



Progetto di Programmazione di Dispositivi Mobili  
a.a. 2022/2023

Oltolini Edoardo 869124  
Gherardi Marco 869138  
Lombardo Matteo 869232  
Monti Lorenzo 869960



Università degli Studi Milano-Bicocca,  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Informatica Triennale

## **INDICE:**

1. Introduzione
2. How It Started
3. Casi D'Uso
4. Architettura del Sistema & Pattern Utilizzati
5. Librerie Utilizzate
6. User Interface & User Experience
7. Navigation Graphs
8. Sviluppi Futuri

# INTRODUZIONE

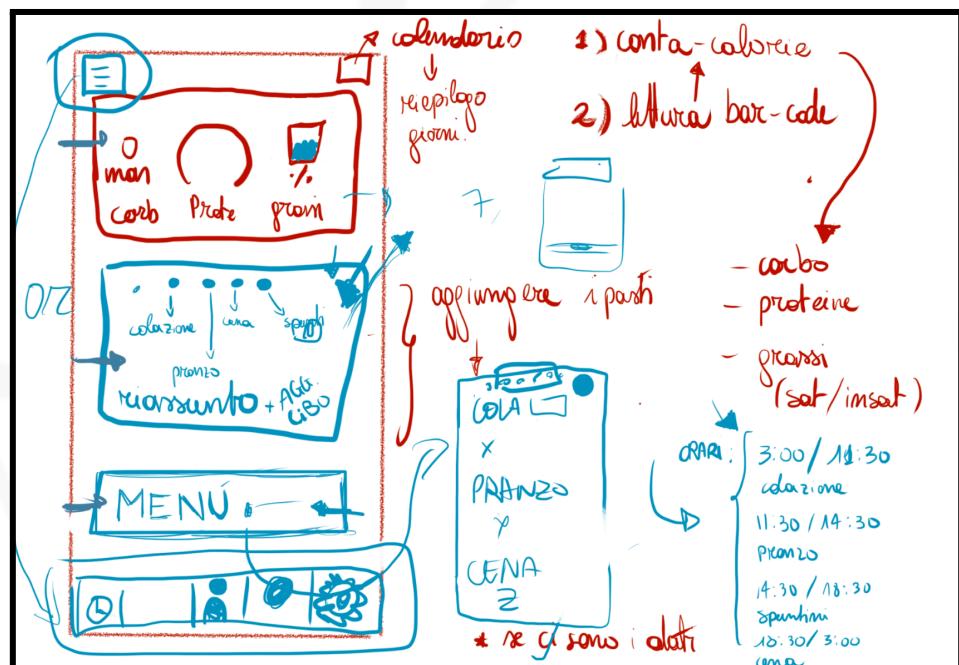
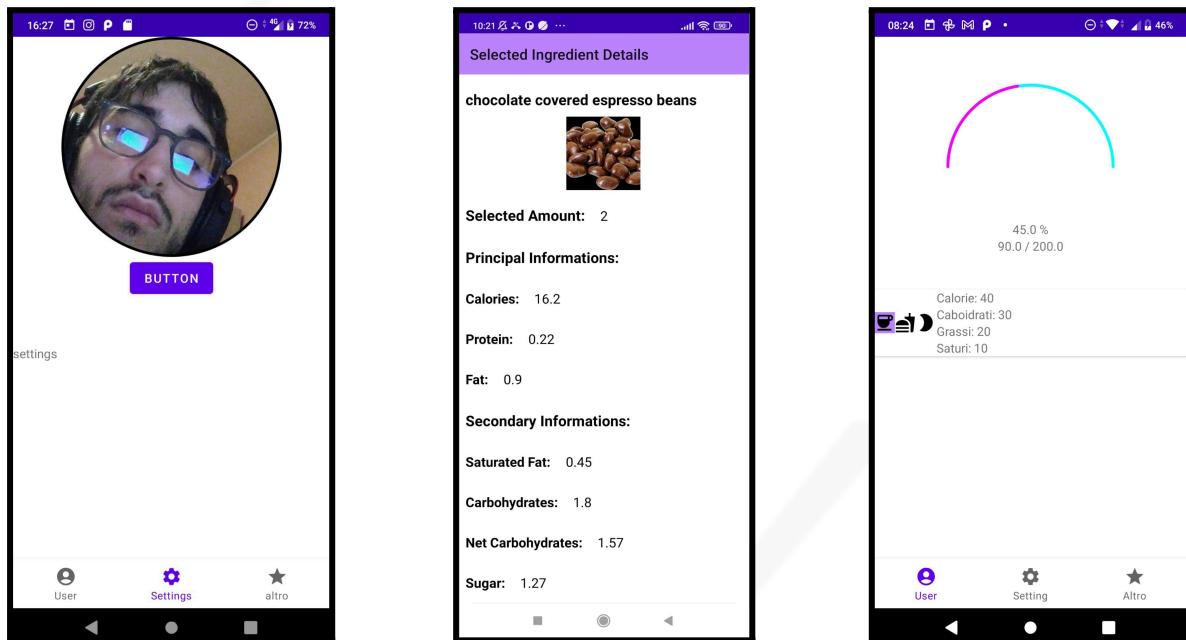
Il nome dell'app, “Karori”, nasce dalla traduzione della parola “Caloria” in giapponese.

