Shopping Cart

Progetto d'esame per "Sviluppo di applicazioni mobili"

a.a. 2017/18 – Università di Pisa - Informatica

a cura di Davide Montagno Bozzone (535910 – Corso A)

1. Introduzione

1.1 Specifiche dell'applicazione

L'applicazione sviluppata permette all'utente di poter stilare la propria lista della spesa, scambiare quest'ultima tramite NFC utilizzando una comunicazione P2P chiamata Android Beam e ricevere notifiche dei negozi presenti nelle vicinanze che possiedono i prodotti indicati. L'applicazione è disponibile per tutti i dispositivi mobili Android che soddisfano determinati requisiti come la versione della piattaforma SDK con cui l' app lavora per la compilazione (CompileSdkVersion 27), il livello API a cui l'applicazione si rivolge" (targetSdkVersion 25) e il requisito minimo di API (minSdkVersion 23).

1.2 API

Vengono utilizzate le API Google, tra cui Maps SDK, Place Search e infine Location. Per scelta personale il codice è organizzato in modo tale che l'URL, fornito con le informazioni sulla posizione per poter ricevere tutti i negozi nelle vicinanze, possa essere settato creando file JSON personali simili a quelli di Google o tramite l'utilizzo dell'API Place Search.

2. Organizzazione del codice

2.1. Main Activity

L'activity Principale permette all'utente, tramite supporto di Action Bar, Floating Button, TextView e menù personalizzati, di interagire con tutte le funzionalità dell'applicazione. La "user experience" è molto semplice da utilizzare grazie al supporto di icone molto semplici riprese dal Material Design. Al momento della creazione (onCreate()) l'activity invoca il

metodo startService();, creando e successivamente eseguendo il Service in background (Sez. 2.6) indefinitamente.
Tramite i Floating Button l'utente può interagire con altre 2 activity (MapsActivity, Database_display_info); il costo delle animazioni per gli effetti di comparsa e scomparsa dei FB o della rotazione del FB centrale è minimo, portando alla scelta di eseguire gli effetti nel Thread UI.
Attraverso un Database (gestito tramite la classe Database_helper) si ricavano le informazioni sui prodotti da acquistare e sui negozi che ne hanno la disponibilità, per permettere al Service e al BroadcastReceiver di collaborare per informare l'utente (tramite una notifica) che è nei paraggi di qualche negozio.
Ogni qual volta viene avviata l'applicazione viene aggiornata la "table_test2" del Database tramite richieste http, in modo da aver una mappa chiave-valore (OBJECT,SHOP) sempre aggiornata.

2.2. Maps Activity

Questa Activity permette all'utente di interagire con la mappa (fornita tramite l'utilizzo dell'API Maps SDK di Google), di mostrare tutti i negozi nei paraggi (localizzati tramite Marker con l'aiuto di un AsyncTask) e di controllare in real-time la posizione. L'Activity sfrutta lo stesso principio ideato per la "Main Activity", difatti tramite richieste http e accessi al database vengono ricevuti i dati (in formato JSON) in modo da poter garantire le funzionalità descritte in precedenza.
L'Activity implementa le interfacce per la creazione della mappa (attraverso le **GoogleApiClient)**, per la connessione alle API, e l'eventuale fallimento, "clickListner" e "LocationListner" per poter controllare dove si trova l'utente e mostrare ciò che lo circonda.

2.3. Database display info

L'utente all'interno di questa Activity può inserire, selezionare, deselezionare o rimuovere i prodotti da acquistare. Una volta che l'utente preme "Invio" dalla tastiera, i dati inseriti vengono passati al database che li memorizza permanentemente fin quando non vengono rimossi dall'utente. Una ListView permette l'interazione con i vari elementi, e tramite un'Adapter collegato ad essa, viene mostrata un'interfaccia sempre aggiornata tramite accessi al database. La TextView mostra un testo personalizzato in base al "sesso" espresso nelle preferenze dell'utente.

