Proposta di progetto

Nome del progetto di laboratorio:

TotalShop

GRUPPO

Informazioni sul gruppo di laboratorio.

Nome del gruppo:	Gruppo TotalShop
Componenti:	* Vitale Pietro * Ninivaggi Saverio

DATE

Le date principali del documento.

Data di sottomissione della proposta di progetto	28/11/23
Data di accettazione della proposta di progetto	##DATA ACCETTAZIONE##

DESCRIZIONE BREVE

Descrizione della App in una frase.

Un'applicazione che permette di poter prenotare prodotti di tutti i tipi, da qualsiasi attività commerciale iscritta all'applicazione che sia nei dintorni dell'utente.

DEFINIZIONI

Di seguito la definizione dei termini, abbreviazioni e acronimi utilizzati.

Termine	Definizione

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

SOMMARIO

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

1 CONTESTO DEL PROGETTO

Lunghezza suggerita: 2 pagine

1.1 Situazione attuale

Descrizione dell'AS-IS, ovvero di cosa esiste già sul Play Store e in cosa le soluzioni attuali presentano problemi / limiti / mancanze.

Come sappiamo, attualmente esistono varie app che gestiscono la prenotazione e l'acquisto di prodotti.
Le più conosciute sono:

- JustEat
- Deliveroo
- Glovo

Le app tipo JustEat e Deliveroo permettono l'acquisto di cibo da una qualsiasi attività alimentare nella propria città.
Con la nostra applicazione miriamo ad ampliare questo tipo di servizio non solo per attività alimentari ma anche per qualsiasi altro genere di attività che venda prodotti. Così da poter dare visibilità a queste attività e migliorare le loro vendite.
Invece a differenza di Glovo che permette il ritiro solo da attività partner (processo molto lungo che richiede la firma di un contratto di collaborazione con Glovo) noi permettiamo anche a piccole aziende l'accesso all'applicazione.

1.2 Benefici e creazione di valore

Quali saranno i vantaggi / opportunità / risoluzione di problemi che rendono il vostro progetto appetibile / interessante?

La possibilità di poter collegare qualunque tipologia di negozio all'app e poter prenotare da qualunque negozio nella propria zona i prodotti di cui si ha bisogno. Il vantaggio principale della nostra app è quello di permettere anche a venditori di nicchia, non troppo conosciuti o semplicemente attività che non possiedono un e-commerce per acquisti online, di poter offrire i propri prodotti a utenti nelle vicinanze della propria attività (500m-10km) e guadagnare visibilità. Inoltre la possibilità di accedere come venditore all'applicazione con facilità e senza lunghi processi burocratici

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

1.3 Obiettivi del progetto

Elenco di aspetti specifici e verificabili che realizzerete nella vostra App e che la renderanno "appetibile".

Aspetto 1	Gli user della app verranno aggiornati sui cambiamenti dei loro ordini attraverso un servizio di notifiche
Aspetto 2	L'utente finale potrà designare un raggio (in metri) in cui verrà effettuata la ricerca per nome,categoria o prodotti dei negozi presenti nel raggio
Aspetto3	Il venditore potrà registrare la posizione del proprio store attraverso il click di un pulsante.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

2 PROFILO DEL PROGETTO

Lunghezza suggerita: 1 pagina

2.1 Ambito del progetto

Una descrizione di cosa la vostra App si propone di realizzare, cosa sarà sviluppato e cosa resterà da sviluppare alla fine del lavoro e di come il vostro progetto si legherà ad eventuali servizi e/o server esterni al telefono o all'emulatore.

