DOCUMENTAZIONE DEI METODI E DELLE CLASSI

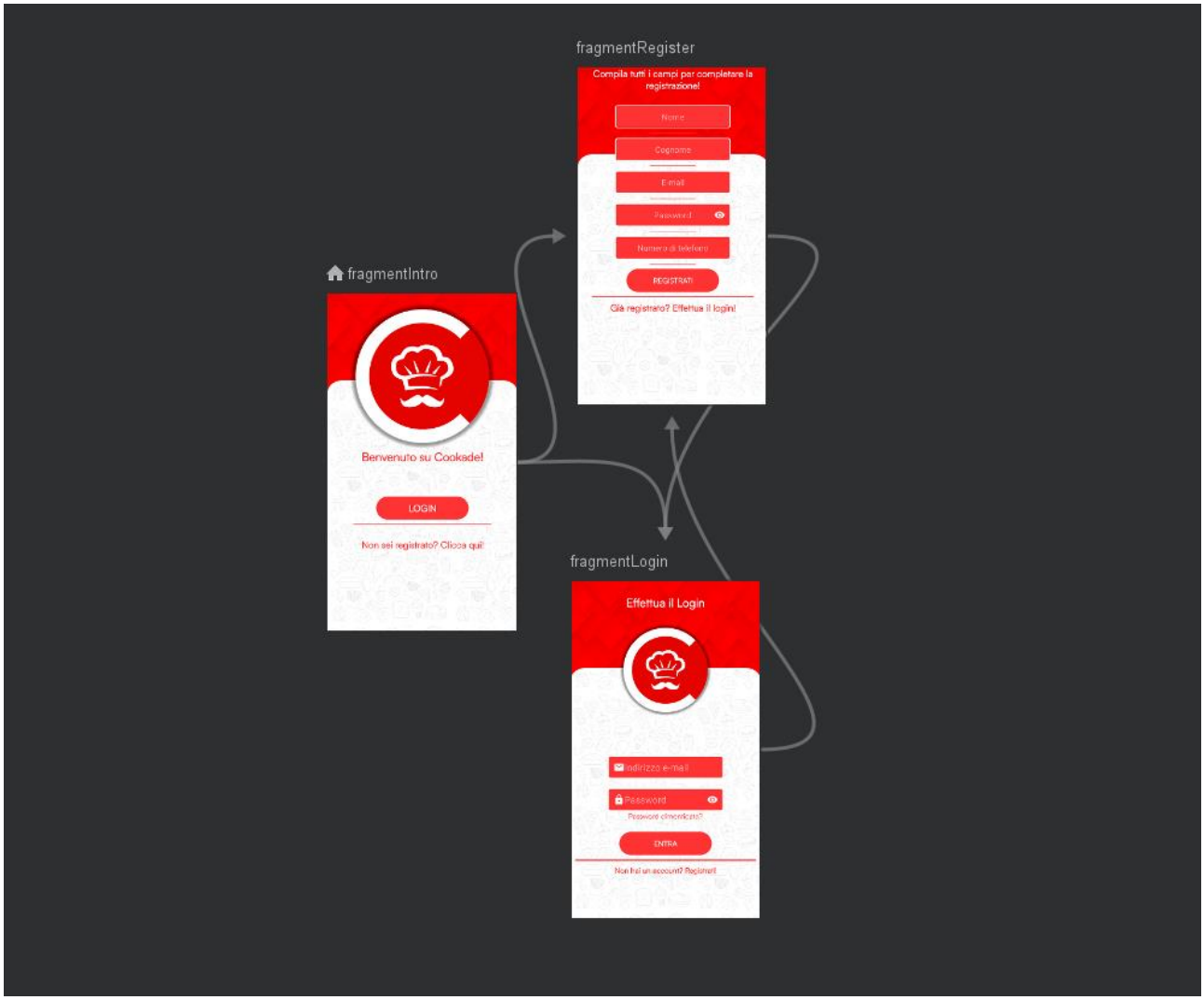
CREDENZIALI INIZIALI:

- Utente di livello 1:
  E-mail: useruser@gmail.com
  Password: useruser
- Utente di livello 2:
  E-mail: dipdip@gmail.com
  Password: dipdip
- Utente di livello 3:
  E-mail: admadm@gmail.com
  Password: admadm

