

# COOKERY



Documentazione progetto per il corso di **Dispositivi Mobili** 2021/2022

Cattaneo Alessandro **851612**

Inajjar Adam **859260**

Riva Andrea **838845**

# Indice

<b>Progetto .....</b>	<b>3</b>
<b>Architettura .....</b>	<b>4</b>
<b>Classi .....</b>	<b>5</b>
Classi UI .....	5
Classi Firebase .....	6
Classi Repository .....	6
Classi Utils & Service .....	7
<b>Navigazione .....</b>	<b>8</b>
<b>Implementazioni future .....</b>	<b>11</b>
<b>Strategie di remunerazione .....</b>	<b>12</b>

# Progetto

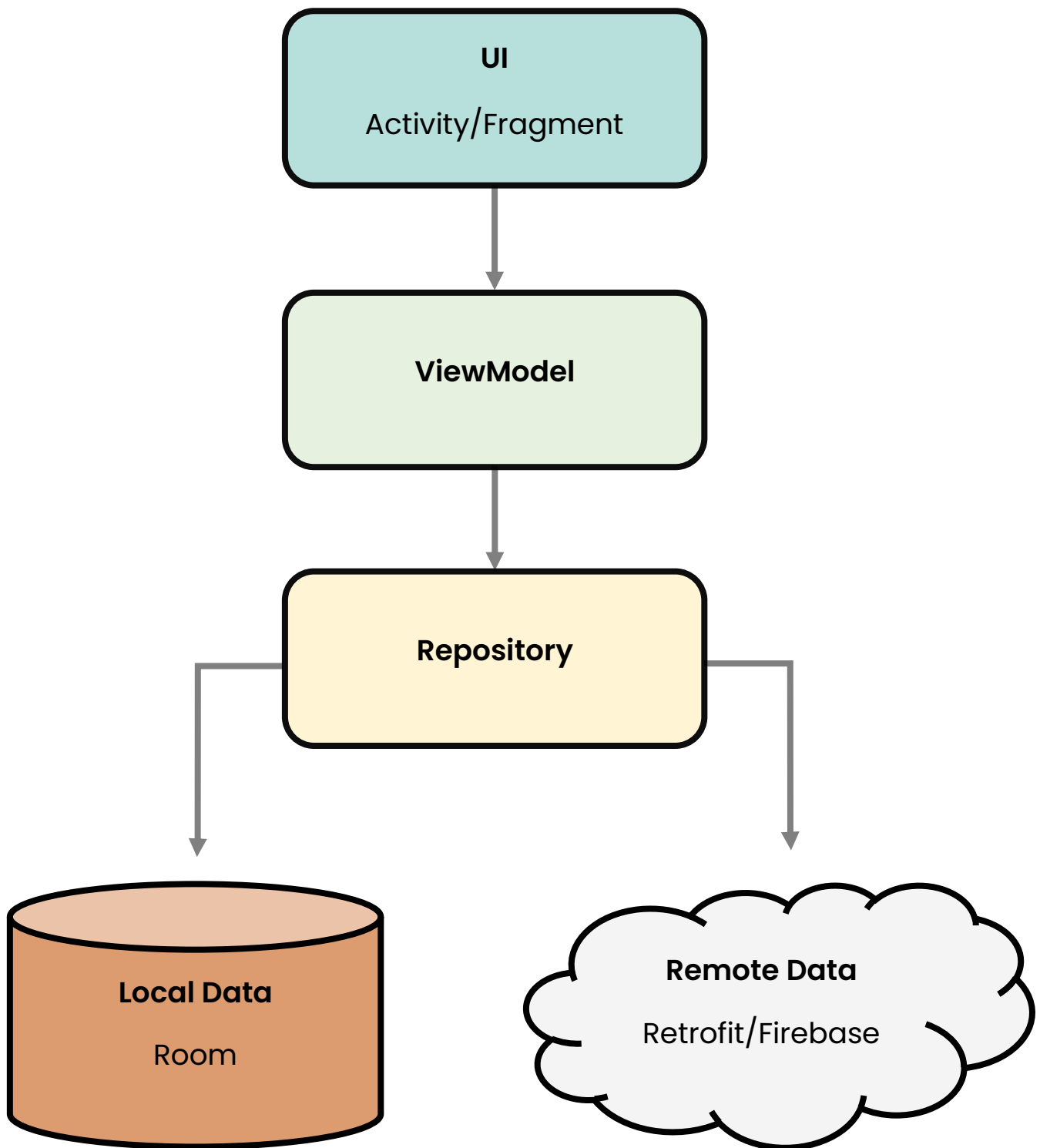
Cookery nasce con l'obiettivo di ridurre gli sprechi di cibo e valorizzare gli ingredienti che abbiamo a casa grazie ad una vastissima collezione di ricette già presenti sulla API utilizzata e pubblicate dagli utenti.