L'obiettivo della nostra app è quello di poter permettere ai propri utenti di poter prenotare qualsiasi tipo di prodotto da qualsiasi attività commerciale che sia nei dintorni dell'utente (raggio a scelta del cliente). Qualsiasi attività commerciale che vorrà iscriversi all'app lo potrà fare come tale, dopodiché potrà inserire la categoria della propria attività e la lista di prodotti che offre. L'utente finale avrà una barra di ricerca dove potrà cercare sia per categoria che per nome dello store; inoltre potrà anche trovare lo store che vende il prodotto inserito nella ricerca; a questo punto verranno mostrate solo le attività nel raggio designato dall'utente che risponderanno ai requisiti della ricerca, da cui l'utente potrà creare il proprio carrello e prenotare i prodotti che in un secondo momento dovrà andare a ritirare. L'app inoltre permetterà al venditore di confermare o annullare l'ordine ricevuto, potendo scrivere un commento per la conferma/annullamento dell'ordine. Sulle informazioni dell'ordine inoltre sarà disponibile il numero dell'utente con il quale il venditore lo potrà contattare nel caso ci siano problemi con l'ordine (tipo problemi con la disponibilità di un prodotto). In caso di conferma/annullamento dell'ordine, verrà inviata una notifica all'utente, il quale potrà raggiungere la sua sezione ordini per controllare il messaggio del venditore.
Ai fini della privacy andrebbe collegata l'applicazione ad un servizio voip per permettere al venditore di chiamare l'utente senza conoscere il suo numero.
L'app userà il sensore di localizzazione per determinare la posizione e la distanza tra i negozi e l'utente.

2.2 Profilo della soluzione da realizzare

Una descrizione di come sarà strutturata la App che verrà realizzata.

L'applicazione verrà strutturata seguendo il paradigma MVVM.
La navigazione tra una schermata e un'altra verrà gestita utilizzando Jetpack Navigation.
L'app in totale avrà 3 views che utilizzeranno un ConstraintLayout per un migliore posizionamento dei contenuti:

- View di login
- View dell'utente
- View del venditore

All'apertura dell'app si avrà la view di login contenente due pulsanti con i quali si sceglierà se si vuole accedere come utente o venditore.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

Grazie all'utilizzo di jetpack navigation, viene caricato il fragment successivo, che oltre a mostrare i vari campi per effettuare il login, passerà anche dei dati che permetteranno all'app di sapere nei passaggi successivi se l'user è un venditore o un utente. Nel caso di un user non registrato, c'è la possibilità di passare al fragment che gestisce la registrazione, che nel caso in cui sia un venditore inserirà anche i dati del suo store. Dopo aver inserito tutti i dati, l'user verrà reindirizzato nella schermata successiva (utente o venditore a seconda della tipologia)
Per la parte dell'utente, una volta loggato si verrà reindirizzati nell'Activity dell'utente, che sarà composta da una Toolbar e una bottom navigation bar collegata ai vari fragments su cui l'utente potrà navigare.
Fragments:

- HomeFragment: contente contenente la barra di ricerca dei prodotti con filtro per cambiare il range (metri) e la lista dei negozi più vicini mostrata tramite una personalizzazione del RecyclerView. Dopo aver usato la barra di ricerca verrà aggiornata la lista di negozi sottostanti con quelli che coincidono con la ricerca, per inserire i negozi, si effettueranno chiamate a Room per estrarre i dati dal db[1], e verranno utilizzati i LiveData. Una volta selezionato il negozio dalla lista, verrà caricato un secondo fragment chimato UtenteProdListOrders.
- UtenteProdListOrders: contente la lista dei prodotti venduti dal negozio; l'utente potrà creare il suo carrello della spesa selezionando uno o più prodotti dalla lista. Una volta confermato il carrello l'ordine dovrà essere confermato dal venditore, per poi essere ritirato dallo stesso utente in un orario specificato dal venditore nel commento dell'ordine.
- OrdiniFragment: contenente la lista degli ordini effettuati dall'utente; cliccando su un ordine è possibile navigare su un altro fragment chiamato DettagliFragment.
- DettagliFragment: contenente i dettagli dell'ordine cliccato e un pulsante per cancellare l'ordine da parte del'utente/sul venditore due pulsanti (annulla e conferma). Questo fragment viene caricato insieme a quello degli ordini se lo schermo è in landscape.
- Settingsfragment: contenente le impostazioni dell'account dell'utente, è composto da due fragment, il primo ha la lista delle impostazioni, ed il secondo mostrerà le info dell'impostazione selezionata. E' possibile fare il logout da questo fragment.

