Ivan Lo Greco

31/08/2022

Relazione Sviluppo di Applicazioni Mobili

**LevelUp** è un’applicazione di “*Gamification*” delle attività giornaliere (chiamati **Task**), l’utente dovrà inserire le proprie attività da svolgere in un determinato intervallo di tempo, e dopo averla **completata**, gli verranno assegnati dei **punti esperienza** alle **categorie** a cui l’attività si riferiva.

Queste categorie sono realizzate dall’utente stesso e sono assegnabili alle singole attività, dopo il completamento, l’applicazione genererà una quantità di esperienza variabile data da un’algoritmo che tiene in conto la **difficoltà** e l’assegna in maniera equa tra le categorie.

I task creati dall’utente possono essere di **tre** tipi:

* *Generici*, confermati dall’utente stesso.
* *Contapassi*, confermati dal service tramite **sensore**.
* *Prossimità*, confermati dal service tramite delle funzionalità **GPS** di Android.

Single-Activity

L’applicazione adotta un approccio **Single-Activity** navigando tra i **Fragments** mediante il **NavController**. La maggior parte dei fragments fa riferimento ad un proprio *ViewModel* che si occupa di fornire dati per popolare l’interfaccia grafica o aggiornare dati nel database.

I dati sono spesso offerti ai fragments come **LiveData**, che rendono il loro aggiornamento semplice e automatico.

ViewModel

Un **ViewModel** oltre ad offrire funzionalità essenziali per l’interazione col database, *mantiene lo stato in memoria anche dopo la distruzione dei fragments dovuta ad un cambio di pagina o ad un cambio di configurazione.*

Ogni ViewModel utilizza uno o piu **Repository** necessari per interagire con le funzionalità offerte dal database.

Database

L’interazione con il Database è gestita tramite **Jetpack ROOM (ORM)** che permette di creare uno schema a partire da delle classi annotate come *@Entity* in cui al loro interno possono essere stabilite delle *relazioni tra tabelle, chiavi primarie, indici o informazioni necessarie sui singoli campi (o colonne)*.

Il Database si basa quindi su tre strati:

* *Database*.
* Dao.
* Repository.

I **Dao** sono delle interfacce annotate come *@Dao* e permetto di definire dei metodi con associate delle query, l’implementazione di questi metodi viene generata automaticamente da Jetpack durante la compilazione.

Nel caso di LevelUp è stata creata un’interfaccia generica con dei metodi *CRUD*, ogni entità ha il proprio Dao che deriva dal *BaseDao<T>*, quindi ereditando le funzionalità principali, eventuali altre funzionalità possono essere implementate nel dao specializzato.

I **Repository** utilizzano i *Dao* e sono delle classi il cui scopo è svolgere della logica di business durante l’accesso al database, questo perchè alcune funzionalità potrebbero richiedere piu passaggi per completare la richiesta (*salvataggio di un task le sue eventuali specializzazioni e categorie)*, alcune dovranno svolgere delle computazioni (*es. generazione esperienza*).

Nel caso di LevelUp, come prima, è stata realizzata una classe *BaseRepository<E, D>* che offre un’implementazione comune *CRUD* a **tutte** le repository, ogni repository avrà un tipo di *Entità* di riferimento (*E*) e un *Dao* di riferimento (*D*).

I repository possono richiedere altri repository dove necessario, inoltre I repository sono **singleton** gestiti **interamente** dalla classe *BaseRepository*, che se non presente, crea un’istanza e la restituisce.

Step Counter Task

Il Task basato sul **sensore** di contapassi è una possibile specializzazione del task generale, il **goal** in questo caso è il raggiungimento di un certo numero in kilometri, quindi il task risulta completato quando il *numeroDiPassiCorrente \* lunghezzaPasso >= goalKm*.

*LunghezzaPasso* è definito nelle **SharedPreferences** ed è impostabile dall’utente mediante il menu a tendine delle impostazioni.

I passi vengono contati dal sensore **TYPE\_STEP\_COUNTER** che restituisce il numero di passi totali dopo il boot dello smartphone, è presente quindi un *offset* che viene impostato all’inizio del conteggio dei passi per un determinato Task tale per cui *currentSteps = totalSteps – offset*.

Questa logica è gestita all’interno del **Foreground Service** che si occupa di far funzionare i servizi anche se l’applicazione è in **background**.

Localization Task (Proximity)

Il Task basato sulla **prossimità** è anch’esso una specializzazione del task generale, il **goal** è di raggiungere un indirizzo (*basato su latitudine e longitudine*) all’interno di un certo raggio di **X** metri.

Per far ciò sono state utilizzate le librerie di android native (a*ndroid.location*), sia per il **Geocoding** che per il **ProximityCheck**.

*Pur funzionando queste librerie native non garantiscono un’implementazione e questo dipende dai singoli dispositivi, inoltre la qualità degli indirizzi trovati tramite Geocoding non è delle migliori e l’incosistenza del ProximixyCheck (viene chiamata ad intervalli diversi, alcune volte in ritardo di parecchi minuti), sono state utilizzate queste per problemi relativi alla creazione di un account con billing su Google Cloud.*

Simile all’implementazione precedente, questi task vengono gestiti dal Foreground service che si assicura che tutto funzioni anche in background.