Infatti è possibile l'inserimento di una ricetta creata dall'utente selezionando gli ingredienti presenti nel database e descrivendo i vari step di preparazione.

Una volta aperta l'applicazione, ed inseriti i primi ingredienti nella nostra dispensa, sarà possibile visualizzare le ricette che potremo preparare con quegli stessi ingredienti che abbiamo inserito; Inoltre sarà possibile visualizzare le ricette per cui ci mancano pochi ingredienti e con un semplice tocco saremo in grado di riempire la lista della spesa con gli ingredienti mancanti.

L'utente potrà anche impostare delle preferenze alimentari, in modo da ridurre il tempo impiegato nella ricerca di una ricetta in base alle sue allergie o gusti!

# Architettura



# Classi

## Classi UI

- **MainActivity**: Questa classe gestisce i collegamenti con i fragment principali (Home, Pantry, Recipe) e permette, tramite un drawer menu, di collegarsi all'activity adibita alle operazioni di login e registrazione tramite **Firestore**.
- **HomeFragment**: Questa classe si occupa di mostrare all'utente le ricette che può preparare con gli ingredienti che ha in dispensa (Ready to Cook), e delle ricette casuali suddivise nelle seguenti categorie: First courses, Main courses, Desserts.
- **PantryFragment**: Questa classe gestisce le diverse dispense dell'utente e tutti gli ingredienti che possono essere facilmente aggiunti ad esse grazie ad una bottomSheet adibita alla ricerca degli ingredienti nel **database locale**.
- **RecipeFragment**: Questa classe mostra all'utente tutte le ricette da lui create in generale o filtrandole con gli appositi filtri e search bar, collegandosi direttamente al **database locale** dove vengono salvate una volta create.
- **MakeRecipe**: Questa classe gestisce la creazione delle nuove ricette. Vengono recuperati tutti gli ingredienti presenti nel **database locale** e passati all'utente da selezionare per essere aggiunti alla ricetta, inoltre è possibile aggiungere tutti gli step di preparazione.
- **SingleRecipeActivity**: Questa classe si occupa di mostrare i dettagli di una singola ricetta selezionate dall'utente,

mostrando gli ingredienti, evidenziando in rosso quelli mancanti e gli steps.

## Classi **Firestore**

- **LoginRegisterUser**: Questa classe permette all'utente di effettuare il login o passare all'activity di registrazione tramite **Firestore**.
- **RegisterUser**: Questa classe permette all'utente di registrarsi tramite **Firestore**.
- **User**: Questa classe definisce i campi di dati presenti nel **realtime database** di **Firestore**.

## Classi **Repository**

- **DatabasePantryRepository**: Questa classe permette di comunicare con il **database locale** per andare a leggere, modificare e cancellare gli ingredienti della dispensa. Inoltre popola la tabella ingredientApi con i 1000 ingredienti più usati forniti dall'API.
- **IngredientAndStepRepository**: Questa classe effettua le chiamate all'API per ottenere gli ingredienti e gli step delle ricette ReadyToCook una volta che si effettua un tap su una di esse.
- **IngredientMeasureUnitRepository**: Questa classe effettua le chiamate all'API per ottenere le unità di misura di un determinato ingrediente ogni volta che viene aggiunto alla dispensa o quando si aggiunge durante la creazione, da parte dell'utente, di una ricetta.

- **RecipeRepository**: Questa classe si occupa delle operazioni di Read e Write delle ricette nel **database locale**.

