RISTODROID

SVILUPPO DI MOBILE SOFTWARE

A.A. 2019/2020

Sommario

INTRODUZIONE

Ristodroid è un'app sviluppata in ambiente Android, con lo scopo di facilitare il ruolo del "Cameriere", permettendogli quindi di ridurre gli sprechi di tempo nella fase di acquisizione dell'ordinazione.

Abbiamo deciso di separare il sistema in due App distinte, destinate rispettivamente ai due attori.

Ristodroid inoltre necessita di dispositivi come Tablet o Smartphone, che permetteranno al "Cliente" di effettuare l'ordinazione dando la possibilità di consultare direttamente il menù del ristorante.

Questo potrà in totale autonomia, creare l'ordinazione in modo da permettere al Cameriere di riceverla attraverso il pairing dei due dispositivi. In seguito, esso potrà aggiungere maggiori dettagli all'ordine come il numero del tavolo di riferimento, la quantità ed il tipo di coperti.

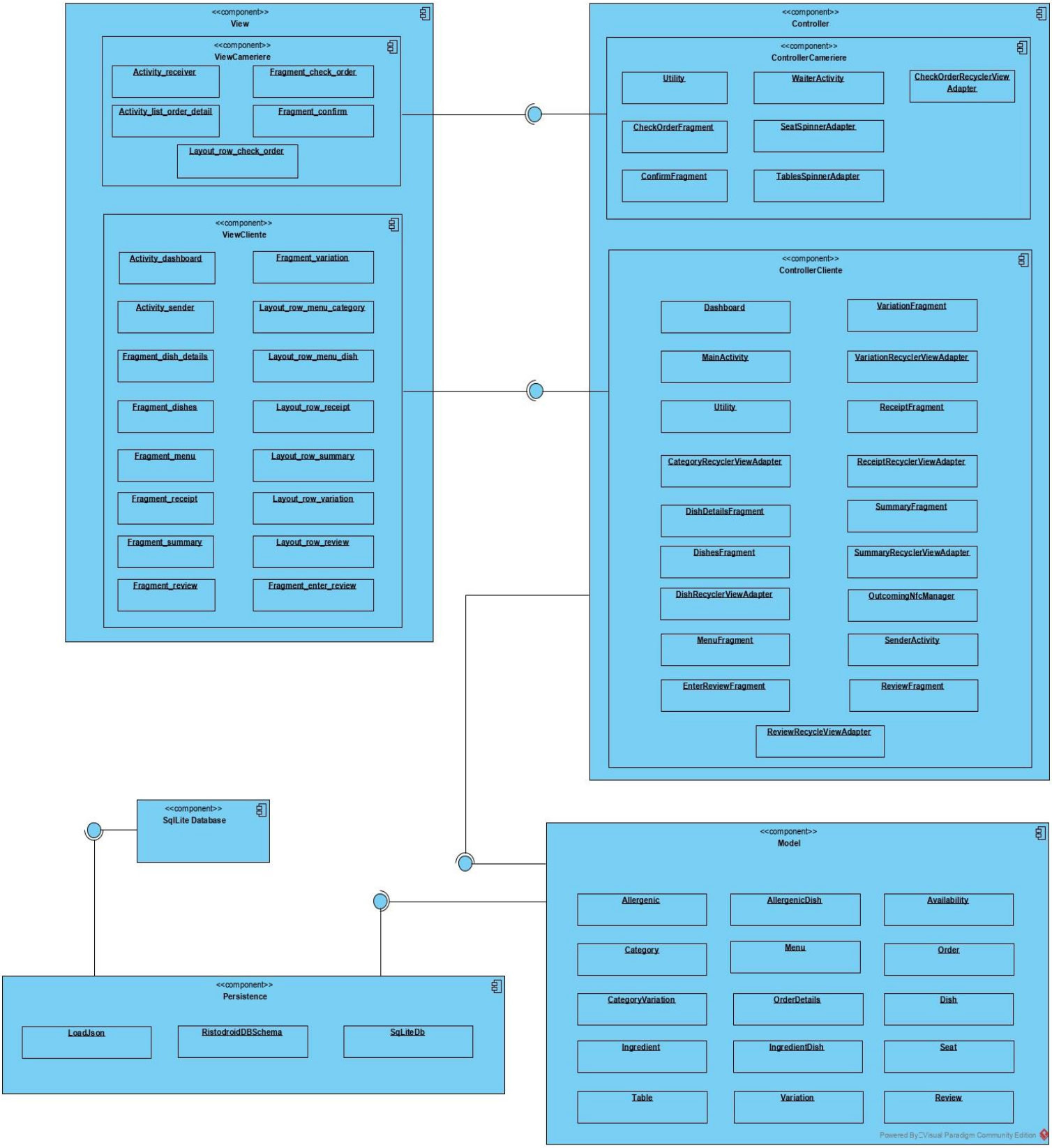
ANALISI E PROGETTAZIONE

REQUISITI

Codice Requisito	Descrizione
RCL-1	Visualizza categoria piatti
RCL-2	Visualizza lista piatti
RCL-3	Visualizza dettagli piatto
RCL-4	Inserimento quantità piatto
RCL-5	Personalizzazione piatto
RCL-6	Visualizza riepilogo
RCL-7	Aggiunta piatto all'ordine
RCL-8	Modifica quantità piatto dal Riepilogo
RCL-9	Rimozione piatto dal Riepilogo
RCL-10	Conferma ordine
RCL-11	Visualizza conto
RCL-12	Ricerca piatto per nome
RCL-13	Inserisci recensione
RCL-14	Visualizza recensioni
RC-1	Visualizza ordine
RC-2	Rimozione piatti
RC-3	Modifica quantità piatti
RC-4	Seleziona tavolo
RC-5	Aggiunta tipologia di coperto
RC-6	Modifica quantità di coperti
RC-7	Conferma ordine

DIAGRAMMA DEI CASI D'USO

DIAGRAMMA DELLE COMPONENTI APPLICAZIONE



ARCHITETTURA CLIENT - SERVER

Il sistema è formato dalle seguenti componenti:

● *Client*: applicazione Android, supporta versioni del sistema ≥ 8.0 (Android Oreo)

● *Server*: Elabora le richieste inoltrate dal client e ne restituisce i dati.

Entrambe le componenti implementano il pattern *Model-View-Controller (MVC)* e la comunicazione avviene tramite richieste HTTPS di tipo POST e GET.

SCELTE IMPLEMENTATIVE

STRUTTURA DEL PROGETTO

Per quanto riguarda la struttura del progetto si è pensato di creare un package per ogni tipo principale di classe, in modo tale da rendere più semplice ed intuitiva la navigazione.
Di seguito possiamo notare la struttura fisica della cartella Src:

- **Java**
  - Controllers
    - Ui
      - Menu
        - Review
      - Receipt
      - Summary
  - Model
  - Nfc
  - Persistance

- **Res**
  - Anim
  - Drawable
  - Font
  - Layout
  - Menu
  - Navigation
  - Values

MODALITÀ DI NAVIGAZIONE

Abbiamo differenziato il sistema di navigazione delle due app.