Per la parte del venditore, una volta loggato si verrà indirizzati sull'Activity del venditore contenente una Toolbar e una bottom navigation bar utilizzata per caricare i vari fragments.
Fragments:

- HomeFragment: contenente una searchBar con la quale filtrare i prodotti tramite nome e descrizione. Un pulsante per l'aggiunta di un nuovo prodotto, che farà apparire un

[1] Ai fini dell'esame, per utilizzare Room, non verrà implementato l'accesso ad un database remoto, e di conseguenza il lato venditore ed il lato utente non potranno comunicare direttamente. Per simulare il corretto funzionamento dell'app in tutti i casi descritti sopra, verranno utilizzate delle funzioni che simuleranno cambiamenti all'interno dei db locali dell'applicazione (AppInspector)

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

AlertDialog per l'inserimento delle informazioni del nuovo prodotto; e infine la lista dei prodotti del proprio store mostrata utilizzando una personalizzazione del RecyclerView con Adapter. Su ogni card dei prodotti è presente un pulsante per la rimozione del prodotto dalla lista.

- OrdiniFragment/DettagliFragment: stessa logica dell'utente, unica differenza sui Dettagli il venditore potrà sia confermare un nuovo ordine che annullarlo, sempre indicando una motivazione.
- SettingsFragment: contenente le impostazioni dei dati personali del venditore e dei dati dell'attività commerciale inserita (nome,indirizzo,categoria), sarà composto da due fragment, il primo ha la lista delle impostazioni, ed il secondo mostrerà le info dell'impostazione selezionata. E' possibile fare il logout da questo fragment.

Per la parte tecnica, per quanto riguarda la visione dei dati in tempo reale (lista dei negozi nelle vicinanze, interazione con il sensore gps, interazione con il db) seguendo il paradigma MVVM, non ci sarà un collegamento diretto tra View e dati (sensore o db), ma tutte le interazioni avverranno attraverso più View model, che si comporteranno da tramite per passare i dati alla view. Per effettuare ciò, utilizzeremo i LiveData (e gli observer) che ci permetteranno di avere dati in tempo reale.
Per la persistenza dei dati, utilizzeremo Room, che ci permette di salvare su un db locale tutti i vari dati necessari per l'utilizzo dell'app (informazioni dell'user, ordini passati, ecc). Come citato precedentemente, useremo i LiveData per aggiornare dinamicamente l'interfaccia dell'utente con i dati presi dal Room db.
Per mantenere l'utente loggato all'apertura dell'app dall'ultimo utilizzo verrano utilizzate le SharedPreferences (key,value), che verranno eliminate al logout.
Inoltre utilizzeremo i service per il funzionamento asincrono dell'app per gestire le notifiche che informeranno l'utente e il venditore sulla creazione / variazione dello stato dell'ordine.
Infine per gestire la posizione dell'utente rispetto ai negozi nelle vicinanze, utilizzeremo il sensore di locazione, che grazie alla latitudine e longitudine dell'utente e dei negozi, potremo mostrare i negozi che si trovano entro un raggio definito dall'utente, centrato nella sua posizione.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

3 VINCOLI E ASSUNTI (REQUIREMENTS)

Lunghezza suggerita: 2 pagine

3.1 Vincoli tecnologici

Functional Requirements

Linguaggio: Kotlin
Navigazione tra view: Jetpack Navigation, Fragment Manager
Design pattern: MVVM
Cambiamento del dato: LiveData
Database: Room
Chiamate db: Coroutines
Posizione utente/venditore: sensore di locazione
Il dispositivo utilizza una versione android compresa tra API 21 e API 33

Non-functional requirements

Notifiche: Service
Schermo landscape adattabile su ordini: FragmentManager

3.2 Eventuali assunti

- Quando il venditore deve registrare la posizione del proprio store si deve trovare fisicamente nello store e ovviamente deve tenere accesso il gps.
- Quando l'utente deve effettuare la ricerca dello store, deve avere il gps attivo
- Per ricevere aggiornamenti delle notifiche, l'user deve aver autorizzato l'applicazione ad inviare notifiche, e deve essere loggato nell'applicazione (anche se l'applicazione è chiusa)