Karori è un'applicazione che nasce con l'obiettivo di poter accompagnare ed aiutare una persona che sta seguendo un percorso di dieta volto al dimagrimento o all'aumento della massa muscolare o semplicemente permettere di monitorare le calorie assunte durante i vari pasti della giornata. L'utente sarà in grado di cercare un alimento, visualizzarne le informazioni complete e aggiungerlo al pasto, può visualizzare un riepilogo, per ogni pasto, di quante calorie, proteine, grassi e carboidrati sono stati assunti e può infine visualizzare tutti i cibi inseriti, da cui trarre anche informazioni aggiuntive oltre a quelle mostrate dal riepilogo.

# HOW IT STARTED...

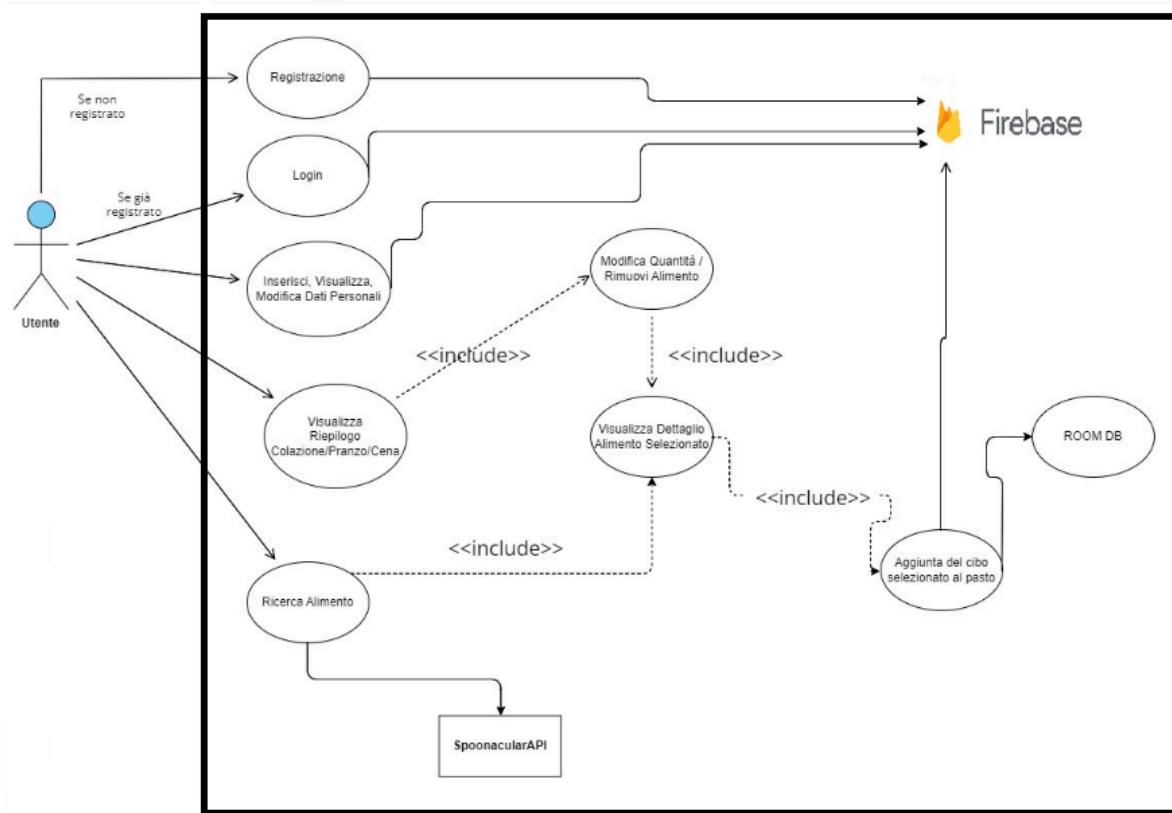
Wait, it's all... PURPLE?