Escludendo la prima activity in cui eseguiamo la chiamata al micro-servizio per recuperare i dati dal database e quella relativa all'NFC, l'app del Cliente è composta da un'activity sulla quale scambiamo dinamicamente i vari fragment che compongono il layout.

In funzione di questo, abbiamo preferito utilizzare un NavHostFragment in combinazione con un Nav Graph per gestire in maniera flessibile sia lo scambio dei fragment che il backstack dell'app. Per il passaggio da un'activity all'altra invece abbiamo utilizzato un Intent ed eventualmente un Bundle (per passare i dati tra activity o tra fragment).

Per quanto riguarda l'app del Cameriere, avendo solo due activity e due fragment, abbiamo optato per una gestione manuale dei fragment attraverso un FragmentManager, avviando la FragmentTransaction.

COMPONENTI LAYOUT

In quasi ogni sezione dell'applicazione è presente almeno una lista e le componenti utilizzate sono:

- ConstraintLayout: Uno dei layout maggiormente utilizzati è stato il ConstraintLayout, che ci fornisce grande flessibilità nella progettazione del layout che diventa praticamente responsive. Ogni elemento può essere ancorato al suo contenitore o ad un altro elemento dall'alto, dal basso, da destra o da sinistra. È possibile inoltre utilizzare il padding ed il Vertical/Horizontal Bias per regolare al meglio la distanza tra i vari elementi.
- CoordinatorLayout: Nel caso in cui abbiamo visualizzato una snackbar o abbiamo avuto la necessità di gestire il movimento di un Floating Action Button, abbiamo usato questo layout che permette a questi elementi di spostarsi l'uno in funzione dell'altro.
- LinearLayout: Abbiamo utilizzato questo layout per definire i layout delle righe per le varie RecyclerView, perché consente di

posizionare gli elementi in successione orizzontalmente o verticalmente di default.

- RecyclerView: Componente che può essere vista come una naturale evoluzione della classica ListView, ma più flessibile e ottimizzata per la visualizzazione di layout di riga complessi. Inoltre, consente una buona ottimizzazione delle risorse a runtime grazie alla recycler bin, che mantiene in una coda gli elementi non visualizzati per evitare un sovraccarico spiacevole dell'applicazione. È stata implementata anche l'interfaccia Filterable per il tasto della ricerca, per rendere più agevole trovare le portate nel menù.
- Nested ScrollView: Questo componente, come suggerisce il nome, viene utilizzato quando è necessaria una visualizzazione a scorrimento all'interno di un'altra. Normalmente ciò sarebbe difficile da realizzare poiché il sistema non sarebbe in grado di decidere quale visualizzazione scorrere.
- CardView: Per visualizzare le categorie, i dettagli di un piatto ed ogni riga nelle RecyclerView, abbiamo utilizzato questo componente per la sua flessibilità, che consente di creare delle schede graficamente gradevoli.