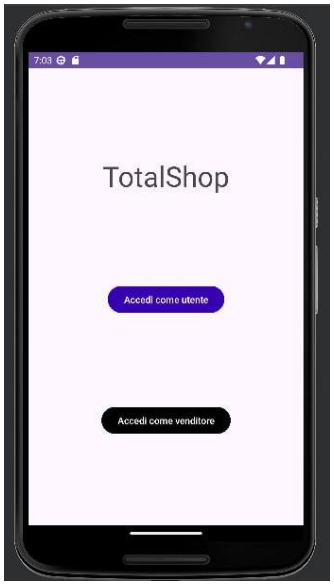
Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

4 DESIGN

Lunghezza suggerita: 2-3 pagine

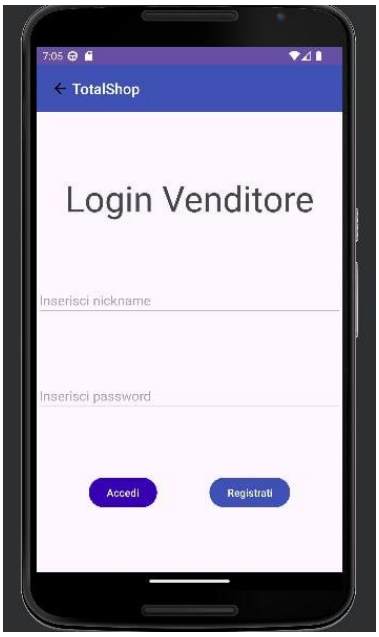
Descrivere le diverse parti del progetto (es. usando un diagramma) e spiegare come in linea di principio l'architettura soddisfa i requirements. Spiegate bene come per esempio il progetto segue il modello MVVM, come le diverse capacità di Jetpack sono usate (Live Data, Room, co-routines, etc.)

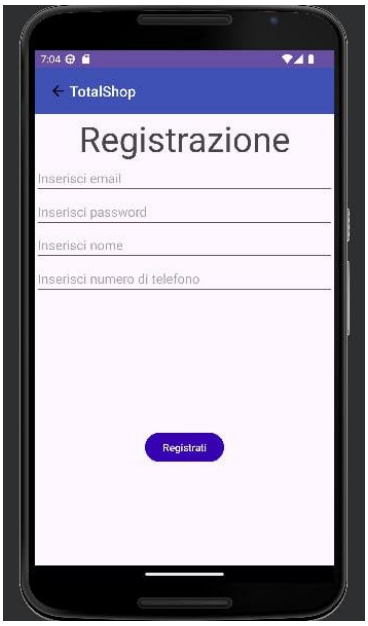
La prima schermata che appare all'apertura dell'applicazione è la schermata Home della LoginActivity. Nella quale si potrà scegliere se loggarsi come utente o come venditore.
Entrambi i pulsanti ci porteranno alla pagina di login utilizzando arguments differenti sfruttando un navController.