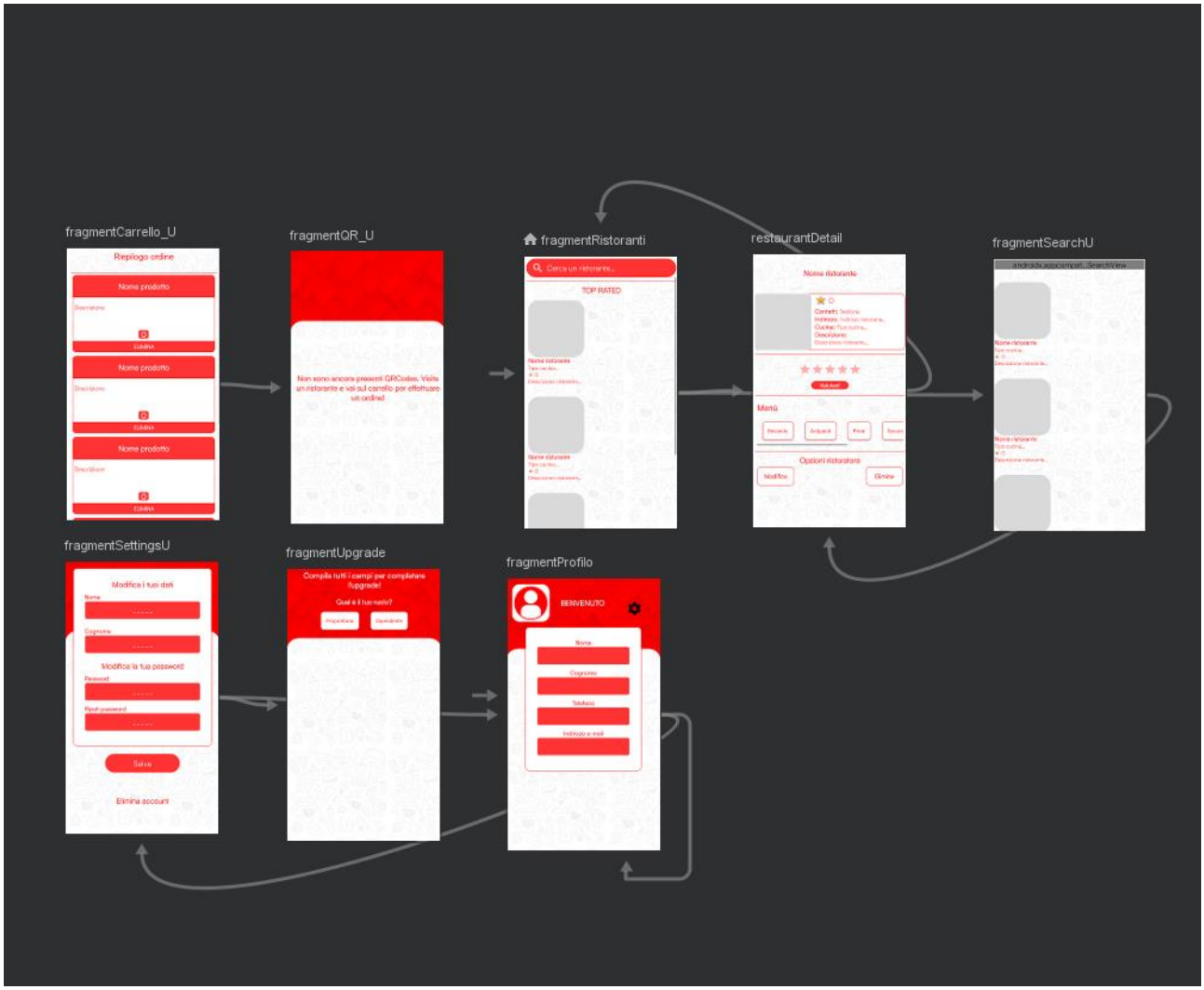
SVILUPPO IN ANDROID

GRAFICI DI NAVIGAZIONE:

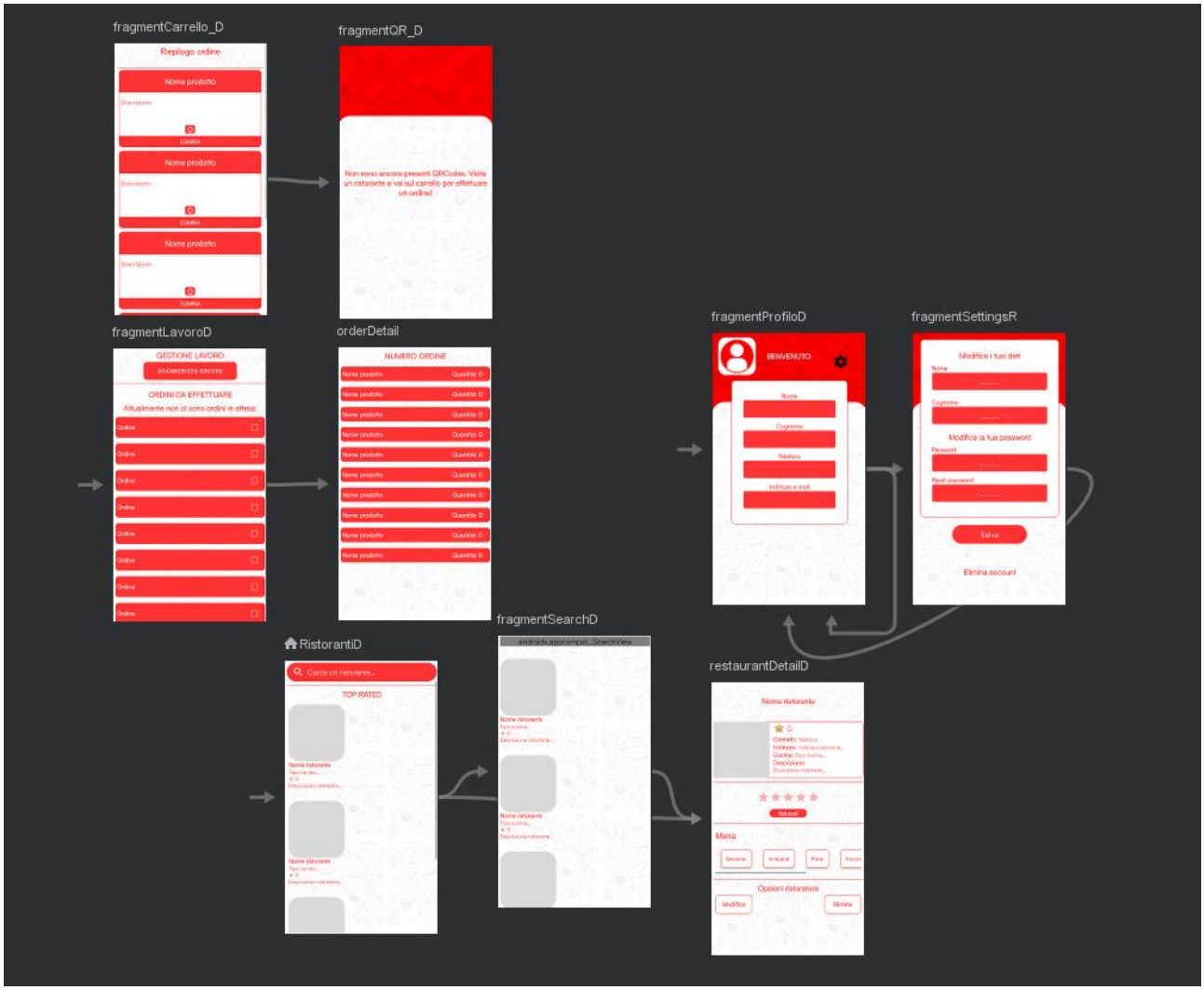
Intro Activity



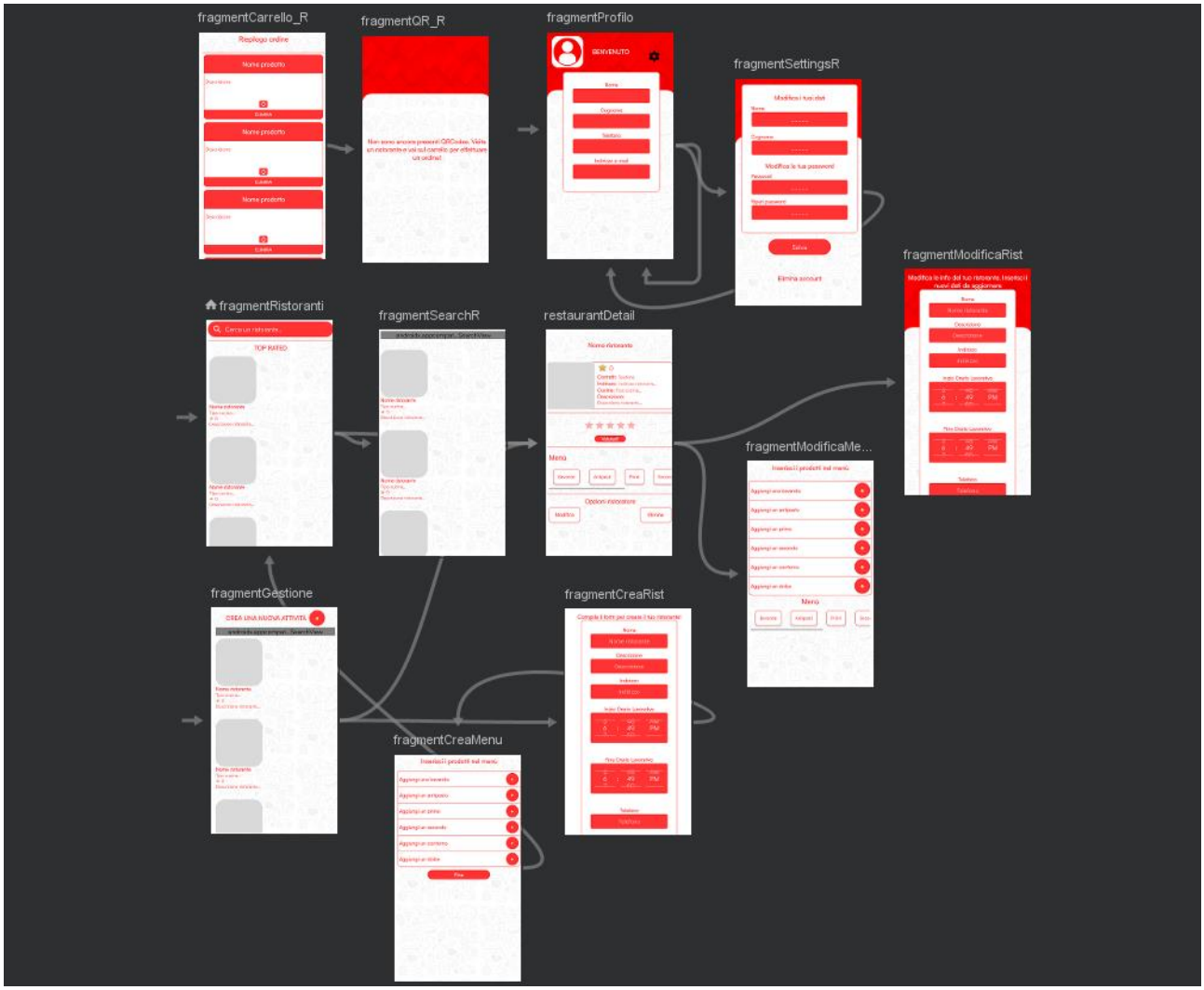
User Activity



Employee Activity



Restaurateur Activity



In tutte le Activity:

- viene inizializzato il NavController per gestire la navigazione dalle pagine ai propri fragment.
- nella funzione onDispatchTouchEvent() viene gestito il tocco fuori dagli elementi in focus.
- tranne per la IntroActivity, vengono gestiti i tocchi sulla navbar.
- tranne per la IntroActivity, è presente la gestione del tasto indietro nell'AppBar tramite l'override della funzione onBackPressed().
- tramite le funzioni onCreateOptionsMenu() e onOptionsItemSelected() viene inizializzato un menù in cui è possibile navigare verso la visualizzazione del carrello e del QRCode.
- tranne per la IntroActivity, si ricevono i dati passati dal FragmentIntro.

DESCRIZIONE DELLE CLASSI E DEI METODI:

UserActivity

Nella navbar si hanno 3 bottoni: uno per la visualizzazione dei ristoranti, uno per la visualizzazione del profilo (passando un bundle contenente i dati dell'utente) e uno per il logout.

EmployeeActivity

Nella navbar si hanno 4 bottoni: uno per la visualizzazione dei ristoranti, uno per la gestione del proprio lavoro, uno per la visualizzazione del profilo (passando un bundle contenente i dati dell'utente) e uno per il logout.

RestaurateurActivity

Nella navbar si hanno 4 bottoni: uno per la visualizzazione dei ristoranti, uno per la gestione dei propri ristoranti, uno per la visualizzazione del profilo (passando un bundle contenente i dati dell'utente) e uno per il logout.

IntroActivity

Questa activity consente di navigare tra le pagine per effettuare il login o la registrazione.

AUTHENTICATION

FragmentIntro

Nell' onViewCreated() del FragmentIntro viene settato il bottone "LOGIN" in modo che, dopo aver controllato che l'user non sia nullo, e quindi già loggato in precedenza, esegue getUserData(), che prende dal database i dati dell'utente loggato. Dopo aver richiamato getQRData(), quindi aver ottenuto i dati del carrello dell' utente, e getRestaurantData(), che riceve i dati di tutti i ristoranti, viene inizializzata una nuova Activity a seconda del livello utente precedentemente ottenuto, passando come argomenti i dati dello user, del carrello e dei ristoranti. Se di livello 1 viene inizializzata la UserActivity, se di livello 2 la EmployeeActivity e se di livello 3 la RestaurateurActivity.

Nel caso in cui non ci si era loggati in precedenza, verificando che l'utente loggato è nullo, è possibile essere reindirizzati alla pagina di login.

Viene anche gestito il tocco sul link per la registrazione nel caso in cui non lo si è già, navigando alla pagina di login.

FragmentLogin

Al click sul tasto “ENTRA” prende i dati dal form, ed esegue signInWithEmailAndPassword() di FirebaseAuth, permettendo il login con l’email e password. Se non si riscontrano problemi si effettua lo stesso procedimento del tasto “LOGIN” precedentemente descritto sul FragmentIntro; nel caso in cui e-mail e password non corrispondono, verrà gestito l’errore.

Sono presenti anche le gestioni per i tasti del recupero password e per la navigazione verso la pagina del FragmentRegister nel caso in cui non si è registrati e lo si voglia fare.

FragmentRegister

Vengono gestiti gli errori dati quando i requisiti minimi per il completamento del form, in cui vengono registrati i propri dati, non vengono rispettati (ad esempio se il nome è superiore a 20 caratteri verrà mostrato il messaggio "Il nome non può essere lungo più di 20 caratteri."). Inoltre, viene effettuato il controllo per garantire che nessun campo venga lasciato vuoto al momento dell’invio del form; se quest’ultimo controllo è andato a buon fine, utilizzando createUserWithEmailAndPassword di FirebaseAuth, si andranno a registrare le credenziali d’accesso in Firebase. Successivamente, solo se la registrazione è andata a buon fine si creerà una istanza di User con i dati precedentemente inseriti nel form, per poi andare a creare un record in firebase nel documento “Utenti” dove si andranno a registrare i dati dell’istanza. Infine, si verrà reindirizzati alla IntroActivity e quindi al FragmentIntro dove è possibile entrare con le proprie credenziali.