## Classi **Utils** & **Service**

- **ServiceLocator**: Classe che serve per la creazione di due oggetti singleton che permettono l'accesso al **database locale** mediante Room e crea l'oggetto che sarà utilizzato per effettuare le chiamate API mediante **retrofit**.
- **ResponseCallbackDb**: Classe che definisce le firme dei metodi che vengono invocati quando si ottiene il risultato di una query fatta al **database locale**.
- **CsvReader**: Classe che serve per leggere il file csv, contenente i vari ingredienti, locato nella cartella assets. Restituisce una lista di oggetti IngredientApi che verranno salvati nel **database locale** al primo avvio dell'applicazione.
- **SpoonacularApiService**: Questa interfaccia definisce quali sono le richieste da effettuare all'API definendo come deve essere formato l'URL e i suoi vari parametri e che tipo di oggetti devono restituire le varie richieste effettuate.
- **IngredientUnitMeasureResponseCallback**, **ResponseCallbackAPI**, **ResponseCallbackStepAndIngredients**: Queste interfacce definiscono le firme dei metodi da invocare una volta che è stata ricevuta la risposta a una chiamata API sia che essa sia andata a buon fine sia che non abbia avuto successo.

# Navigazione

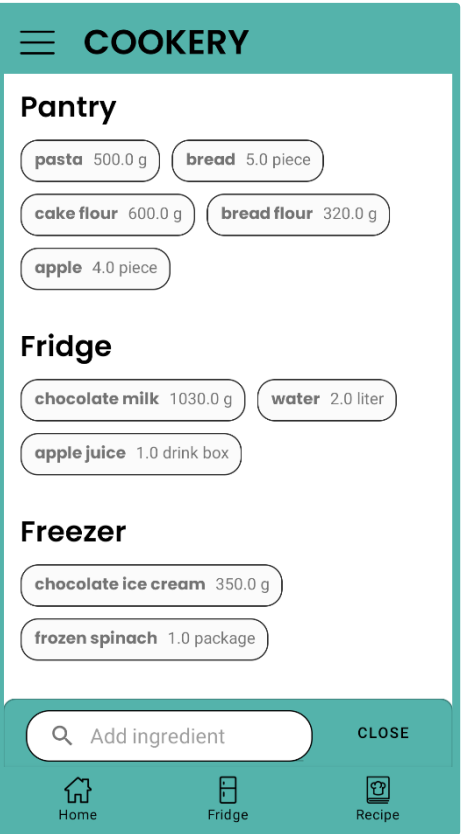
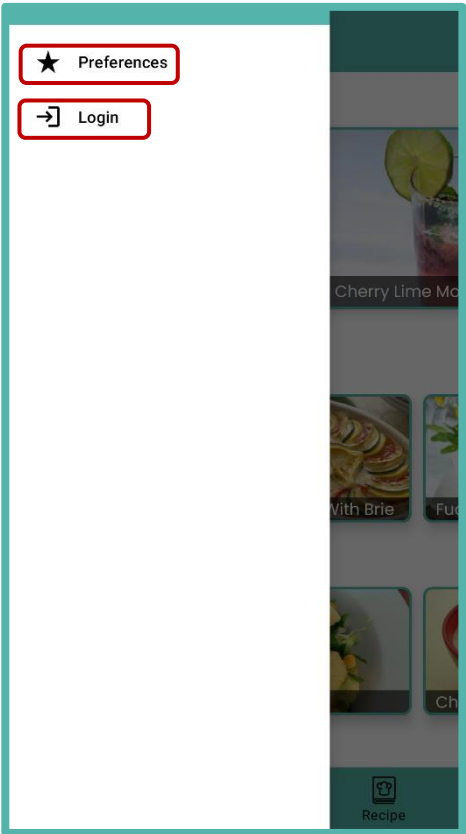
## Home