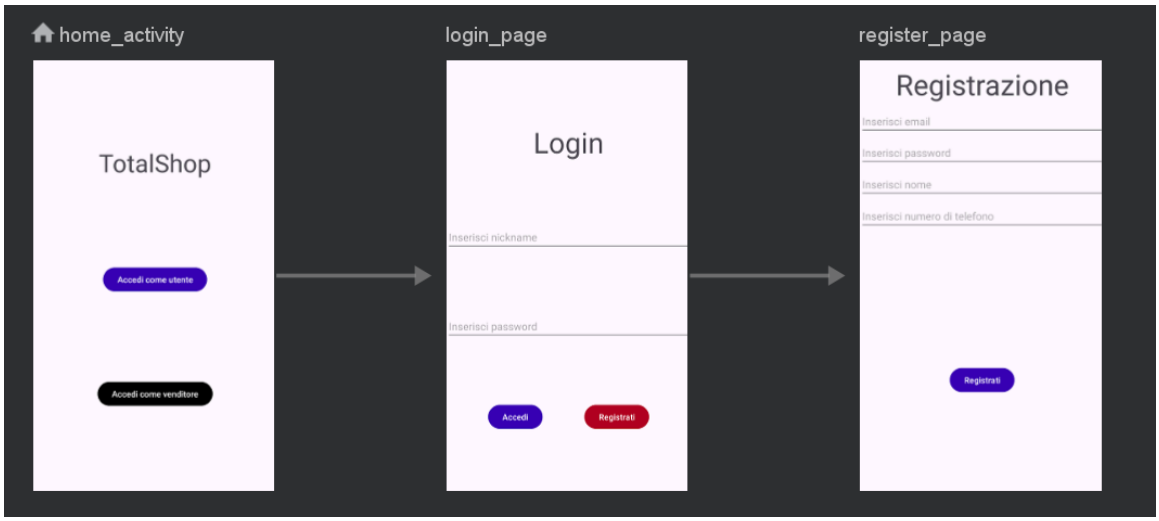
Una volta arrivati nel Login (utente o venditore) si potranno inserire email e password per poi loggarsi passando soliti controlli di esistenza nel db e che la password sia corretta.
Nella schermata è anche presente un pulsante per navigare nella pagina per registrarsi sempre passando i medesimi arguments per definire il tipo dell'user (utente o venditore).

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603





Per ogni Activity è stato creato un Navigation Graph per gestire lo spostamento dei vari fragments.
Il navigation graph per la LoginActivity è il seguente:



Una volta fatto l'accesso possiamo trovarci nelle seguenti Activity:
-UtenteActivity
-VenditoreActivity

Per quanto riguarda l'utente activity, appena loggati viene caricato il fragment HomeFragmentUtente. Il quale contiene una la lista dei negozi più vicini all'utente, filtrabili comodamente grazie ad una searchBar accompagnata da un filtro per il radius di ricerca (metri).

Alle card degli store più vicini è stato aggiunto un OnClickListener il quale si occuperà di caricare il fragment contenente la lista dei prodotti dello store. Anch'essi mostrati a card sulle quali appare

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

nome, descrizione e prezzo del prodotto con un drawble che al click farà aprire un modale per l'inserimento della quantità del prodotto che si vuole aggiungere al carrello.

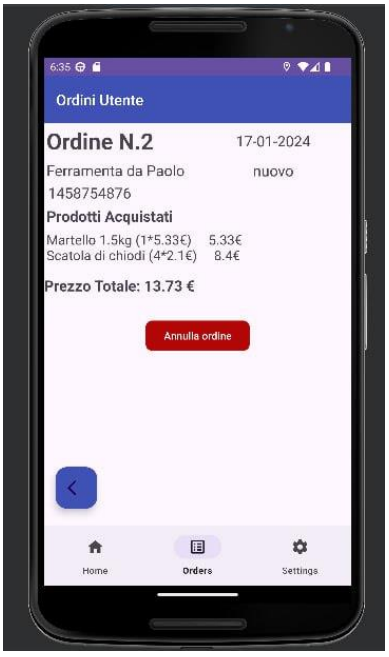
Una volta aggiunto almeno un prodotto al carrello apparirà un FloatingButton che al click aprirà il carrello dei prodotti in un AlertDialog nel quale si potrà confermare l'ordine o andare indietro.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

Nelle card della lista dei prodotti nel carrello è presente un drawable che al click rimuoverà il prodotto dal carrello.

Se si vuole aggiungere un prodotto nonostante sia già nel carrello lo si può fare modificandone semplicemente la quantità (che risulterà aggiornata) nel AlertDialog precedente.