Always has been



...HOW IT'S GOING

# CASI D'USO



## Registrazione

Attore: Utente

L'utente può registrarsi a Karori attraverso account Google o mail personale.

## Inserisci Dati Personalni

Attore: Utente

Per terminare la registrazione, l'utente deve inserire i propri dati personali (Altezza, Età, Peso) al fine di poter calcolare le calorie adatte a lui. Potrà successivamente visualizzare e modificare i suddetti dati all'interno di "Impostazioni".

## Login

Attore: Utente

L'utente può effettuare un Login tramite credenziali (email e password) o tramite account Google, d'ora in poi può accedere ai servizi forniti dall'applicazione.

## Visualizza Riepilogo

Attore: Utente

L'utente può visualizzare il riepilogo dei pasti principali della giornata, può quindi vedere le principali informazioni: Calorie, Grassi, Proteine, Carboidrati.

## **Ricerca Alimento <<include>> Visualizza Dettagli Alimento**

Attore: Utente

L'utente, mentre visualizza il riepilogo di un pasto, cliccando sul pulsante + potrà cercare il nome dell'alimento di cui vuole ottenere informazioni specifiche o aggiungerlo al pasto selezionato precedentemente.

## **Modifica Quantità / Rimuovi Elemento**

### **<<include>> Visualizza Dettagli Alimento**

Attore: Utente

Cliccando sul riepilogo desiderato, vengono mostrati gli elementi inseriti, l'utente può rimuovere gli elementi o modificarne la loro quantità.

## **Visualizza Dettagli Alimento <<include>> Aggiunta Alimento**

Attore: Utente

Una volta visualizzati tutti gli alimenti provenienti dal caso “Ricerca Alimento”, l'utente può visualizzare i dettagli cliccando sull'alimento desiderato.

## **Aggiungi Alimento**

Attore: Utente

Dalla schermata dei “Dettagli”: l'utente, se ha scelto di aggiungere un alimento, può aggiungerlo al pasto selezionato precedentemente, verranno aggiornati i dati di “Riepilogo”.

## **Visualizza Pasto Passato**

Attore: Utente

L'utente andando nella sezione “Calendario”, potrà selezionare la data nella quale vorrà visualizzare i dati inseriti e cliccando su essa usciranno le informazioni relative a quest'ultimi.

# ARCHITETTURA DEL SISTEMA & PATTERN UTILIZZATI

L'applicazione si compone essenzialmente di tre strati:

## Activity e Fragment:

Gestiscono l'interfaccia grafica dell'applicazione, popolando le View ed effettuando i vari passaggi da una schermata all'altra. Ottengono i dati dai LiveData contenuti nei ViewModel.

## ViewModel:

Uno strato di collegamento tra la UI ed i repository; il repository scrive le informazioni sul LiveData contenuto nel ViewModel e la UI, dopo aver ricevuto notifica della scrittura, leggerà dai LiveData le nuove informazioni con cui popolare le View.

## Repository:

Recuperano dal database e dalla API i dati su cui lavoreranno le Activity e i Fragment; I dati verranno scritti nei LiveData contenuti nel MealViewModel. (Viene sfruttato Room DB).

