classe ed eseguire il metodo “saveAccountData” che è in grado di salvare i dati inseriti all'interno del database gestito

dallo stesso Firebase. Viene controllato se esiste già un record relativo a tale utente o se si necessita di inserirlo da zero.

-L'intera architettura è supportata da due classe principali:

1. **Service Locator:** La classe che mette in comunicazione l'interfaccia utente(UI) con Viewmodel e Repository, tale classe infatti si occupa del ricevere in ingresso i dati necessari alla creazione delle Repository che lavorano con i LiveData (sia per le notizie che per gli account), permette la creazione del database locale (News Database) e relativa interfaccia NewsDao per comunicare con esso. Permette anche l'istanziazione di Retrofit per comunicare con l'API esterna.
2. **Result:** Questa classe rappresenta il modello con il quale viene identificato il flusso di LiveData con cui si sta lavorando, in caso positivo tale flusso di lavoro sarà identificato come Result.Success, verrà quindi trattato come tale, in caso contrario verrà identificato come Result.Error. Le interfacce hanno diversi comportamenti a seconda se le loro richieste al ViewModel producono un successo o un errore.

-Descrizione Interfaccia utente (UI)-



-Navigazione: I fragment legati alle operazioni di login e registrazione sono caratterizzati da un flusso di spostamento permesso grazie ad un navGraph apposito. I fragment legati all'attività principale sono gestiti sempre mediante un navGraph apposito ma presentano anche una bottomNavigationView che permette un approccio più rapido per viaggiare tra i diversi fragment.

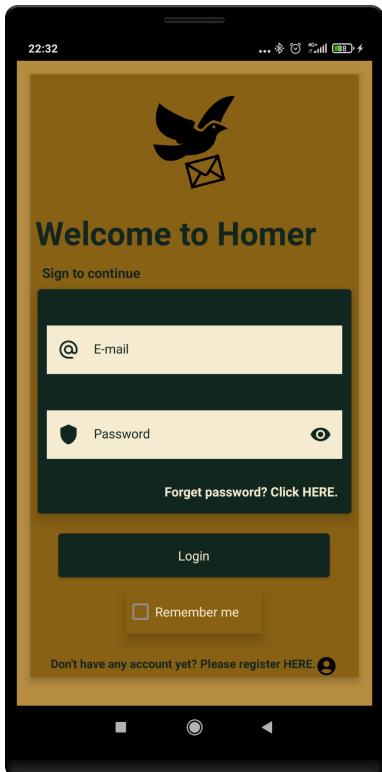
-Starting Animation Activity :



Durante l'apertura dell'applicazione verrà sempre eseguita un'animazione molto semplice di introduzione, si verrà poi rimandati al fragment di login oppure della homePage.

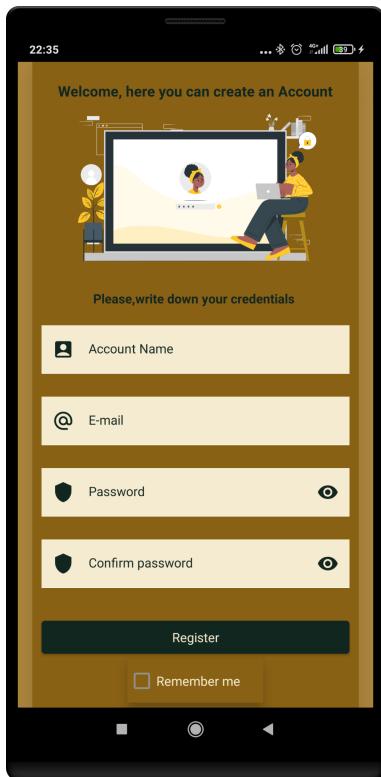
Tale animazione consiste nella semplice traslazione verso il centro dell'immagine rappresentante il logo dell'applicazione e dello slogan di accompagnamento. E' stata realizzata mediante l'utilizzo della splash animation e l'operazione di makeSceneTransitionAnimation di android studio.

-Login Fragment :



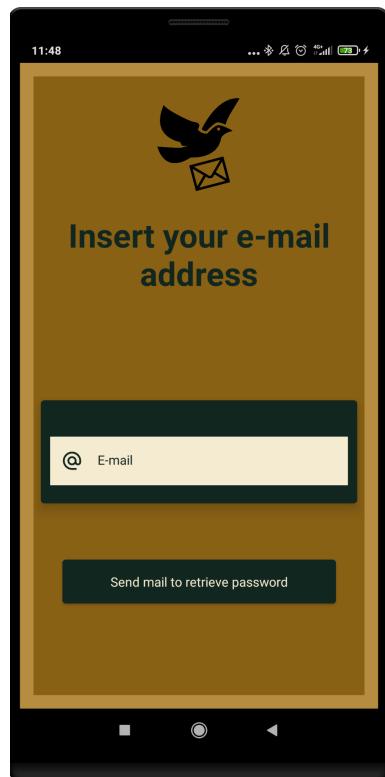
Login è la schermata che permette di effettuare l'accesso all'account utente, una volta inseriti email e password si verrà rimandati alla home fragment. Nel caso non si possedesse alcun account è possibile premere sul testo "register here" per essere rimandati al fragment di registrazione. Presenta una checkbox che permette di abilitare la funzione "Remember me" dell'applicazione. Il testo "Forgot password" rimanda al relativo fragment di interesse.