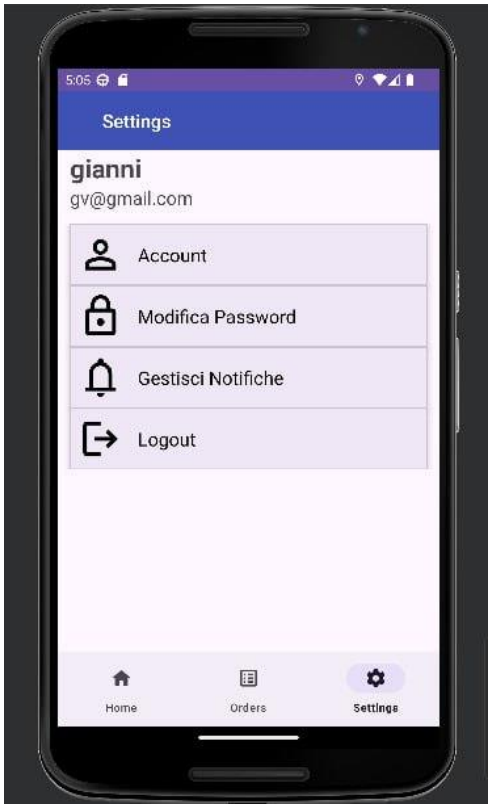
Una volta creato l'ordine che risulterà nello stato "nuovo", potra essere visionato nella schermata degli ordini.
La schermata degli ordini è raggiungibile utilizzando la bottom nav bar e si presenta con una lista di ordini fatti dall'utente / nel caso del venditore gli ordini dello store.
Le cards degli ordini possono essere cliccate per vederne i dettagli o per cancellare l'ordine(utente).

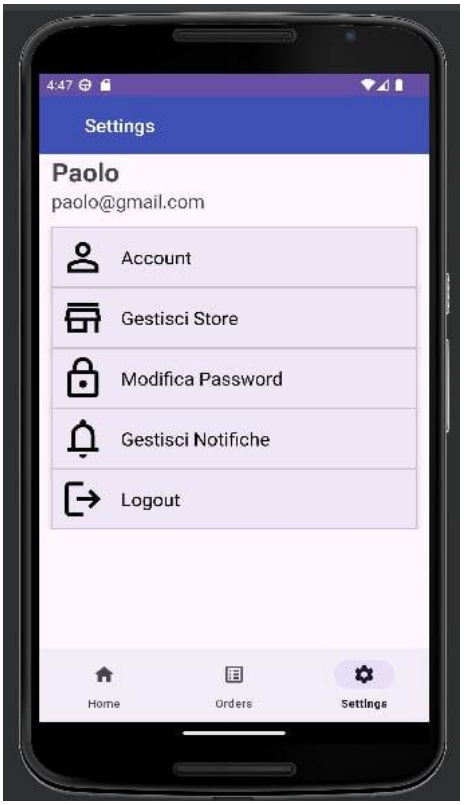


Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

Come detto in precedenza questi due fragment, nel caso in cui si metta la modalità landscape, vengono caricati insieme.

Infine l'utente potrà navigare sulla schermata delle impostazioni per fare il logout o per modificare i suoi dati / nel caso del venditore anche i dati del suo store.





Di seguito le schermate delle modifiche dei dati, password, dati store(venditore), e attivazione/disattivazione notifiche.

Per quanto riguarda VenditoreActivity, le schermate sono praticamente le stesse tranne per la home del venditore che presenta tutti i prodotti presenti nel suo store filtrabili da una searchBar e rimuovibili tramite icona sulla card del prodotto. E' presente anche un pulsante per l'aggiunta

delle informazioni del nuovo prodotto; apparirà un AlertDialog con delle EditText e due pulsanti per aggiungere il prodotto o tornare indietro.

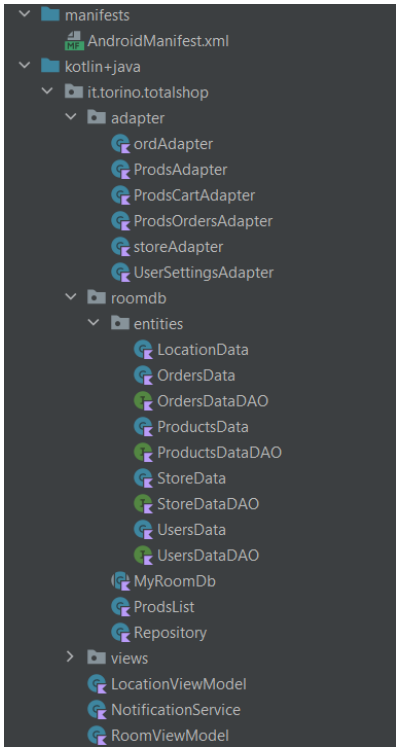
Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