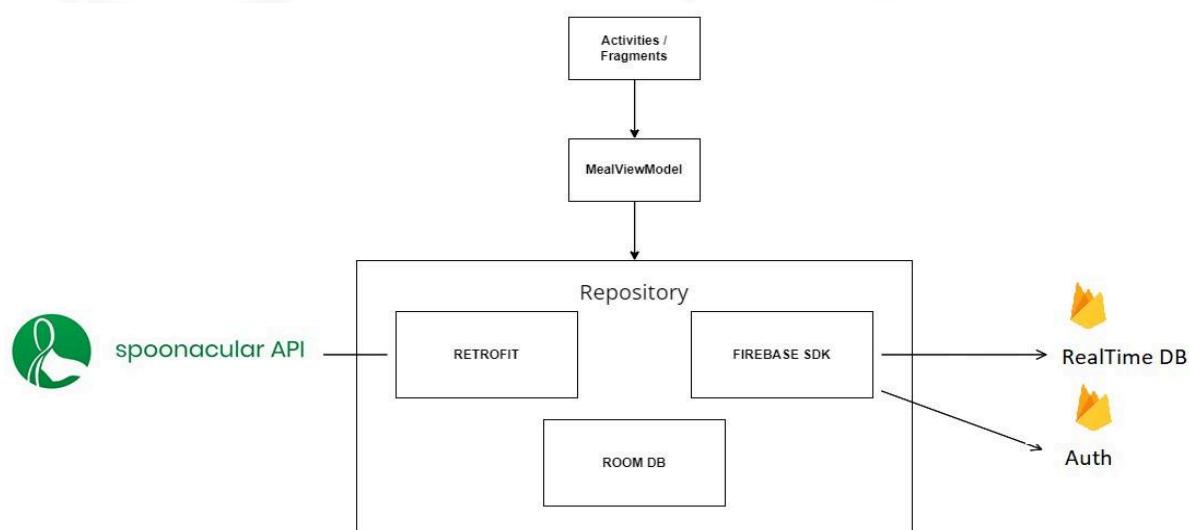
In supporto a questi componenti abbiamo quindi utilizzato i seguenti servizi esterni:

## SpoonacularAPI:

Per recuperare tutte le informazioni necessarie di un alimento quali calorie, grassi, proteine, carboidrati e dati “secondari”.

## Firebase:

Utilizzato per il suo servizio di autenticazione (Auth) e salvataggio (RealTime DB) degli utenti per il login con email e password e/o attraverso account google e per salvare i loro dati personali (altezza, peso, età, goal).



Oltre al pattern architetturale **MVVM**, abbiamo utilizzato i seguenti **pattern**:

### **Listeners:**

Abbiamo utilizzato i listener soprattutto per dialogare con la classe “RequestManager”, classe che dialoga con SpoonacularAPI, i listener in questo contesto implementano metodi “onResponse” per poter gestire le risposte fornite dall’API, in più abbiamo implementato metodi del tipo “onClick” listener più specifici per ottenere gli id degli alimenti cliccati o per selezionare gli elementi delle recyclerView.

### **Qualche Listener in dettaglio:**

Listener per RequestManager e RecyclerView, in riferimento a “RicercaEAggiungiFragment”

Se l’utente vuole cercare “ingredienti”, viene invocato il metodo `getIngredientSearchResult` della classe RequestManager, questa, grazie a Retrofit, effettua Query a SpoonacularAPI per ottenere informazioni (in questo caso si otterrà una response contenente il nome dell’ingrediente e la sua immagine), `getIngredientSearchResult` ha come parametri “ingredientListener” e il testo della ricerca, “ingredientListener” essendo un’interfaccia fornisce la signature di due metodi: “didFetch” e “didError”, la cui funzionalità viene implementata poi nella classe fragment, “didFetch” in particolare ha come parametri la risposta della chiamata all’API e un messaggio di ritorno, la risposta viene data all’adapter per popolare correttamente gli ingredienti all’interno della recyclerView. Viene poi implementato nella classe fragment un “idListener”, questo serve soltanto ad ottenere l’id dell’alimento cliccato (grazie al metodo “onClickedIngredient”), l’id viene passato al fragment successivo (su cui poi si effettuerà una seconda chiamata per ottenere le informazioni complessive).

### **Adapters:**

Gli adapters sono stati utilizzati, insieme ai viewHolder, per poter creare e popolare le varie recyclerView che sono delle “liste” che possono contenere elementi dotati di una View particolare (abbiamo utilizzato CardView) e quindi possono essere riempite con elementi composti da vari.

### **Un Adapter in dettaglio:**

AdapterChangeRiassunti: istanzia gli estratti del giorno grazie al componente ViewPager 2.