CONNETTIVITÀ

Le funzionalità offerte dalla libreria Volley hanno soddisfatto quasi tutte le situazioni. Questa ci ha permesso di far dialogare le due app grazie a delle chiamate asincrone che dialogano con i micro-servizi server-side. Questi ci permettono di sincronizzare il database locale con quello remoto per recuperare il materiale da visualizzare e, inserire gli ordini che vengono confermati dal Cameriere.

Per far dialogare le app del Cliente e del Cameriere invece ci siamo serviti della tecnologia NFC. Grazie a quest'ultima, ponendo i due dispositivi a meno di 10 cm di distanza tra loro, è possibile scambiare dei dati. Nel nostro caso specifico, ciò che transita tramite l'NFC è la stringa JSON che contiene tutti i dati dell'ordinazione.

PERMESSI

I permessi definiti nel manifest delle 2 app sono i seguenti:

- Android.permission. INTERNET
- Android.permission.NFC

Il permesso riguardante lo stato dell'NFC viene notificato all'utente nel momento in cui avvia l'app.

INTERFACCIA

Tutte le componenti utilizzate nell'applicazione sono state sviluppate seguendo le linee guida del Material Design. Per quanto riguarda la palette dei colori, si è deciso di crearne una personalizzata che possa identificare al meglio il dominio di appartenenza oltre che garantire una buona visibilità.

INTENT FILTER

L'Activity WaiterActivity dell'app del Cameriere è l'unica che possiede un intent filter. Questo consente di avviare un'activity quando viene scoperto un tag con payload NDEF.