UTILS

RestaurantUtils

La funzione createRestaurant(), ricevendo come argomento una istanza di Restaurant, va a salvare i dati del ristorante ricevuto nel path corretto sfruttando l’id all’interno dell’istanza.

La funzione getRatings(), ricevendo come argomento l’id di un ristorante, da come risultato, tramite una response, la lista delle recensioni fatte sul ristorante corrispondente.

La funzione getRestaurantData() ritorna come risultato la lista dei ristoranti presenti nel database.

ProductUtils

La funzione getProdotti() prende come parametri, oltre il context, l’id del ristorante, la categoria della portata (antipasti, primi, ecc.) e la callback di tipo FireBaseCallbackProdotto, permettendo di prendere tutti i prodotti a seconda del tipo di categoria data come parametro.

La funzione addProdotto() permette di aggiungere i dati di un nuovo prodotto, passati come parametro, nel database tramite un path composto anche dall’id del ristorante e dalla categoria, anch’essi passati come parametro.

La funzione deleteProd(), con stessa metodologia di creazione del path precedente, permette di rimuovere un prodotto, passato come parametro, dal database.

La funzione modifyProd() prende come parametri i dati di un prodotto da aggiornare, l’id del ristorante e la categoria del prodotto, permettendo di andare nel path giusto ad aggiornare il prodotto passato con dati ricevuti dall’AlertDialog creato quando la funzione è richiamata.

UserUtils

La funzione deleteUserData() elimina sia le credenziali dell'utente loggato da FirebaseAuth sia i suoi dati dal database.

La funzione updateUserData() riceve una HashMap contenente i dati da aggiornare e fa l'update sul database.

La funzione updateUserPassword() riceve la nuova password da aggiornare in FirebaseAuth e fa l'upgrade.

La funzione recoverUserPassword() riceve l'email dell'utente del quale si deve recuperare la password e sfrutta la funzione sendPasswordResetEmail() di FIrebaseAuth per inviare una mail all'email dell'utente dove è possibile effettuare il reset della password.

La funzione getUserData() riceve una interfaccia FirebaseCallbackUser che ha come metodo onResponse(), che prende come parametro una classe ResponseUser. Dopo aver fatto il collegamento con il database si crea una variabile di tipo ResponseUser dove si registrano i dati presi dallo snapshot. Infine, si inserisce la ResponseUser nella funzione onResponse dell'interfaccia.

FiltriUtils

La funzione searchFilter() riceve in input una lista di ristoranti e una CharSequence, poi si controlla se la CharSequence è nulla, se non lo è allora va a ricercare in quali ristoranti è contenuta la CharSequence nel nome e li mette in un'altra lista, che verrà poi ritornata come risultato.

La funzione typeFilter() riceve una lista di ristoranti e una stringa. Effettuando un controllo su quest'ultima si verificheranno 3 casi:

- se la stringa è "rating" verrà restituita una lista di ristoranti in cui la valutazione di ognuno di essi è maggiore di 3;
- se la stringa è "vegan" verrà restituita una lista di ristoranti che offrono anche tipologie di cibo vegan;
- se la stringa è diversa dalle due precedenti, si andrà ad effettuare una ricerca a seconda del tipo di cibo offerto dal ristorante, verificando che la stringa passata è contenuta nel record TipoCibo di ogni ristorante, restituendo come risultato una lista di solo ristoranti che offrono la categoria di cibo corrispondente alla stringa passata come parametro.

CartUtils

La funzione getQRData() riceve come parametri l'id dell'utente e una callback di FirebaseCallbackCart. Dopo la connessione al database viene salvato lo snapshot come json.

La funzione addQRData() riceve come parametri una lista di prodotti e l'id dell'utente, successivamente, dopo aver effettuato l'accesso al database, vengono inseriti i dati di tutti i prodotti ricevuti nel record del Cart in FIrebase.

DipendenteUtils

La funzione createDipendente() crea un record in "Dipendenti" su Firebase con i dati ricevuti negli argomenti.

La funzione getDipendenteData() riceve come argomenti l'email del dipendente e la callback di tipo FireBaseCallbackDipendente che ha come metodo onResponse(), che prende come parametro una classe ResponseDipendente. Dopo aver fatto il collegamento con il database si crea una variabile di tipo ResponseDipendente e in seguito scorre tutti i dipendenti fin quando l'e-mail ricevuta come argomento corrisponde con l'e-mail del dipendente dello snapshot. In questo caso viene creata un'istanza di

Dipendente in cui verranno messi tutti i dati del dipendente corrente, per poi metterla nella response precedentemente creata, per poi poterla riutilizzare tramite la callback.

OrderUtils

La funzione createOrder() riceve come parametri l'id del ristorante e una stringa json. Dopo aver effettuato l'accesso al database, si trova prima il numero di ordini totale e lo si salva su una variabile, poi si crea una HashMap dove salvare tutti i dati del nuovo ordine. A questo punto si andrà a salvare tutti i record nel database.

La funzione getOrders() riceve come parametri un'istanza di un dipendente e una callback di tipo FireBaseCallbackOrder che ha come metodo onResponse(), che prende come parametro una classe ResponseOrder. Dopo aver fatto il collegamento con il database si crea una variabile di tipo ResponseOrder. In seguito, viene fatto scorrere tutto lo snapshot, e per ogni elemento viene creata una istanza di Order, che verrà poi aggiunta alla response, che a fine ciclo avrà tutti gli ordini. Infine, si inserisce la response nella funzione onResponse dell'interfaccia.

La funzione deleteOrders() riceve come parametro l'id del ristorante e dopo aver effettuato l'accesso al database cancella tutti gli ordini presenti per il ristorante corrispondente utilizzando removeValue() di FirebaseDatabase.

FRAGMENTS COMUNI

FragmentRistoranti

Nell' onCreateView():

- viene inizialmente preso, tramite getUserData(), il livello dell'utente loggato, poi vengono caricati tutti i ristoranti tramite getRestaurantData(), mettendoli in una RecyclerView e filtrandoli tramite la funzione typeFilter();
- viene gestito il refresh con lo swipe verso il basso, ricaricando e aggiornando i ristoranti;
- viene gestito il tocco sulla SearchBar, che permetterà di passare al FragmentSearch.

Nell' onViewCreated() a seconda del bottone cliccato verranno mostrati i ristoranti che comprendono la tipologia di cibo scelta, utilizzando, come in precedenza, la funzione typeFilter().