### **Factory:**

La factory che è stata utilizzata è quella per la creazione del UserViewModel, definendo un metodo per creare un ViewModel passandogli come attributo un oggetto Repository;

### **Un Factory in dettaglio:**

UserViewModelFactory: interfaccia in cui è definito il metodo create per un UserViewModel;

# LIBRERIE UTILIZZATE

Per lo sviluppo dell'applicazione sono state utilizzate le seguenti librerie:

**Picasso:** libreria che permette la gestione delle immagini e contenuti multimediali. E' stata utilizzata per il caricamento delle immagini attraverso gli URL che l'API restituiva riguardante ingredienti e ricette.

**Firebase:** la libreria di Firebase, necessaria per l'autenticazione dell'utente e l'utilizzo del Realtime Database;

**Google Play Services:** libreria necessaria per effettuare l'autenticazione usando "Accedi Con Google".

**Retrofit:** una libreria che permette di effettuare richieste HTTP alle API; in particolare grazie a questa libreria, i thread vengono gestiti in automatico in modo trasparente al programmatore, in modo da non occupare il thread principale dell'applicazione, quindi ne utilizza uno dedicato per ogni richiesta.

**JSON:** libreria che viene utilizzata da Retrofit per convertire (GsonConverterFactory) le response in formato JSON restituite dalle API in classi Java.

**CircularProgressorIndicator:** utilizzata nella schermata Home per mostrare le calorie raggiunte su un totale prefissato tramite dei parametri e una formula.

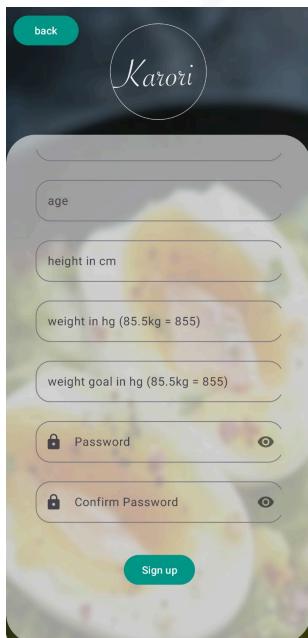
**Material Calendar:** Utilizzata per mostrare il calendario, così dopo aver selezionato il giorno si può vedere il resoconto specifico..

**Google material components:** Per usare il material Components con i nuovi sistemi del Material 3.

# USER INTERFACE & USER EXPERIENCE

## & cosa accade dietro alle azioni utente?

### REGISTRAZIONE UTENTE & INSERIMENTO DATI PERSONALI



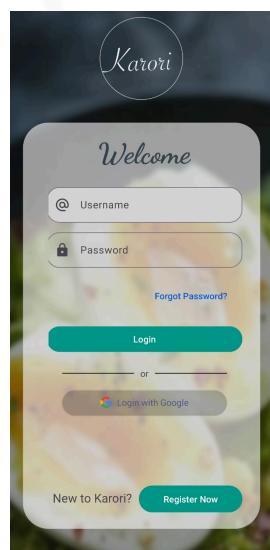
All'interno di "Firebase: RealTime DB" viene quindi inserita la mail e la password (che verranno utilizzate per l'accesso), l'altezza, il peso, l'età dell'utente e il peso che l'utente vorrebbe raggiungere (goal), al termine della registrazione viene inserito nel database anche il numero di calorie da assumere durante la giornata. Viene utilizzata la formula "parziale" per il calcolo del TDEE: Total Daily Energy Expenditure, utilizzando il BMR: Basic Metabolic Rate. Siccome non facciamo scegliere all'utente il suo sesso, ipotizziamo che sia un maschio che abbia uno stile di vita abbastanza sedentario; la formula utilizzata è:

$$\text{BMR} = 66 + (13.7 * \text{goal in kg}) + (5 * \text{altezza in cm}) - (6.8 * \text{età})$$