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />

    <category android:name="android.intent.category.DEFAULT" />

    <data android:mimeType="text/plain" />
</intent-filter>
```

LIBRERIE DI TERZE PARTI UTILIZZATE

- *Volley*: libreria utilizzata per la gestione delle richieste HTTP, utilizzata per popolare un'interfaccia utente e il recupero di una pagina dei risultati di ricerca come dati strutturati. Si integra facilmente con qualsiasi protocollo e fornisce immediatamente il supporto per stringhe non elaborate, immagini e JSON.

- *Gson*: è una libreria che permette di effettuare un Parsing completo di oggetti Java in Json.

GESTIONE MULTILINGUA E PLURALS

Abbiamo gestito la traduzione dell'app attraverso le resources di Android. Le lingue gestite sono l'italiano e l'inglese, quest'ultima viene impostata di default se il sistema adotta una lingua differente. Inoltre, abbiamo gestito i plurali nel caso della visualizzazione di uno o molti ingredienti e allergeni all'interno del fragment DishDetailFragment.

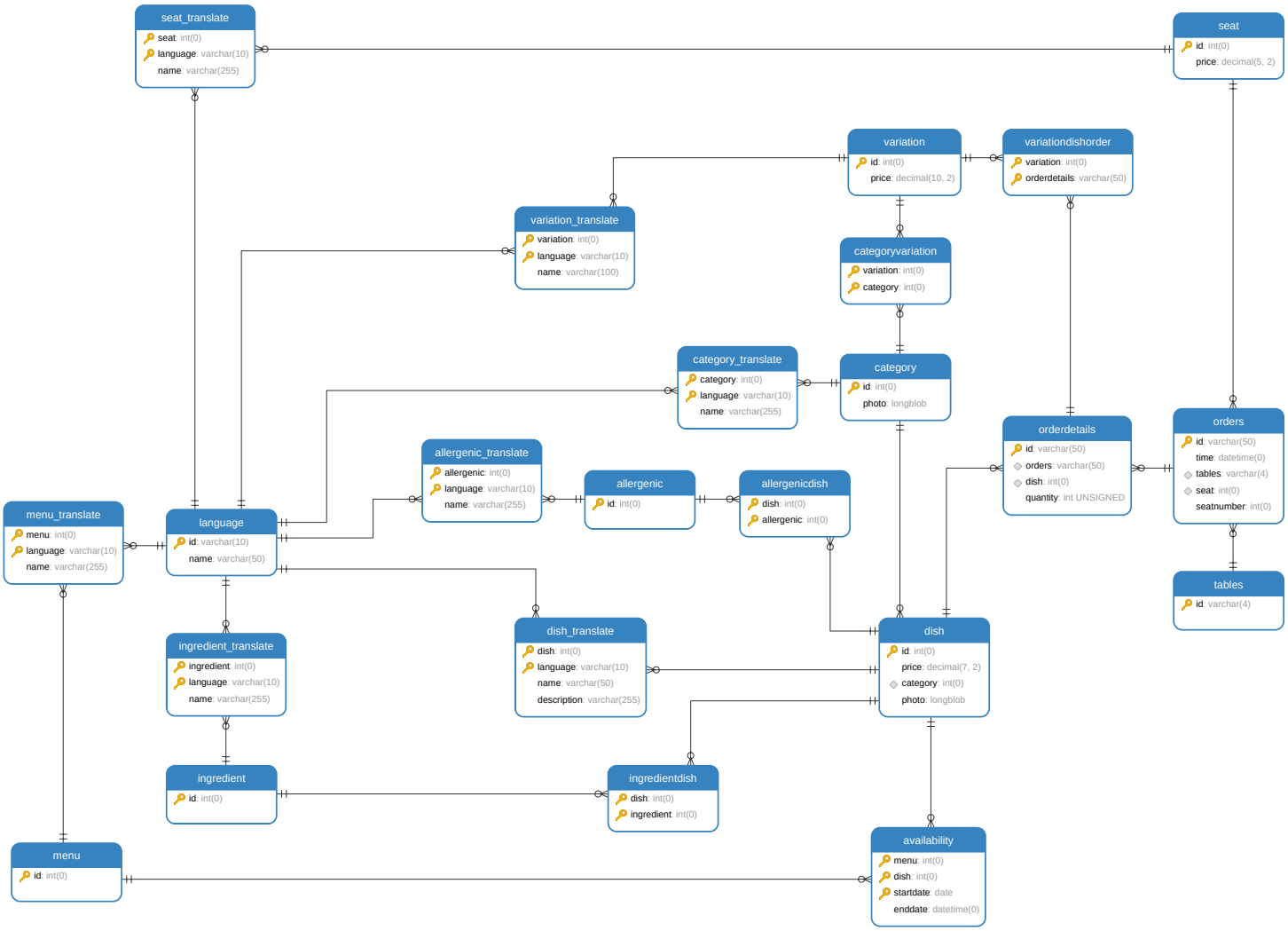