La funzione onClickRestaurant() permette di navigare verso il fragment RestaurantDetail, passando un bundle contenente l'id del ristorante cliccato, la lista dei ristoranti e il livello dell'utente loggato.

RestaurantAdapter

Permette di comunicare al ViewHolder, tramite bindRestaurants(), l'istanza del ristorante corrispondente alla posizione nella RecyclerView in cui l'utente ha cliccato.

RestaurantViewHolder

Ricevendo il ristorante dall'Adapter, vengono messe le informazioni all'interno della card card_restaurant e inoltre viene richiamata onClickRestaurant() (definita su FragmentRistoranti) al momento in cui la card viene cliccata, passando come argomento il ristorante attuale.

RestaurantDetail

Nell' onCreateView() ,dopo aver ricevuto i dati del bundle, viene effettuato un controllo che verifica se il ristorante non è nullo, se l'utente è di livello 3 e se si è il proprietario del ristorante in questione; se sì,

vengono visualizzati i tasti per la modifica del menù, della modifica dei dati del ristorante e per l'eliminazione del ristorante.

In seguito, vengono caricati e mostrati i dati inerenti al ristorante da visualizzare e caricando il relativo menù.

Nell' onViewCreated() si gestiscono per lo più i tocchi ai bottoni:

- il bottone Modifica nel menù inserisce in un bundle tutti i dati del menù del ristorante e si effettua la navigazione verso il fragment FragmentModificaMenu;
- i bottoni delle sezioni del menù permettono la visualizzazione della sezione del menù a seconda delle categorie scelte tramite una RecyclerView, utilizzando ProductAdapter e ProductViewHolder;
- il bottone Modifica inerente al ristorante inserisce in un bundle i dati del ristorante e si effettua la navigazione verso il fragment FragmentModificaRist;
- il bottone Elimina permette l'eliminazione del ristorante dal database, chiedendo la conferma tramite un AlertDialog;
- il bottone per la valutazione di un ristorante che, a seconda di quante stelle sono opzionate, inserisce una valutazione nel database sia nel documento del ristorante, sia nel documento dell'utente memorizzando le recensioni fatte per ogni ristorante votato.

Richiamando la funzione getRatings() è possibile ricevere tutte le valutazioni, per poi sommarle e farne una media, assegnando il valore al campo ratingR del ristorante.

La funzione restaurantFromId() permette, passando come parametro l'ID di un ristorante, di ricevere un istanza del ristorante corrispondente all' ID passato.

ProductAdapter

Riceve come parametri, oltre al context, la lista dei prodotti, il CartViewModel e l'id del ristorante di cui fanno parte i prodotti.

Nell' onCreateViewHolder() viene passato al ViewHolder in costruzione il binding della card in cui saranno inseriti i dati del prodotto e il CartViewModel.

Nell' onBindViewHolder() con bindProdotti() vengono passati i dati del prodotto corrispondente alla posizione nella RecyclerView in cui l'utente ha cliccato. Inoltre, tramite createShoppingCart(), oltre al context, vengono passati lo stesso prodotto di bindProdotti() e l'id del ristorante.

ProductViewHolder

La funzione bindProdotti() inserisce i dati nella card.

La funzione createShoppingCart() gestisce il tocco sul tasto aggiungi della card. Inizialmente prende in input la quantità dalla card e la registra nella variabile quantity, se non è zero allora viene creata una istanza del model CartProduct con tutti i dati del prodotto selezionato, calcolando anche il prezzo totale per quel prodotto. Successivamente si va ad aggiungere il prodotto al CartViewModel, effettuando il controllo che permette di non aggiungere prodotti di diversi ristoranti allo stesso carrello, gestendo anche il caso in cui aggiungendo un prodotto di un ristorante diverso, dando la possibilità tramite un AlertDialog di rimuovere i dati precedenti del carrello creandone uno nuovo con i prodotti del nuovo ristorante.

FragmentProfilo

Nell' onCreateView(), tramite la funzione getUserData(), prende tutti i dati dell'utente dal database passando come parametro un oggetto di tipo FireBaseCallbackUser, ed effettuando un override della

funzione onResponse possiamo accedere alla response, per poi inserirli negli elementi della pagina xml corrispondenti.

Nell' onViewCreated() si può gestire il tocco nell'immagine del profilo, permettendo di aprire la galleria e scegliere una nuova foto nel caso in cui la si voglia cambiare. Si gestisce inoltre il tocco sul simbolo delle impostazioni che porterà ad una form in cui è possibile modificare i propri dati, navigando quindi verso FragmentSettings.

La funzione getImageId() riceve una stringa riferita al nome dell'immagine e genera un id associato ad essa.

FragmentSettings

Nell' onCreateView(), oltre a ricevere i dati del bundle, si rende visibile il tasto per fare l'upgrade da utente di livello 1 a dipendente o ristoratore solo se si è di livello 1.

Nell' onViewCreated() si gestiscono i tocchi a tasti come:

- il tasto per fare l'upgrade dell'account, che richiama updateUserData() passando i nuovi dati e che successivamente reindirizza verso il fragment FragmentUpgrade;
- il tasto salva, che modifica i dati dell'utente nel database prendendoli da quelli inseriti nel form, riportando poi alla pagina del profilo;
- il tasto per eliminare l'account, che richiamando deleteUserData() permette l'eliminazione dell'account.

FragmentSearch

Nell' onCreateView() grazie all'override della funzione onQueryTextChange() si può fare la ricerca del ristorante desiderato sfruttando la funzione getRestaurantData() e richiamando poi la funzione searchFilter() alla quale verranno passati la lista dei ristoranti e il testo che si sta immettendo nella barra.

Inoltre viene fatto l'override della funzione onClickRestaurant() dove viene gestito il tocco ad un ristorante trovato tramite la ricerca con la SearchBar, permettendo la navigazione alla pagina di dettaglio di esso.

FragmentCarrello

Nell' onCreateView(), dopo aver ricevuto i dati dal bundle e aver fatto qualche controllo per la grafica, e dopo aver inizializzato il cartViewModel, viene richiamata la funzione getcartItem() per ricevere i dati già presenti nel ViewModel e grazie ad observe è possibile gestire i cambiamenti di esso in tempo reale, infine vengono presi i prezzi di tutti i prodotti al suo interno e sommati per ottenere il conto totale. Inoltre, verranno visualizzati i prodotti presenti all'interno del carrello grazie ad una RecyclerView che sfrutta CartAdapter e CartViewHolder.

Nell' onViewCreated() viene gestito il tocco sul bottone per generare il QRCode, chiamando la funzione addQRData() passando come parametri la lista dei prodotti e l'id dell'utente loggato. In seguito, viene svuotato il ViewModel visto che il codice è stato generato. Infine, si viene reindirizzati alla pagina della visualizzazione del QRCode.