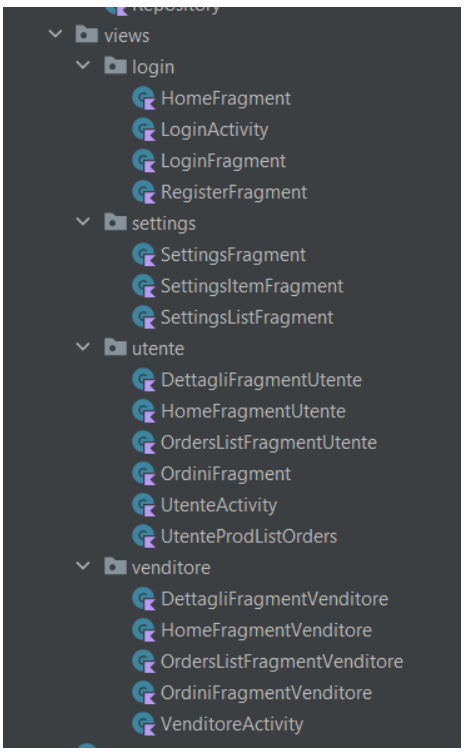
5 IMPLEMENTAZIONE

Lunghezza suggerita: 2-3 pagine

Dettagliare l'implementazione (come avete realizzato il design – es. organizzazione delle classi – il diagramma di navigazione di Android (vedere la lezione su navigation) è un ottimo punto di partenza.

Il nostro progetto segue il paradigma MVVM, di conseguenza abbiamo suddiviso i file seguendo i raggruppamenti Model-View-ViewModel.





Nel nostro caso il model è la cartella roomDb nella quale ci sono tutti i file per la creazione del db e le varie entities (con associati i Dao) che lo compongono insieme al repository.
La cartella views contiene tutti i fragment collegati ai layouts che interagiscono con gli user precedentemente mostrati.
E infine i viewModels utilizzati:

- RoomViewModel , per fare chiamate al DB e aggiornare i LiveData che verrano osservati dalle view.
- LocationViewModel, per chiedere i permessi di accedere alla posizione,per interfacciarsi con il sensore di locazione e aggiornare i LiveData della latitudine e longitudine, sempre osservati dalle view. Per calcolare la distanza tra un utente e uno store, viene calcolata la distanza tra 2 punti (identificati nel nostro caso con latitudine e longitudine) sulla terra

attraverso una formula matematica. Se la distanza calcolata dello store si trova entro quella selezionata dall'utente allora verrà mostrato tra gli store disponibili.

Come citato nel design dell'app, all'interno del nostro progetto ci sono due logiche per navigare tra i vari fragment:

- La prima logica utilizza la Jetpack Navigation, che viene implementata per gestire la navigazione tramite action nella LoginActivity; e tramite l'utilizzo della bottom navigation bar tra i fragment principali nelle altre Activity (UtenteActivity, VenditoreActivity);
- La seconda logica utilizza il Fragment Manager, e viene usata in tutti i casi in cui abbiamo un fragment della prima logica, che all'interno ha un FragmentViewContainer dove verranno alternati altri fragment, con l'utilizzo delle transaction.

Per quanto riguarda la visualizzazione di tutte le liste (stores,prodotti,ordini) vengono utilizzati dei RecyclerView con adapter personalizzati che caricheranno i layout personalizzati delle cards e aggiungeranno i vari onClickListeners.