Per gestire le lingue del contenuto dinamico recuperato dal database, questo è stato progettato opportunamente. Inoltre, l'url per la richiesta http eseguita grazie alla libreria Volley è opportunamente modificato a runtime in base al "Locale" del dispositivo per permettere di scaricare solo i dati della lingua in uso.

PROGETTAZIONE DATABASE

Il database utilizzato è di tipo centralizzato, utilizzato per gestire la comunicazione tra la Cassa, il Cliente e il Cameriere e peraltro contiene informazioni importanti per il corretto funzionamento del sistema. Ristodroid esegue delle richieste a dei micro-servizi situati sul server per comunicare con il database in lettura e scrittura.

Viene inoltre utilizzato un sistema di persistenza locale per garantire la corretta visualizzazione del menu nell'eventualità di assenza di collegamento alla rete Internet (previa almeno una sincronizzazione del database).

I dati al suo interno sono raggruppati nelle seguenti tabelle:



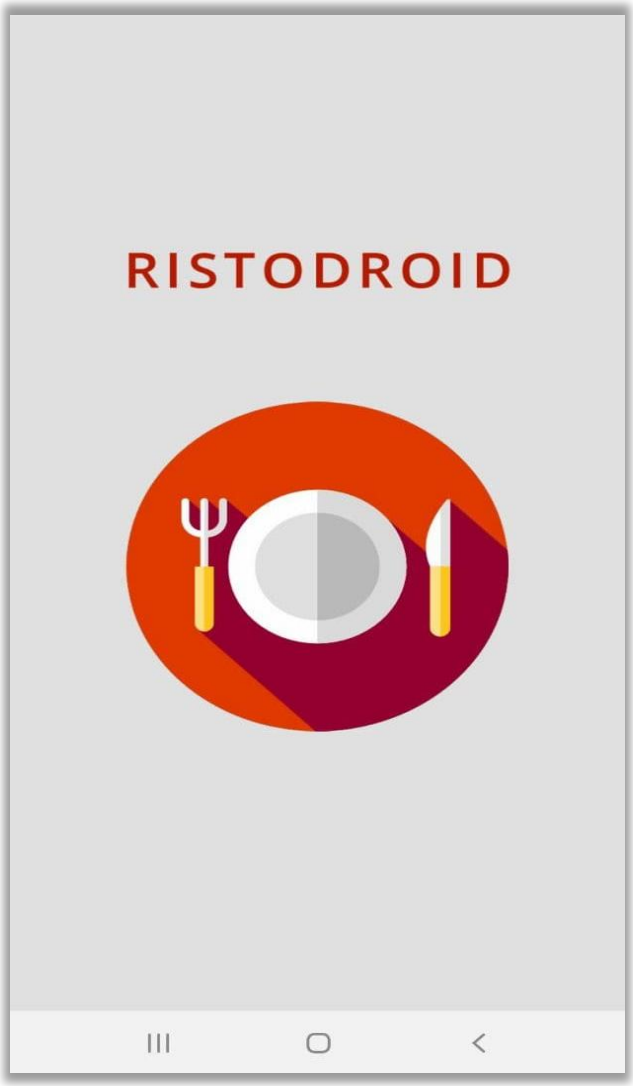
DATA ENTRY