CartAdapter

Riceve come parametri una lista dei prodotti e il ViewModel.

La funzione setData() riceve la lista di prodotti e la inserisce all'interno della classe osservando gli eventuali cambiamenti.

Nell' onCreateViewHolder() viene passato al ViewHolder in costruzione il binding della card in cui saranno inseriti i dati del prodotto.

Nell' onBindViewHolder() con bindCart() vengono passati al ViewHolder:

- i dati del prodotto corrispondente alla posizione nella RecyclerView in cui l'utente ha cliccato;
- Il cartViewModel;
- Il binding della pagina del carrello.

CartViewHolder

La funzione bindCart() inserisce i dati del prodotto ricevuto nella card di un elemento della RecyclerView. Inoltre, gestisce il click nel bottone per l'eliminazione di un prodotto, generando un AlertDialog per la conferma e in caso affermativo viene richiamata deleteSpecificCart() del ViewModel che elimina l'elemento selezionato. Se il carrello rimane vuoto richiama la funzione deleteCartItems() e aggiusta la grafica della pagina del carrello.

FragmentQR

Nell' onCreateView() viene richiamata la funzione getQRData() che prende tutti i dati del carrello, passando come parametri l'id dell'utente loggato e un oggetto di tipo FireBaseCallBackCart ed effettuando un override della funzione onResponse possiamo accedere alla response. A questo punto, si controlla che il carrello nella response non sia nullo e in questo caso si andrà a creare il QRCode con all'interno i dati della response.

LIVELLO 1 - UTENTE BASE

FragmentUpgrade

Nell' onViewCreated() si gestiscono i tocchi ai due RadioButton presenti nell'xml, ", i quali permettono di effettuare l'inflate della View in un container:

- bottone "Proprietario", immette nel container il fragment FragmentCreaRist, passando tramite bundle l'utente;
- bottone "Dipendente", immette nel container il fragment FragmentUpgradeDipendente, passando tramite bundle l'utente;

FragmentUpgradeDipendente

Nell' onViewCreated(), dopo aver ricevuto i dati dal bundle, viene principalmente gestito il tocco sul tasto "Upgrade". Ricevendo da input i dati necessari si va a richiamare getUserData(), per poi andare ad aggiornare il livello dell'utente a 2, così da risultare un dipendente, e poi si richiama updateUserData() per inserirlo nel database. Successivamente viene creata un'istanza di Dipendente e viene richiamata createDipendente(), passando i dati della nuova istanza. Infine, si avvia la nuova activity (EmployeeActivity).

LIVELLO 2 - DIPENDENTE

FragmentLavoro

Nell' onCreateView() viene richiamata la funzione getDipendenteData() passando come parametri l'email dell'user ricevuto nel bundle e un oggetto di tipo FireBaseCallBackUser, dove, con l'override della funzione onResponse(), è possibile ricevere il dipendente e successivamente si richiama la funzione getOrders(), passando come parametri il dipendente e un oggetto di tipo FireBaseCallbackOrder, dove, con l'override della funzione onResponse(), si ottiene una lista contenente tutti gli ordini inerenti al ristorante corrispondente. Se questa lista non è vuota, allora è possibile mostrarli a schermo in una RecyclerView tramite OrderAdapter e OrderViewHolder.

Nell' onViewCreated() vengono gestiti i tocchi:

- al bottone "Scannerizza QRCode" che avvia il lettore per la scansione del QRCode
- al bottone per l'eliminazione degli ordini, che genera un AlertDialog, che in caso di selezione positiva esegue prima deleteOrders() passando come parametro il codice del ristorante in cui il dipendente lavora, poi getUserData() per poter riavviare l' EmployeeActivity passando in un bundle lo user ricavato dalla response.

L'override della funzione onActivityResult() gestisce i dati ricevuti dallo scanner del QRCode, trasformandoli in json. Inoltre, prendendo il codice del ristorante dall'ordine è possibile controllare se il lavoratore ha effettivamente eseguito lo scan di un QRCode riferito al ristorante di cui è dipendente, solo passando questo controllo è possibile richiamare la funzione createOrder() passando come parametri l'id del ristorante e il json dell'ordine.

La funzione onClickOrder() permette, ricevendo come parametro l'ordine, di navigare al dettaglio dell'ordine passando in un bundle la lista degli ordini e il numero dell'ordine selezionato.

OrderAdapter

Nell' onCreateViewHolder() viene passato al ViewHolder in costruzione il binding della card in cui saranno inseriti i dati del prodotto e un clickListener di tipo OrderClickListener

Nell' onBindViewHolder() viene passato l'ordine al ViewHolder

OrderViewHolder

La funzione bindOrders() riceve come parametro un ordine e inserisce i suoi dati nella card della RecyclerView, inoltre permette di mettere una spunta per ogni ordine e memorizzare il check sul database.

OrderDetail

Nell' onCreateView(), dopo aver preso i dati dal bundle prende i dati dal JSONArray e li inserisce uno ad uno un una HashMap, popolando una lista di HashMap, da adattare in seguito in una RecyclerView tramite un OrderProductAdapter e un OrderProductViewHolder

OrderProductAdapter

Nell' onCreateViewHolder() viene passato al ViewHolder in costruzione il binding della card in cui saranno inseriti i dati del prodotto.

Nell' onBindViewHolder() vengono passati i prodotti di un ordine al ViewHolder.

OrderProductViewHolder

La funzione bindCart() inserisce i dati del prodotto ricevuto nella card di un elemento della RecyclerView.

LIVELLO 3 - RISTORATORE

FragmentCreaMenu

Nell' onCreateView, dopo aver preso i dati dal bundle, per ogni bottone, viene richiamata la funzione showDialog(). Inoltre, per il tasto "Fine" viene richiamata la funzione getRestaurantData() a cui si passa un oggetto di tipo FireBaseCallbackRestaurant per poter poi passare la response di tipo ResponseRistorante, contenente la lista dei ristoranti, come bundle insieme allo user al fragment FragmentRistoranti.

La funzione showDialog() riceve come parametri una stringa e il nome del ristorante, viene creato una AlertDialog che permette di inserire nel form i dati di un prodotto da aggiungere, creando un'istanza di

Product contenente i dati immessi. A seconda della stringa ricevuta viene chiamata addProdotto() che andrà ad aggiungere nel database il prodotto nella categoria inerente alla stringa.