## Recipes



## Fridge





MainActivity

## Login



←



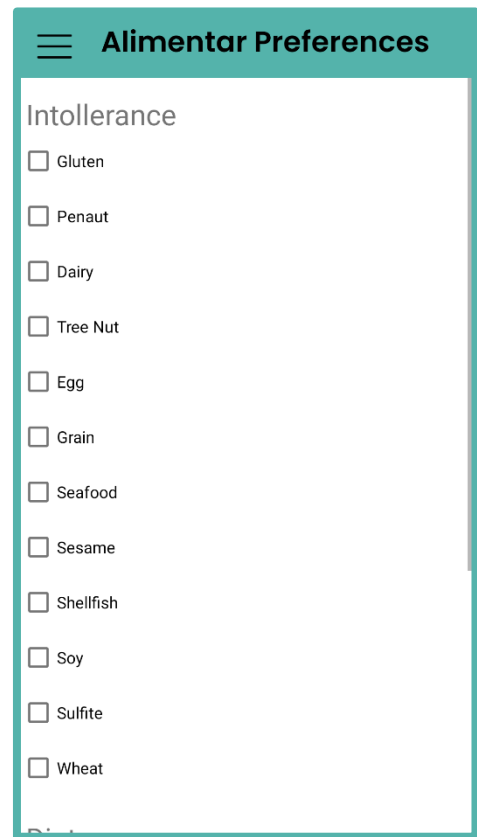
email address

password

LOGIN

Forgot password? Register

## Preferences

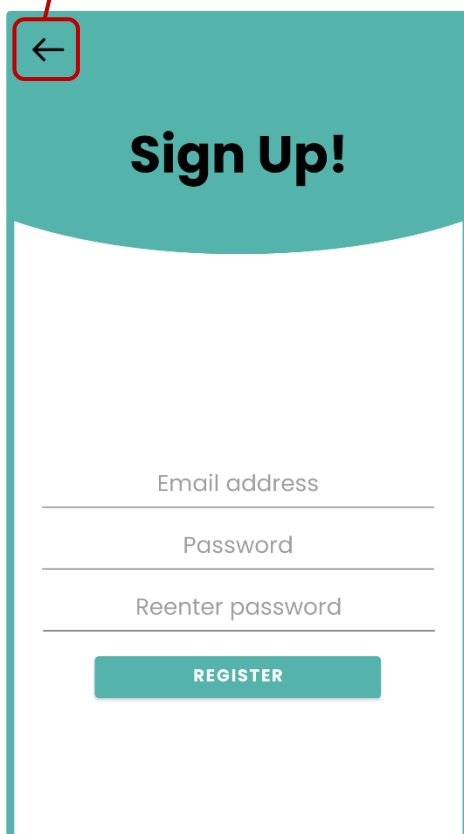


≡ Alimantar Preferences

Intolerance

- ☐ Gluten
- ☐ Penaut
- ☐ Dairy
- ☐ Tree Nut
- ☐ Egg
- ☐ Grain
- ☐ Seafood
- ☐ Sesame
- ☐ Shellfish
- ☐ Soy
- ☐ Sulfite
- ☐ Wheat