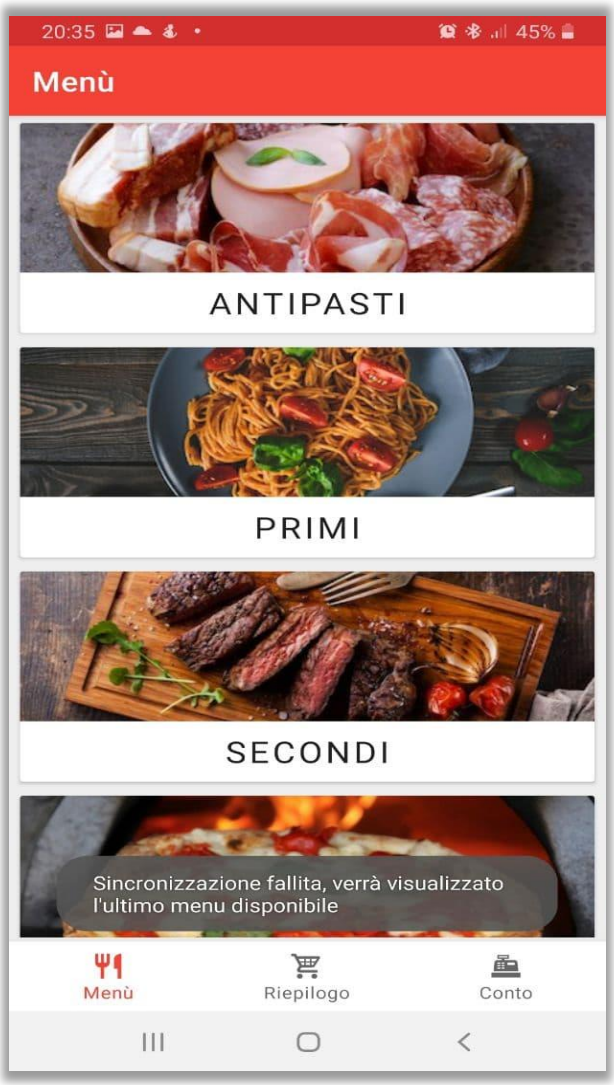
Per il data entry, non essendo stato possibile sviluppare una terza app, abbiamo sviluppato un sito utilizzando HTML/CSS/JS per il frontend e PHP per il backend che ci permette di inserire i dati nel database. Il tutto viene svolto grazie a delle richieste POST.

IMPLEMENTAZIONE DELLE FUNZIONALITÀ

INTRODUZIONE

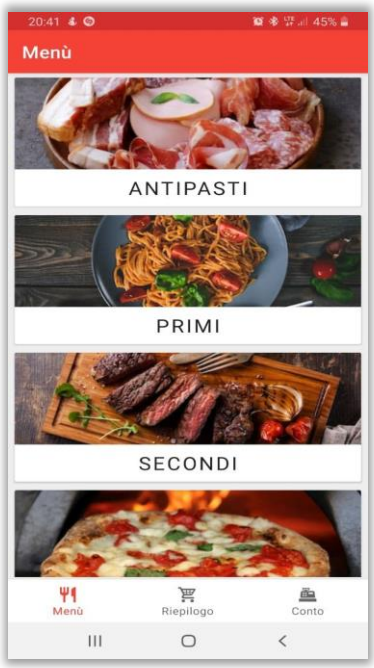
All'avvio dell'app viene mostrato uno splash screen necessario per consentire all'applicazione di sincronizzarsi con il database remoto. Se la sincronizzazione fallisce, viene mostrato all'utente un messaggio di avviso.

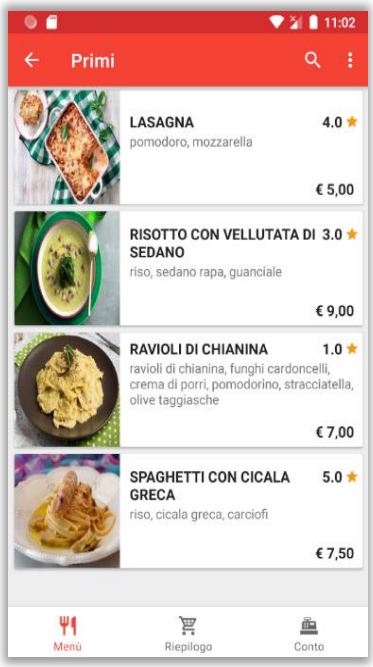


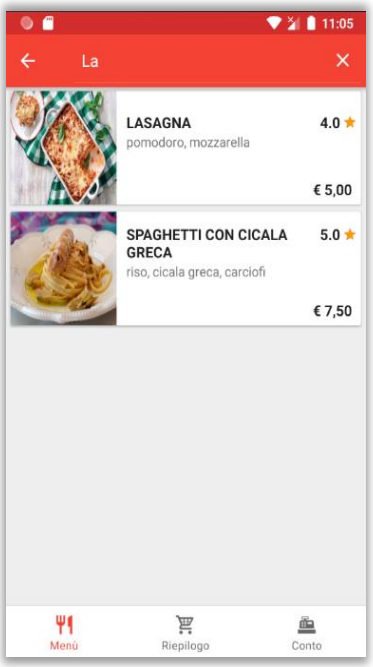


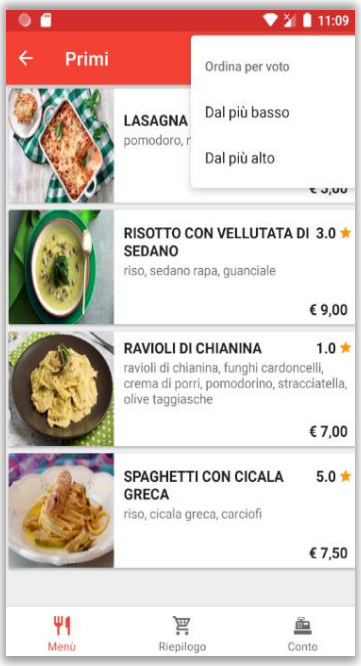
MENU

Il menu del ristorante è mostrato suddividendo i piatti per categoria. Inoltre, è possibile cercare un piatto inserendo il nome del piatto desiderato.



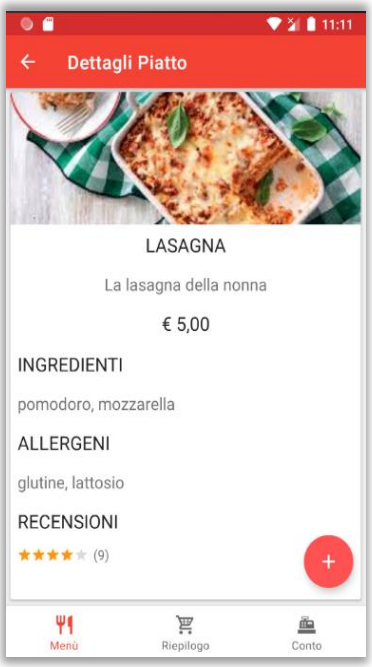


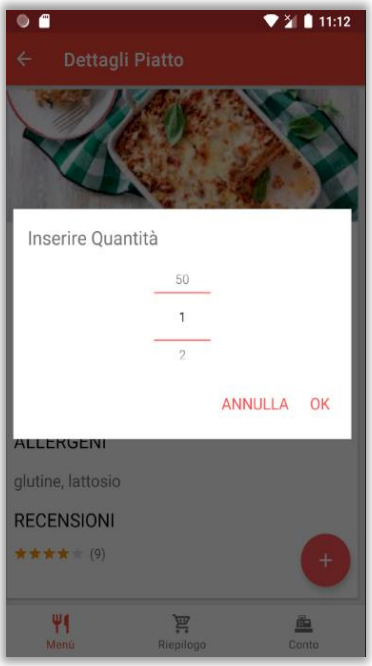


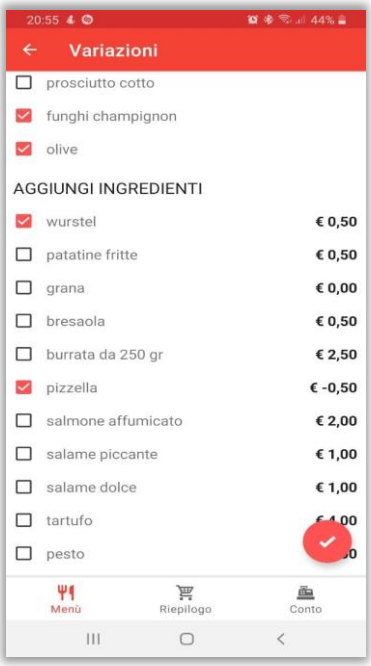


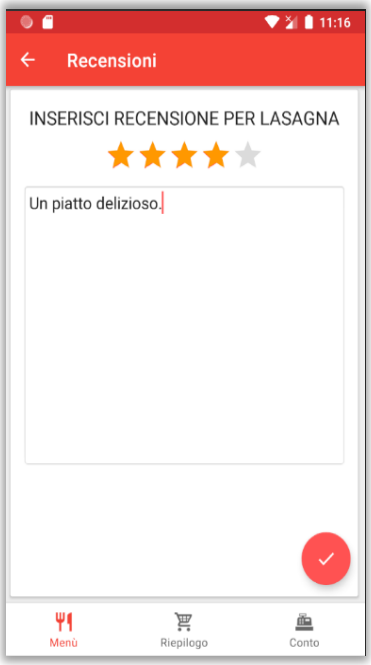