FragmentCreaRist

Nell' onCreateView(), si effettua l'inflate di un form per la compilazione dei dati principali di un ristorante. Inizialmente, dopo aver preso i dati dal bundle si inizializza una lista di stringhe che rappresentano le tipologie possibili di cibo di un ristorante. Vengono utilizzate effettuando il click nel bottone Tipologia Cibo, che, aprendo un dialog, permette di selezionare con scelta multipla le differenti tipologie.

Nell' onViewCreated(), quando si effettua il click sul bottone "Crea ristorante", vengono controllati tutti i campi compilati e, se non presentano errori, si procede con la creazione e scrittura del ristorante sul database, attraverso la funzione getUserData(), che ci permette di utilizzare l'utente attualmente connesso per modificare il proprio livello se non è già "3", proseguendo poi con la funzione createRestaurant() per scrivere il ristorante nel database, ed infine richiamare getRestaurantData() per scaricare la lista di ristoranti aggiornata ed effettuando la navigazione verso la pagina FragmentCreaMenu.
Dentro il form è stato utilizzato un widget chiamato TimePicker, quindi la funzione OnClickTime permette di modificare un orario temporaneamente all'interno del fragment.

FragmentGestione

Nell' onCreateView(), dopo aver preso i dati dal bundle si richiama la funzione getRestaurantData() per ottenere la lista dei ristoranti appartenenti all'utente loggato e vengono inseriti all'interno di una recyclerview. Successivamente, effettuando il refresh della pagina, verrà aggiornata la suddetta lista effettuando una nuova query nel database. Grazie all'override della funzione onQueryTextChange() si può fare la ricerca del ristorante desiderato sfruttando la funzione getRestaurantData() e richiamando poi la funzione searchFilter() alla quale verranno passati la lista dei ristoranti e il testo che si sta immettendo nella barra.

Nell' onViewCreated() invece, quando si clicca sul bottone "Crea ristorante" verrà effettuata la navigazione al fragment FragmentCreaRist.

Infine, la funzione onClickRestaurant() prende come parametro un ristorante specifico ed effettua la navigazione verso il fragment RestaurantDetail relativo a quel ristorante.

FragmentModificaMenu

Nell' onCreateView(), oltre alle stesse funzionalità del fragment FragmentModificaMenu, vengono inizializzate le liste dei prodotti già esistenti nel menù del ristorante.

Nell' onViewCreated(), a seconda del bottone cliccato imposta una RecyclerView tramite la funzione bindrecyclerview(), che sfrutta ProductEMAdapter e ProductEMViewHolder, inserendo i prodotti che vengono passati come parametri.

ProductEMAdapter

La funzione setData() adatta la lista e controlla se sono stati effettuati dei cambiamenti.

Nell' onCreateViewHolder() viene passato al ViewHolder in costruzione il binding della card in cui saranno inseriti i dati del prodotto.

Nell' onBindViewHolder() vengono passati i dati di un prodotto, l'id del ristorante e il tipo del prodotto al ViewHolder.

ProductEMViewHolder

La funzione bindProdotti() riceve come parametri i dati di un prodotto, l'id del ristorante e il tipo del prodotto e li adatta alla card. Inoltre, gestisce i tocchi ai bottoni:

- "Elimina", richiamando la funzione deleteProd() passando il prodotto da eliminare
- "Modifica", richiamando la funzione modifyProd() passando il prodotto da modificare

FragmentModificaRist

Questo fragment permette all'utente di modificare i campi di un ristorante già creato con le stesse funzionalità del fragment FragmentCreaRist, con la sola differenza che la query verso il database, invece di creare un nuovo ristorante, effettua l'update sui campi modificati.

SVILUPPO IN FLUTTER

Main

La funzione main() è asincrona: inizializza la connessione con il database di Firebase avendo come opzioni quelle di default, specificate nella classe "firebase_options", e al suo termine l'applicazione viene avviata.
La classe MyApp crea un Widget in cui ritorna un ChangeNotifierProvider, che inizializza il CartProvider. Inoltre, viene dato il titolo all'applicazione e viene impostata come home la classe HomePage.
La classe HomePageState estende lo stato della classe HomePage. Nella funzione _initializeFirebase(), asincrona, restituiamo una variabile di tipo FirebaseApp. Nel widget costruiamo un FutureBuilder passando come future la funzione _initializeFirebase() e, se la connessione è stata effettuata, mandiamo in view la classe "PageIntro". In caso contrario, un indicatore a forma di cerchio comparirà a schermo fin quando la connessione non è stata completata.

CartProvider

Viene inizializzata una lista di prodotti del carrello, che può essere osservata all'interno dell'applicazione grazie al ChangeNotifier.
La funzione addProduct() permette di aggiungere alla lista un nuovo prodotto, passandogli come parametro un CartProductModel ed aggiornando successivamente la lista.
La funzione removeProduct() rimuove dalla lista un prodotto specifico, passandoglielo come parametro ed aggiornando successivamente la lista.
La funzione clearData() rimuove tutti i prodotti all'interno della lista.

PageIntro

Restituisce un widget in cui è presente uno StreamBuilder che controlla se un utente ha già effettuato l'autenticazione in passato: in caso affermativo, effettuando il click sul bottone "Login", la navigazione porterà alla classe PageRistoranti; in caso contrario, la navigazione porterà alla classe PageLogin. Inoltre, cliccando sulla scritta "Non sei registrato?" verrà effettuata la navigazione alla classe PageRegistrati.

PageLogin

La funzione loginUser(), di tipo asincrona, prende come parametri E-mail e Password e provano ad effettuare una richiesta sul database attraverso la funzione signInWithEmailAndPassword: con esito positivo si effettuerà la navigazione alla classe PageRistoranti, in caso contrario verrà mostrato a schermo un errore. Il widget di questa classe si compone di due edittext "Email" e "Password" i cui controller vengono passati alla funzione loginUser() al click del tasto "Entra". Come per la classe PageIntro, cliccando sulla scritta "Non sei registrato?" verrà effettuata la navigazione alla classe PageRegistrati.

PageRegister

La classe PageRegisterState controlla lo stato della classe PageRegister.
La funzione writeUserToDB(), asincrona, prende come parametri delle stringhe (nome, cognome, email, password e telefono) e, convertendoli in una mappa, cerca di scriverla all'interno del database, ritornando poi alla classe PageLogin.
La funzione registerUser(), invece, effettua la registrazione con i soli campi di email e password attraverso la funzione di Firebase createUserWithEmailAndPassword, controllando che l'email già non esista.
Nel widget sono quindi presenti i vari edittext per la registrazione con i relativi controller. Cliccando nel bottone "Registrati" verranno effettuati dei controlli ed in seguito eseguite le due funzioni precedenti.
Infine, cliccando sulla scritta "Già registrato?" verrà effettuata la navigazione alla classe PageLogin.