Per l'invio delle notifiche viene utilizzato il service chiamato NotificationService, il quale partirà in background all'apertura dell'app controllando prima se l'user ha autorizzato l'invio delle notifiche e che abbia le notifiche attive (valore booleano salvato nelle sharedPreferences). Una volta partito avvierà uno scheduler che ogni 30 secondi controlla sul db se ci sono ordini nuovi/cancellati da notificare al venditore o ordini confermati/annullati da notificare all'utente; anche in questo controllo vengono utilizzate le shared Preferences per continuare ad avere traccia dei nuovi ordini anche se il servizio di notifiche viene stoppato (in caso di logout o disattivazione delle notifiche).
Dato che la versione di API minima per la nostra applicazione è 21 e le notifiche dalla 26 in poi sono gestite tramite l'utilizzo di Channel, abbiamo dovuto fare un controllo per implementare entrambe le versioni di creazione di notifica.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

6 TEST

Lunghezza suggerita: 2 pagine

Come avete testato l'app. In genere non è sufficiente che solo l'autore la testi. Utenti esterni (possibilmente una decina) sono richiesti per un progetto eccellente.

L'applicazione è stata testata più volte durante la scrittura del codice per accertarci che i dati venissero passati correttamente tra i vari componenti dell'app; abbiamo utilizzato Log o simulato cambiamenti dello stato dell'ordine nel db tramite AppInspector.
I test che abbiamo fatto durante tutta la creazione dell'applicazione, ci hanno permesso di risolvere eventuali bug o problemi.
Principalmente l'app è stata testata sull'emulatore di Android Studio, ma è stata anche testata su dispositivi fisici, questo ci ha permesso di poter risolvere eventuali bug grafici e ha confermato la capacità del sistema di adattarsi a dispositivi differenti, mantenendo le sue funzionalità.

Dispositivi su cui l'app è stata testata:

Dispositivi Virtuali:
- Nexus 6 API 33
- Small Phone API 21

Dispositivi Fisici:
- Redmi Note 9 Pro API 31
- Xiaomi POCO X3 API 31

Inoltre sono state coinvolte altre persone esterne (amici e famigliari) al progetto per testare la semplicità e logica dell'interfaccia dell'applicazione e per poter simulate come un utente qualunque si sarebbe spostato all'interno dell'app.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

7 CONCLUSIONI

Lunghezza suggerita: 1-2 pagine

Discutere come i requirements sono stati rispettati nella soluzione finale (facendo riferimento sia ai requirements iniziali e quanto discusso nella sezione di design) e come i tests abbiano dimostrato che il progetto faccia effettivamente quanto promesso

In conclusione l'applicazione è in grado di offrire le funzionalità richieste rispettando il paradigma MVVM (Model-View-ViewModel) e quindi dividendo le responsabilità tra i vari componenti e migliorando la gestione dei dati.
L'utilizzo di Room ha semplificato la gestione della persistenza dei dati consentendo un accesso semplice alle informazioni degli utenti, stores, prodotti e ordini. Grazie all'utilizzo di LiveData e Coroutines siamo riusciti ad ottenere interfacce sempre aggiornate e reattive, evitando blocchi nell'interfaccia utente.
L'utilizzo di Shared Preferences ha facilitato la gestione della persistenza del login dell'utente, grazie alla quale ogni utente può accedere direttamente nella sua Home se ha già fatto il Login in precedenza.
L'utilizzo coordinato tra Jetpack Navigation e Fragment Manager ha facilitato la gestione degli spostamenti da una schermata all'altra.
A seguito dei test avvenuti su diversi dispositivi , possiamo confermare che l'applicazione può essere eseguita su diverse categorie di dispositivi, con l'unica condizione che il dispositivo abbia una versione API compresa tra le API 21 e API 33.
In un futuro si potrebbe testare ed estendere l'app affinchè possa essere eseguita su dispositivi con versioni android più recenti (API 34).
Come citato in precedenza dato che utilizziamo Room per salvare i dati in maniera persistente, e Room crea un db locale per ogni applicazione, se installata su più dispositivi non potranno comunicare con le notifiche, poiché il NotificationService controlla il db che è unico per ogni app.
In futuro per rendere l'applicazione effettivamente utilizzabile in maniera completa, andrebbe collegato un database esterno per il salvataggio dei dati.
In definitiva l'applicazione permette di prenotare prodotti di qualsiasi categoria da store vicini alla propria posizione in maniera facile e rapida sia per quanto riguarda l'utente per la prenotazione di prodotti, che per quanto riguarda il venditore per la creazione dello store e aggiunta di prodotti attraverso l'utilizzo delle tecnologie sopracitate.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603