-Registration Fragment:



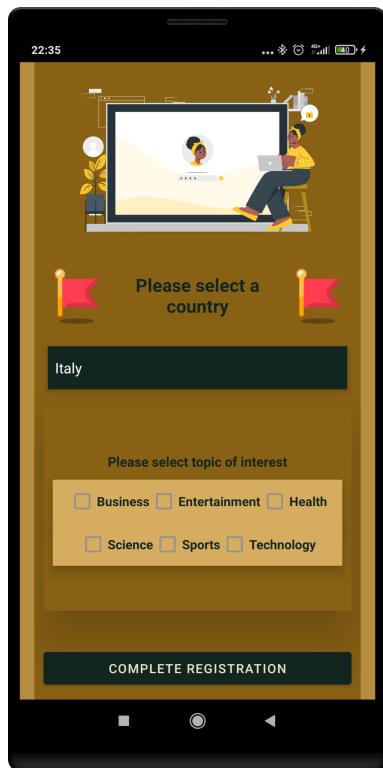
Fragment di registrazione, in questa schermata è possibile registrare un account inserendo nome account, indirizzo email e password. È possibile selezionare l'opzione “remember me” tramite una checkbox, per salvare le proprie credenziali e saltare il processo di login durante un futuro avvio dell'attività, viene abilitata quindi la funzione "Remember me" dell'applicazione.

-Forgot Password Fragment-



Tale fragment viene utilizzato se l'utente desidera reimpostare la password. Una volta inserita la propria e-mail e premuto il bottone di conferma l'utente riceverà all'interno della propria casella di posta un link per reimpostare la password.

-Fragment Selection Interest:



Come ultimo fragment facente parte del processo di registrazione è presente il fragment Selection interest, si accede subito dopo alla registrazione e serve a selezionare paese e topic di interesse. E' necessario selezionare almeno un topic, altrimenti non si potrà completare la registrazione, in caso contrario verrà notificato all'utente mediante un text di errore. Il paese risulta selezionabile tramite l'utilizzo di uno spinner, mentre i topic sono rappresentati da delle checkBox.

-Account Fragment :



Questo fragment si occupa di mostrare le informazioni relative all'account tra cui:

- Nome account
- Mail
- Nazione da cui prelevare le news
- Topic di interesse selezionati

Le informazioni personali sono inserite all'interno di TextInputLayout che presentano un attributo "editable=false", in questo modo non possono essere modificate dall'utente pur mantenendo la loro estetica. In maniera analoga vengono gestite le checkbox rappresentanti gli argomenti di interesse.

Tutti gli elementi sono popolati recuperando prima i dati dell'utente presenti in Firebase. Il TextInputLayout del paese di interesse presenta un'icona della bandiera relativa a tale paese.

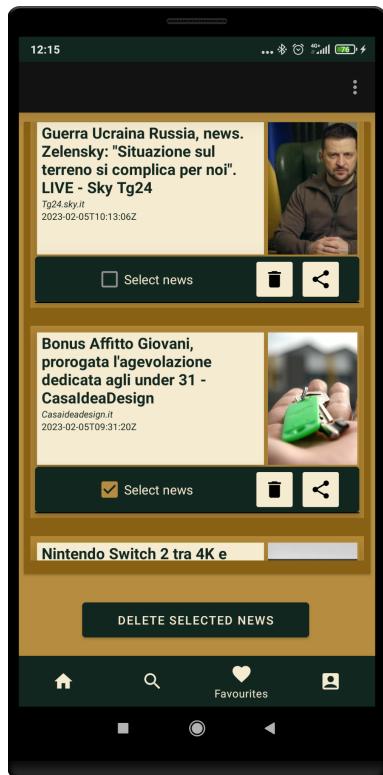
Tramite il pulsante “Update” è possibile raggiungere il fragment legato alla modifica dei propri dati utente.

-Settings Fragment :



Il settings fragment permette la modifica parziale dei dati dell’utente, possono essere modificati i topic preferiti, sempre mantenendo la scelta obbligatoria di almeno un topic, il paese dal quale si desidera ricevere le notizie ed il nome utente. Premendo nuovamente il bottone “Update” verranno effettuate le modifiche applicate. Il layout è molto simile al fragment precedente ma tutti gli elementi sono sbloccati e permettono all’utente di interagire con essi.