PageRistoranti

La classe PageRistorantiState controlla lo stato della classe PageRistoranti.
La funzione _onItemTapped() crea una bottom navigation bar,ricevendo un index (numero) come parametro che, a seconda di esso, porterà a classi diverse:
- Index 1: PageRistoranti
- Index 2: PageProfilo
- Index 3: Effettua il Logout -> PageIntro

Nel widget è presente innanzitutto una Search Bar non funzionale, che se cliccata porterà alla classe PageSearch, passandole come argomento la lista dei ristoranti. Dopo di ciò, grazie ad un ListView.builder, viene creata una lista di ristoranti filtrati per ratings maggiore di 3.5, in cui ogni elemento viene visualizzato a schermo tramite l'utilizzo del Widget "CardRistorante". Grazie all'InkWell, quando si effettua un click su una Card, si effettuerà la navigazione alla classe PageRestaurantDetail.
Inoltre, è stato utilizzato un widget "ChoiceChip" per effettuare operazioni di filtraggio in base alla tipologia di cibo del ristorante. Quindi, premendo un bottone verrà effettuata una query sul database e verrà restituita una lista specifica di ristoranti. Infine, grazie ad un ListView.builder, viene creata un'altra lista di ristoranti, che questa volta vengono filtrati a seconda del bottone premuto sul widget ChoiceChip.
Tutti e tre i widget vengono osservati grazie ad "Obx()", che permette di capire se i valori delle liste e dei bottoni sono cambiati all'interno dell'applicazione. Vengono utilizzate le due classi scritte in seguito per eseguire l'osservazione.

ChipController:

Comprende due classi che estendono GetxController e permettono ai ChoiceChip di ascoltare, cambiare ed osservare il click su un bottone premuto all'interno della lista grazie a metodi di get e set.

FirebaseControllerRestaurants:

Anche questa classe estende GetxController.
La funzione getRestaurantData effettua delle richieste a Firebase per scaricare la lista dei ristoranti e filtrarli attraverso il parametro "Tipologia", di tipo Filter (enum) passatogli. Lo snapshot scaricato dal database viene convertito in una lista di RestaurantModel e la variabile restaurantList all'interno della classe viene aggiornata.

PageSearch

La classe PageSearchState controlla lo stato della classe PageSearch.
La funzione _onItemTapped() crea una bottom navigation bar uguale a quella della PageRistoranti.

Il widget, invece, è costituito da una Search Bar funzionante, dove viene osservato se vengono inseriti o rimossi dei caratteri. Quando la sequenza è vuota, la lista non comparirà; altrimenti, attraverso l'utilizzo di un ListView.builder, come per PageRistoranti, viene creata una lista di ristoranti filtrati secondo se la sequenza compare nel nome, in cui ogni elemento viene visualizzato a schermo tramite l'utilizzo del Widget "CardRistorante". Grazie all'InkWell, quando si effettua un click su una Card, si effettuerà la navigazione alla classe PageRestaurantDetail.

PageRestaurantDetail

La classe PageRestaurantDetailState controlla lo stato della classe PageRestaurantDetail.
La funzione _onItemTapped() crea una bottom navigation bar uguale a quella delle altre classi.
Nel widget viene creata una card in cui sono posti gli attributi del ristorante, come l'immagine, il nome, i contatti etc. Inoltre, è stato utilizzato anche qui un widget "ChoiceChip" per effettuare operazioni di filtraggio in base alla tipologia dei prodotti all'interno del menu. Quindi, premendo un bottone verrà effettuata una query sul database e verrà restituita una lista specifica di prodotti. Infine, grazie ad un ListView.builder, viene creata una lista di prodotti filtrati quindi a seconda del bottone premuto sul widget ChoiceChip, in cui ogni elemento viene visualizzato a schermo tramite l'utilizzo del Widget "CardProduct". Entrambi i widget vengono osservati grazie ad "Obx()", che permette di capire se i valori delle liste e dei bottoni sono cambiati all'interno dell'applicazione. Vengono utilizzate le classi ChipController e FirebaseControllerMenu.

FirebaseControllerMenu

Anche questa classe estende GetxController.
La funzione getMenuData effettua delle richieste a Firebase per scaricare la lista dei prodotti all'interno del menu di un ristorante specifico e filtrarli attraverso il parametro "Tipologia", di tipo FilterMenu (enum) passatogli. Lo snapshot scaricato dal database viene convertito in una lista di ProductModel e la variabile menuList all'interno della classe viene aggiornata.

PageProfilo

La classe PageProfiloState controlla lo stato della classe PageProfilo.
La funzione getUser(), asincrona, restituisce una mappa corrispondente alle informazioni dell'utente loggato attraverso una richiesta al database, che viene poi trasformata in un UserModel.
La funzione _onItemTapped() crea una bottom navigation bar uguale a quella delle altre classi.
Nel widget le informazioni dell'UserModel sono poste all'interno delle textview per visualizzare a schermo le informazioni. Per scaricare l'immagine profilo si utilizza un FutureBuilder che effettua la connessione allo Storage del database e restituisce il path dell'immagine.

PageCarrello

La classe PageCarrelloState controlla lo stato della classe PageCarrello.
La funzione _onItemTapped() crea una bottom navigation bar uguale a quella delle altre classi.
All'interno del Widget principale, la funzione createCart(), asincrona, permette di salvare nel database un file JSon, che viene creato convertendo la lista dei prodotti nel carrello presente nel CartProvider. Alla fine della funzione la lista del CartProvider viene svuotata.
Il widget cartExists() controlla se la lista dei prodotti nel carrello è vuota o meno: in caso affermativo comparirà un testo che lo esplicita, in caso negativo ogni prodotto all'interno del carrello viene inserito in una lista di CardCartProduct.
Cliccando sul bottone "Genera QR-Code" viene avviata la funzione createCart(), convertendo la lista in un file JSon.
Infine, il widget principale ritornerà in una SafeArea la costruzione del widget cartExists().

PageQR_Code

La classe PageQR_CodeState controlla lo stato della classe PageQR_Code.
La funzione _onItemTapped() crea una bottom navigation bar uguale a quella delle altre classi.
La funzione fetchMapData(), asincrona, effettua una connessione con il database e recupera i dati del carrello dell'utente loggato, restituendo una mappa.
Il widget generateQRCode() converte la mappa in un formato JSon e, se non corrisponde a "null", genera una QrImageView che se scannerizzata rappresenta i dati del carrello utente; in caso contrario, viene mostrata a schermo una scritta che esplicita che non è presente nessun carrello.
Il widget principale ritornerà uno Scaffold che costruisce il widget generateQRCode().