$$\text{TDEE} = 1.2 * \text{BMR}$$

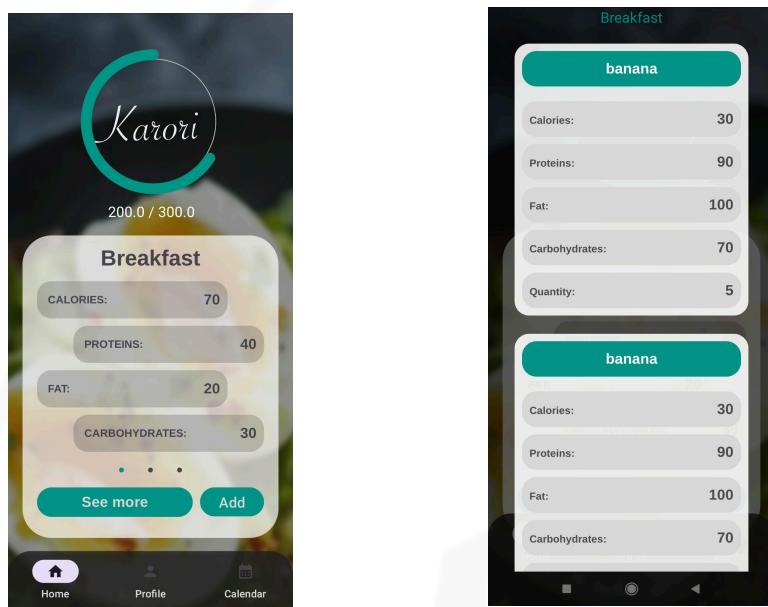
L'utente potrà successivamente modificare le proprie informazioni personali all'interno dell'area "Impostazioni" e può anche inserire manualmente in numero di calorie che vuole assumere giornalmente.

### LOGIN UTENTE



L'utente inserisce i propri dati, email e password o account google, se non ha già effettuato il login precedentemente; In caso contrario il login avverrà automaticamente caricando le informazioni dell'utente che ha effettuato l'accesso manualmente precedentemente. L'utente potrà successivamente effettuare il logout con il proprio account dall'applicazione nell'area "Impostazioni" che resetterà tutte le informazioni riguardanti il login automatico.

## HOME



Nella schermata vengono mostrati i riassunti della giornata: Colazione, Pranzo e Cena. Si può cambiare la visualizzazione tramite uno swipe a destra o sinistra.

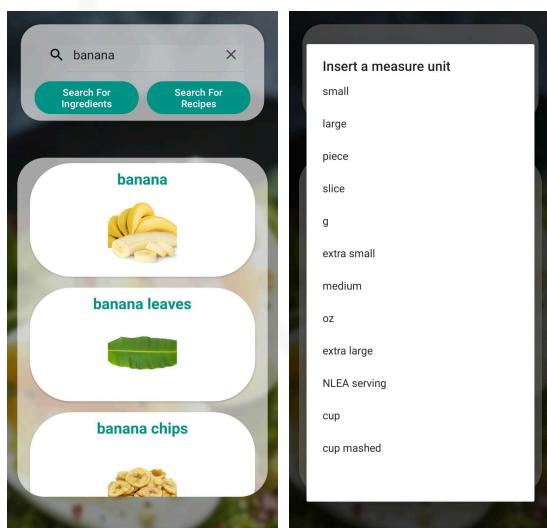
Il bottone “See more” serve per far vedere gli alimenti salvati che hanno portato ai valori mostrati sullo schermo. Andrà a mostrare un pop-up con gli alimenti mangiati, dove cliccando sull’alimento andrà ad aprirsi un’altra schermata che mostrerà dati più specifici. Il bottone “Add” serve per andare a cercare l’alimento che si ha mangiato per aggiungerlo a una delle 3 liste di alimenti (Colazione, Pranzo e cena).

Il logo mostra a quante calorie si è arrivati in base alla formula precedentemente indicata.

## CALENDAR

Nella schermata Calendar c’è la possibilità di andare a selezionare un giorno e mostrare il resoconto di quella giornata e i progressi raggiunti.

## RICERCA ALIMENTO



Nella schermata di “Ricerca”, un Utente può cercare un Alimento da aggiungere ad un pasto selezionato (Colazione, Pranzo o Cena), una volta cliccato un Alimento comparirà prima una schermata per indicarne la quantità e successivamente una schermata per selezionare l’unità di misura; in alternativa un Utente può anche cercare una Ricetta da cui andrà a vedere gli Ingredienti di cui è composta.

## VISUALIZZA INFORMAZIONI ALIMENTO / RICETTA

Nella schermata delle “Informazioni”, un Utente, se ha cercato e cliccato su un Alimento, può visualizzare le informazioni principali: Calorie, Carboidrati, Proteine, Grassi; le sue proprietà glicemiche e le informazioni secondarie: Vitamine, Zinco, Ferro...

Un Utente può successivamente aggiungere l’alimento (al pasto selezionato, attraverso Room) cliccando sul tasto “Add Aliment”.