## Register



←

**Sign Up!**

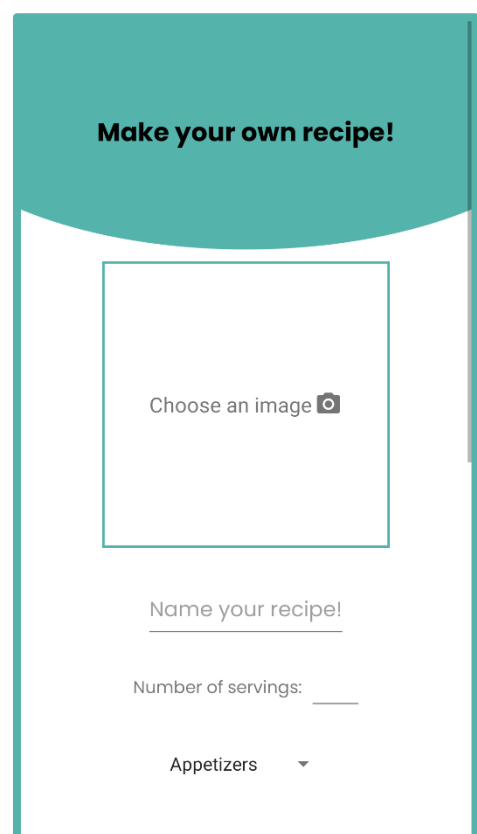
Email address

Password


Reenter password

REGISTER

## Add Recipe



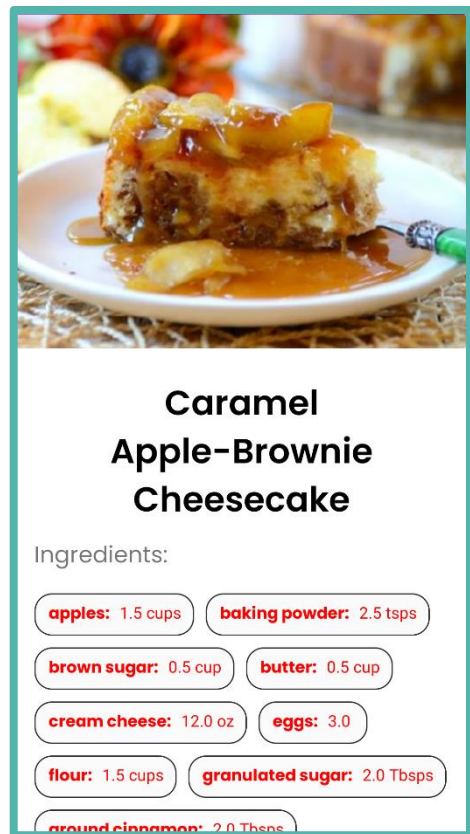
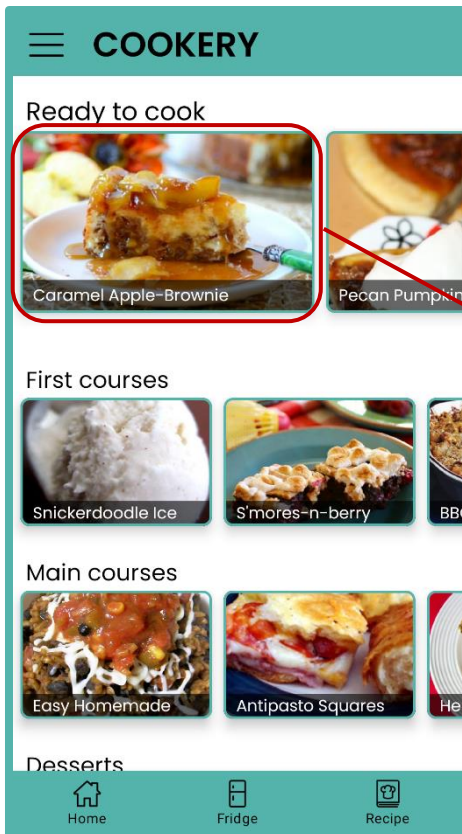
**Make your own recipe!**

Choose an image 

Name your recipe!

Number of servings: \_\_\_\_\_

Appetizers ▾



# Implementazioni future

- **Cerca ricette:** Implementazione di una barra di ricerca che permetterà all'utente di interrogare applicazione in base al nome della ricetta cercata.
- **Lista della spesa:** Possibilità di creare una lista della spesa della ricetta di cui non si dispongono gli ingredienti nella dispensa.
- **Lista di ricette:** Possibilità di creare una lista di ricette selezionate dall'utente per un evento: "cerimonia, festa di compleanno ecc.", l'applicazione potrà consigliare le ricette adatte all'occorrenza o sarà l'utente a scegliere le varie ricette da inserire nel menù dell'evento, sarà possibile generare una lista della spesa di tale insieme di ricette per organizzare e gestire al meglio gli ingredienti mancanti.
- **Ingredienti in scadenza:** Monitoraggio dei prodotti inseriti in dispensa, registrando la data di inserimento nella dispensa, cookery evidenzia gli ingredienti che sono da tempo in dispensa e potrebbero scadere, le ricette consigliate daranno priorità al consumo di tali ingredienti per prevenire gli sprechi.
- **Pantry personalizzata:** Attualmente la dispensa è suddivisa in Pantry, Frigge e Freezer, una futura implementazione prevede la possibilità di lasciare all'utente la possibilità di gestire la suddivisione della sua dispensa come meglio preferisce.

# Strategie di remunerazione

La **remunerazione** dell'applicazione può essere ottenuta aggiungendo una utenza premium grazie alla quale l'utente premium dovrà pagare una quota di iscrizione mensile che gli permetterà di accedere a funzionalità avanzate come:

- Accostamento di vini per le ricette (futura implementazione)
- Creazione di liste della spesa (futura implementazione)
- Creazione di pranzi o cene personalizzate con la possibilità di dare un nome personalizzato e inserire tutte le ricette da cucinare, con creazione automatica della lista della spesa (futura implementazione)
- Suggerimento di negozi per l'acquisto dei vari ingredienti (futura implementazione)

Altra strategia di remunerazione è l'inserimento di annunci pubblicitari che risultino poco invasivi per non influenzare negativamente la user experience