2.4. Setting Activity

Tramite le **Shared Preference**, l'utente può personalizzare il nome, il sesso, le notifiche, il raggio di ricerca dei negozi nelle vicinanze. In versioni successive dell'app potrà anche preservare le preferenze scelte.
L'Activity costruisce un sistema di preferenze tramite 3 risorse di tipo xml. Quello principale (pref_header.xml) contiene i 2 header che riferiscono i fragment PreferenceFragment (pref_general.xml) e LocNotifyPreference (pref_data_sync.xml) i quali permettono all'utente di settare le sue preferenze. Viene implementata l'interfaccia onPreferenceChangeListner, e ad ogni modifica di una delle preferenze attraverso i metodi bindPreferenceSummaryToValue e bindPreferenceSummaryToValue2 si conserva la nuova preferenza (i metodi vengono invocati nell'onCreate()).

2.5. NFCActivity (sender) / NFCDisplayActivity (receiver)

Queste due activity permetto lo scambio della lista della spesa tra due dispositivi mobili che hanno installato l'applicazione. Una volta abilitato NFC e Android Beam, viene trasferita la lista appoggiando semplicemente i due dispositivi mobili l'uno con l'altro (parte posteriore). La modalità P2P conosciuta come Android Beam, permette di scambiare dati al dispositivo NFC con altri peer NFC.

2.5.1. setNdefPushMessageCallBack

E' una callback da chiamare nell'onCreate() dell'NFCActivity tramite un NFCAdapter. Appena i dispositivi vengono avvicinati la callback registra l'evento, viene creato un messaggio (NDEF) che verrà spedito non appena l'utente farà un "tap" sullo schermo.

2.5.2. ACTION_NDEF_DISCOVERED

È l'intento di iniziare un'attività nel momento in cui viene scoperto un payload di tipo NDEF. Una volta ricevuto l'Intent corrispondente, vengono inseriti i valori ricevuti all'interno del

database. Questa action è presente all'interno dell'AndroidManifest come Intent Filter dell'activity NFCDisplayActivity.

2.6. MyService

Questo Service aggiunge dei ProximityAlert, per poter notificare l'utente (tramite BroadcastReceiver (Sez. 2.7)) l'entrata all'interno del raggio d'azione di un particolare negozio che possiede i prodotti da acquistare. Implementa un AsyncTask che esegue il download delle informazioni (tramite file JSON personali caricati online (scelta di questo progetto), o tramite richieste API in formato URL (implementato ma non utilizzato)) per poter settare correttamente i ProximityAlert. Una volta che l'utente entra all'interno di uno dei raggi stabiliti dai ProximityAlert, viene spedito un Intent di tipo "ACTION_PROXIMITY_ALERT". Ogni qual volta viene registrata la posizione dell'utente e quest'ultimo si sposta di 10 metri, viene registrata la nuova posizione, poi vengono rielaborati tutti i dati affinché i ProximityAlert siano sempre aggiornati, fornendo all'utente notifiche sempre aggiornate.

2.7. myLocationReceiver

È un BroadcastReceiver che filtra Intent di tipo "ACTION_PROXIMITY_ALERT". Ogni qual volta viene ricevuto questo Intent (tramite onReceive()) viene inviata una notifica all'utente segnalando che è vicino a qualche negozio.

3. Permessi

Per poter utilizzare le funzionalità dell'applicazione, quest'ultima ha bisogno di permessi per la "Fine Location", "Coarse Location" per poter ricevere informazioni sulla posizione dell'utente, dei permessi per "Internet" e "Network State" per poter ricevere informazioni tramite richieste http, dei permessi NFC per poter condividere la lista della spesa, per la "Lettura su storage esterni" e "Scrittura su storage esterni", e i permessi per la vibrazione.

4. Osservazioni

Poiché il consumo di batteria è una priorità per l'utente, l'implementazione non prevede alcun tipo di WakeLock, quindi il Service (Sez. 2.6) lavora in background solo se il dispositivo ha lo schermo acceso.

Davide Montagno Bozzone