-Favorite Fragment :



Il favorite fragment si occupa di mostrare le notizie che sono state salvate come preferite, questa funzionalità permette infatti di salvare le notizie di interesse per poter essere lette in seguito. E’ inoltre possibile eliminare le notizie da questo fragment , singolarmente, attraverso il tasto delete, o selezionandone più di una tramite l’ausilio di una checkbox e premendo il bottone “delete selected news”. La lista di notizie preferite è gestita mediante l’utilizzo di una RecyclerView la quale presenta un Adapter ed un Listener appositi per gestire tutte le attività descritte sopra.

-Full News Fragment :



Full news fragment si occupa di mostrare una notizia nella sua interezza, essa risulta essere composta da:

- L'immagine che accompagna la notizia
- Il titolo della notizia
- Parte dell'articolo
- Link alla notizia completa

E' poi possibile poter condividere la notizia e salvarla tra i preferiti direttamente su questo fragment.

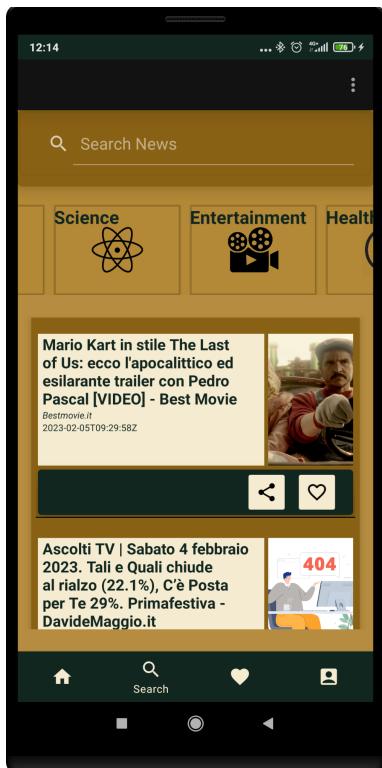
-Home Fragment :



Questo è il fragment principale, a cui si accede subito dopo il login/registrazione o appena si apre l'applicazione se si è deciso di salvare le proprie credenziali al momento del login.

All'interno di esso vengono mostrate le notizie in modalità "TopHeadLines", ovvero le notizie principali filtrate in base agli interessi dell'utente. Le informazioni recuperate localmente o da remoto vengono gestite mediante l'utilizzo di una RecyclerView, un Adapter permette una visualizzazione delle notizie personalizzata ed un Listener consente di gestire le operazioni di interazione con i bottoni "Like" e "Share" identificati da due ImageButton.

-Search Fragment :



Il search fragment è utilizzato per ricercare le notizie in base a parole chiave o topic, entrambi decisi dall'utente.

Una volta avviata la ricerca esso mostrerà le notizie trovate dalla API all'interno di una recyclerView. Sono presenti delle icone relative ai diversi topic, interagendo con esse verrà effettuata una chiamata in modalità “Top-Headlines” sul relativo argomento. Nella parte superiore del fragment è presente una SearchBar che permette all'utente di inserire una query specifica da utilizzare come argomento per la fetch.

La ricerca avviene tramite la modalità “Everything” di NewsAPI per quanto riguarda le richieste effettuate tramite SearchBar.

-Conclusioni e Sviluppi Futuri-

Il web rappresenta un mezzo potente per la diffusione delle notizie, in grado di raggiungere ora un vasto numero di persone in poco tempo.

Notizie facilmente accessibili permettono alle persone di rimanere attente e vigili alle questioni che esse ritengono più importanti, indipendentemente da razza, sesso o estrazione sociale, per questo progetti come “Homer” possono essere di grande aiuto alla democratizzazione dell'informazione, grazie proprio alla facile accessibilità alle notizie.

Gli sviluppi futuri dell'applicazione:

- **Implementazione notifiche:** al momento l'interazione con l'utente nei momenti in cui l'utente non apre l'applicazione è inesistente.
Per evitare che l'utente non sia prontamente informato sulle notizie più importanti e urgenti è necessario implementare un sistema di notifiche che sia in grado di discriminare le notizie più importanti dal rumore di fondo.
- **Scelta testate:** Gli utenti potrebbero avere preferenze per certe testate oltre che per topic, un sistema che permette di favorire le testate preferite all'utente nella home, potrebbe risultare utile.
- **Aggiunta paesi provenienza notizie:** Inserire tutti i paesi supportati dalla NewsAPI.
- **Implementazione lingue:** Inserire diverse lingue per interfacce e messaggistica interna, per le quali al momento abbiamo solo l'inglese, e renderle selezionabili dall'utente stesso.