SELEZIONE PIATTO

Per ogni piatto è possibile visualizzare una sezione dettagliata con le informazioni relative ad allergeni ed ingredienti. Inoltre, è possibile selezionare quantità e personalizzazioni del piatto. Infine, sarà possibile visualizzare le recensioni lasciate per quel piatto, ordinarle secondo vari criteri ed inserire una nuova recensione.



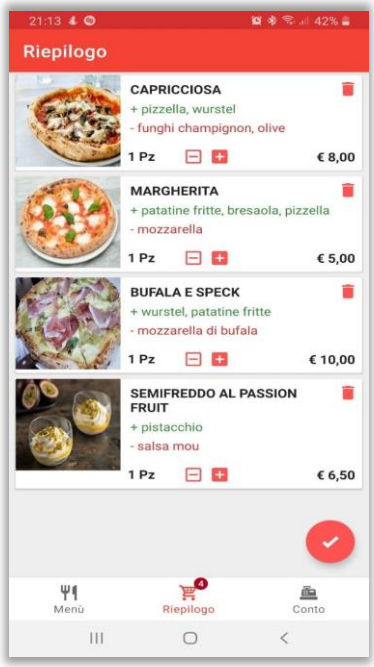


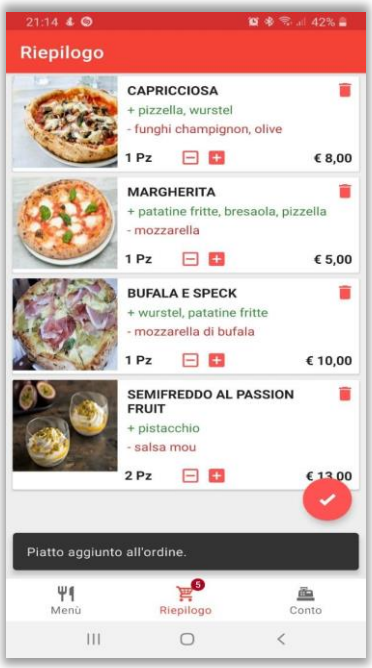


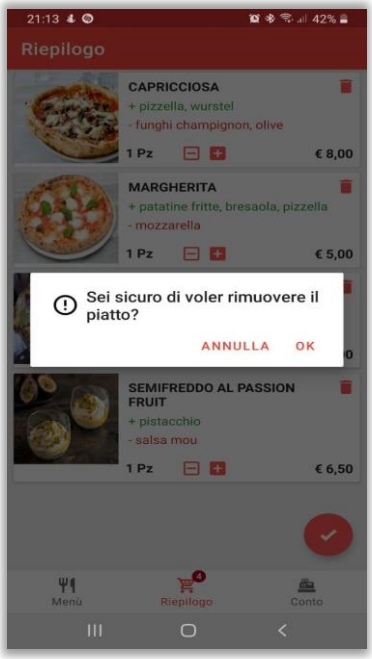


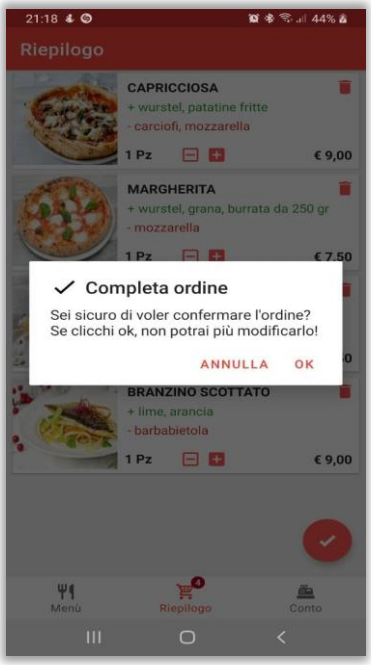
RIEPILOGO

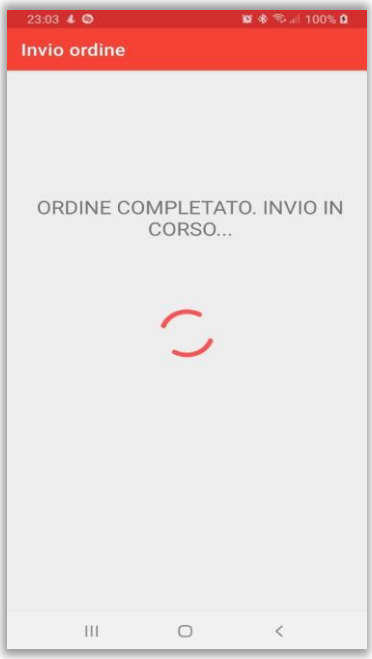
Dalla sezione riepilogo è possibile visualizzare i piatti ordinati, con la possibilità di modificare la quantità ordinata o di rimuovere il piatto dall'ordine. Inoltre, è possibile confermare, restando in attesa dell'acquisizione dell'ordine da parte del cameriere.





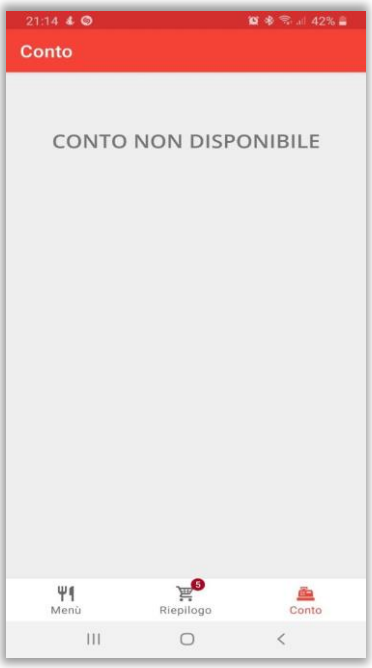


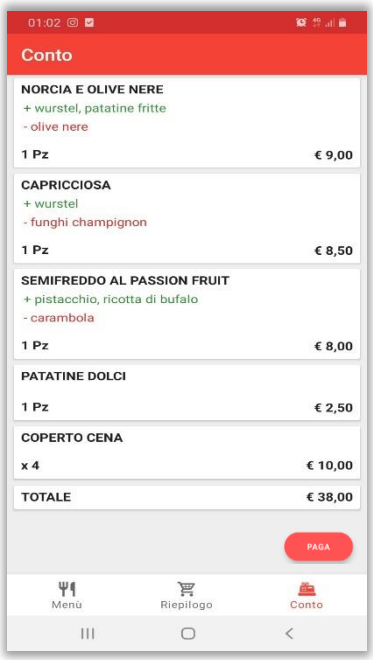




CONTO

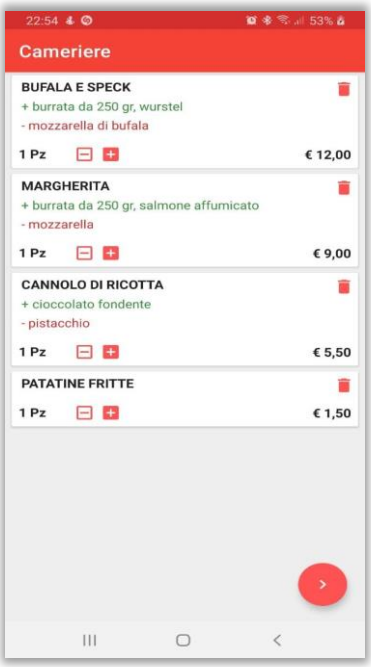