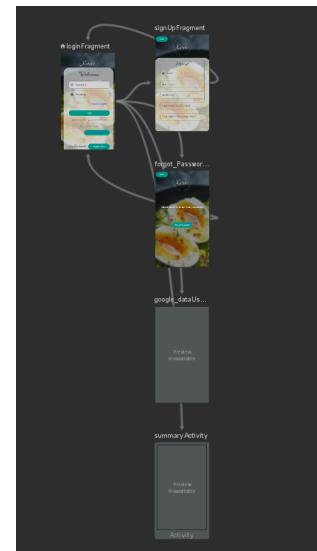
Se un Utente ha cercato e cliccato su una Ricetta, può visualizzare, nella schermata delle informazioni, il nome della Ricetta, un link al sito in cui è stata pubblicata, un riassunto della Ricetta e gli Ingredienti di cui è composta.

## PROFILO

L’utente troverà in questa schermata tutte le informazioni personali: altezza, peso, goal, età, kilocalorie, email; Tutte queste informazioni potranno essere cambiate manualmente tramite un pulsante che permetterà all’utente di sovrascrivere i dati e aggiornarli nel database; il calcolo delle kilocalorie utilizzerà la stessa formula utilizzata nella schermata del login. Un’altra feature presente è quella del cambiamento della password; Premendo un pulsante arriverà un mail all’indirizzo dell’utente con cui potrà cambiare password. Infine è presente anche un pulsante per effettuare il logout che una volta premuto porterà direttamente alla schermata principale del login. Lo switch serve per abilitare le modifiche.

# NAVIGATION GRAPHS

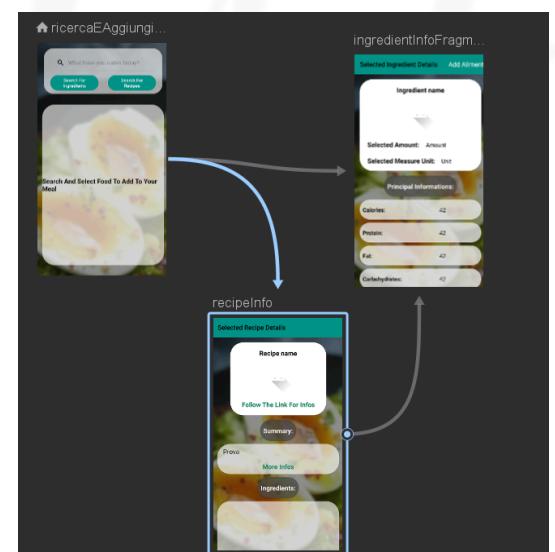
- Login Fragment, si hanno 3 opzioni:
  - SignUp Fragment (torna al Login una volta finita la registrazione)
  - Summary Activity (si va alla schermata principale col pulsante “Login”)
  - Forgot Password Fragment (reimpostazione della password)
  - Google Data User (Sincronizzazione Account Google e firebase e inserimento dei dati)



- Home Fragment (schermata principale), selezionando dalla toolbar si hanno 2 opzioni:
  - Fragment Impostazioni Profilo
  - Fragment Calendario



- Fragment Ricerca e Aggiungi, 2 opzioni
  - Fragment Ingredient Info (vedere e aggiungere alimenti)
  - Fragment Recipe Info (nome, link, riassunto ricetta e ingredienti di cui è composta)
- Fragment Recipe Info
  - Fragment Ingredient Info (vedere e aggiungere alimenti)



## SVILUPPI FUTURI

- Implementare sinergicamente Room e Firebase in modo che non si percepiscano “distacchi” nel modo in cui si ottengono le informazioni.
- Ampliare le informazioni salvabili, in particolare: aggiungere la possibilità di inserire degli alimenti “preferiti” o visualizzare alimenti “recenti”
- Aumentare le funzionalità offerte dall’applicazione, ad esempio avere la possibilità di salvare ricette e poter inserire anch’esse nei riepiloghi giornalieri
- Inserire un contatore per i litri d’acqua assunti giornalmente
- Migliorare l’applicazione dal punto di vista della UI & UX
- Inserire una funzionalità che permette di visualizzare il menù giornaliero dell’utente, ossia un menù personalizzato inserito dall’utente stesso per avere il proprio piano alimentare dato dal dietologo sempre a portata di mano.