Dalla sezione conto è possibile visualizzare il conto di quanto è stato ordinato, nel momento in cui questo sarà disponibile.

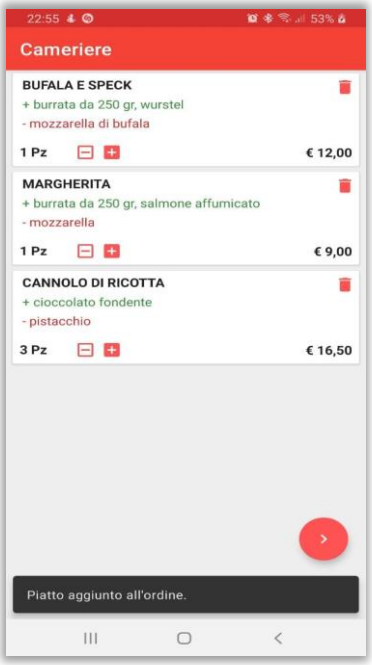




ACQUISIZIONE DATI CAMERIERE

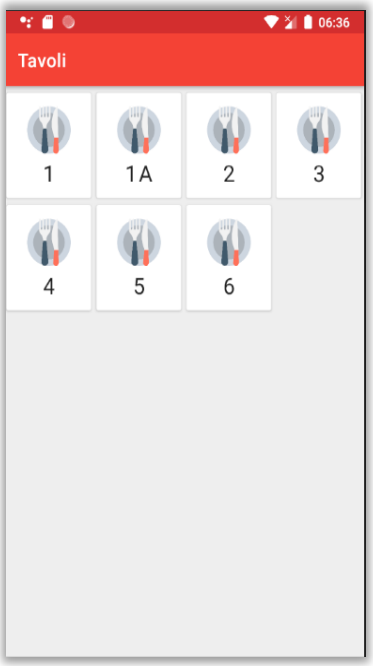
L'app del cameriere resta in attesa di acquisire un nuovo ordine. Quando questo avverrà, visualizzerà l'ordine sul proprio dispositivo avendo la possibilità di modificare la quantità di un piatto o di eliminare un piatto dall'ordine.

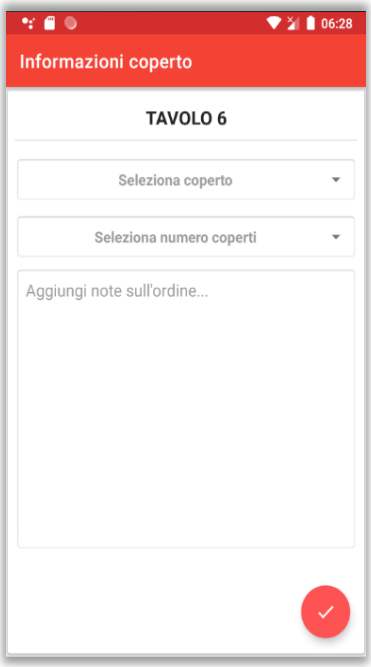




INSERIMENTO DATI ORDINE

Il cameriere inserirà il numero di coperti, la tipologia di coperti e l'identificativo del tavolo prima di confermare ed inviare l'ordine in cucina.





IMPLEMENTAZIONI FUTURE

Ci sono diverse possibili implementazioni future:

- Possibilità di proseguire l'ordinazione dopo aver confermato la prima.
- Oltre ad eliminare i piatti e cambiare la quantità di un piatto, implementare la possibilità di modificare la lista delle variazioni.
- Implementare la stampa dello scontrino.
- Sviluppare un'applicazione Android per la gestione del database e della cassa.