

Relazione LAM - Progetto Tracker



Prodotto da :

Rocco Pastore 0000923786

rocco.pastore@studio.unibo.it (INF)

INTRODUZIONE

Tracker è un'applicazione che è in grado di raccogliere informazioni e calcolare statistiche su percorsi effettuati a bordo di una bicicletta, sfruttando i sensori offerti dal proprio smartphone. Queste informazioni potranno poi essere utilizzate per valutare la qualità della superficie su cui si pedala.

PRINCIPIO DI FUNZIONAMENTO

L'applicazione, tramite l'utilizzo dell'Activity Recognition API di Google, è in grado di capire in automatico l'attività svolta dal possessore dello smartphone. Di conseguenza, nel momento in cui l'API riconosce che l'utente è a bordo di una bicicletta, fa partire il processo di raccolta e registrazione dati. Il funzionamento dell'intera App dipende poi dal servizio MyService, in grado di agire, secondo preferenza dell'utente, anche in background ad applicazione chiusa, in modo da raccogliere più dati possibili e nel modo meno invadente.

ACTIVITY RECOGNITION

L'Activity Recognition viene gestita da due servizi :
ActivityRecognitionService, che si occupa di ricevere aggiornamenti sull'attività umana tramite la funzione *activityRecognitionClient.requestActivityUpdates(0,*

pendingIntent), e *ActivityHandlerIntentService*, che comunica direttamente con *MyService* per inviare informazioni sull'attività corrente. La comunicazione avviene attraverso l'intent "activity_intent", con il quale vengono spediti sia il nome dell'human activity rilevata che la confidence, un numero da 1 a 100 che indica quanto sia sicuro che l'activity trovata sia effettivamente quella reale. Nel nostro caso sono state considerate valide solo le human activity con una confidence > 75, per garantire una migliore precisione. Per ricevere l'intent, nel *MyService* viene implementato un *BroadcastReceiver*, che una volta ricevute le informazioni incarica la funzione *handleUserActivity()* di far partire o terminare la registrazione dei dati. Le human activity che vengono considerate sono : bicycle, walking, onVehicle e still, in modo da reagire con precisione ai cambi di activity magari meno netti come da bicycle a walking o da bicycle ad onVehicle.

RACCOLTA DEI DATI

Per raccogliere i dati vengono utilizzati 4 sensori, ovvero accelerometro, giroscopio, magnetometro e sensore di gravità, e la localizzazione fornita da GPS oppure sfruttando la rete.

```
accelerometer =  
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER); gravity =  
sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY); magneticField =  
sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD); gyroscope =  
sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);  
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, (LocationListener)
```

```
listener);  
locationManager.requestLocationUpdates(LocationMan  
ager.GPS_PROVIDER, 0, 0, listener);
```

I dati provenienti dall'accelerometro vengono processati attraverso i dati provenienti dai sensori di gravità e magnetometro, in modo da orientare i primi in base alla posizione dello smartphone rispetto a quella del veicolo. A questo punto siamo pronti a registrare le informazioni, dividendo quelle destinate ad essere caricate e studiate in seguito, e quelle che invece verranno sfruttate dall'applicazione stessa.

REGISTRAZIONE DEI DATI

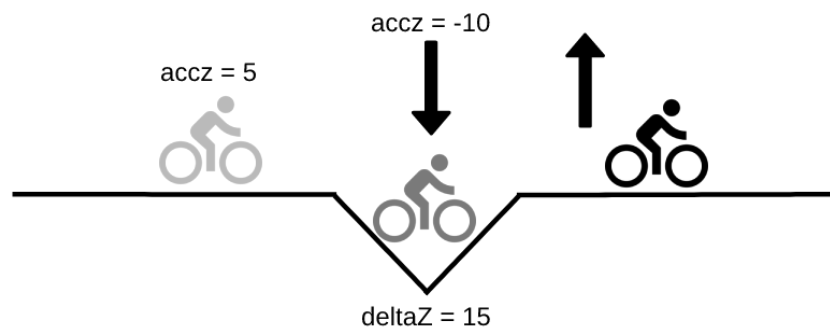
La registrazione vera e propria avviene circa ogni 50 secondi, in modo da non sovraccaricare in maniera eccessiva il dispositivo con variabili di memorizzazione troppo grandi. Utilizziamo a tal proposito due stringhe, a seconda dell'uso che ne dovremo fare, come detto poco prima.

toUpload è la stringa che verrà caricata su un server esterno e che avrà la seguente struttura

`"_accx#accy#accz#gyrx#gyry#gyrz_"` che si ripeterà per ogni cambiamento rilevato dai sensori. I dati del giroscopio non subiscono nessun processo, a differenza dei dati dell'accelerometro. Questa stringa verrà poi caricata tramite la funzione `uploadRide()` insieme a latitudine e longitudine di inizio e fine tratta. Questi ultimi due dati potranno essere utilizzati per isolare il tratto di strada da analizzare.

toWrite è invece una stringa temporanea che verrà utilizzata, dopo alcune analisi, per salvare i dati relativi alle corse all'interno del proprio dispositivo. Essa ha la seguente sintassi :

“_startLat_startLon_deltax#deltay#deltaz_..._speed_stopLat_stopLon” *deltax* rappresenta la differenza tra due rilevazioni consecutive dell’accelerometro sull’asse x. Di fatto verrà utilizzato soltanto *deltaz*, utile per riconoscere deformazioni dell’asfalto come buche o tombini.



ANALISI DEI DATI

Svolta dalla funzione `analysis()`, prende in input la sopracitata stringa `toWrite` e ne analizza ogni elemento, salvando le informazioni su ogni corsa nel formato seguente :

“_startLat_startLon_stopLat_stopLon_percentage0_percentage1_percentage2_asphaltLevel_averageSpeed_evaluation” Per

ottenere queste informazioni, vengono analizzati tutti i *deltaZ*, filtrando quelli considerati non validi (minore di una certa soglia, ovvero 5), e suddividendoli in 3 livelli di gravità : irregolarità trascurabile, irregolarità non grave e irregolarità grave. I limiti di ogni livello sono stati scelti dopo averli valutati a bordo di una bicicletta. Dopo aver completato la suddivisione in livelli, viene trovata la percentuale di ognuno di essi. In base a quest’ultima viene calcolata la qualità generale dell’asfalto o della superficie percorsa : si presuppone che essa sia in media più regolare possibile, per cui percentuali di irregolarità anche non troppo elevate (30% nel nostro caso), determinino un livello di asfalto non ottimale. La velocità media invece viene semplicemente calcolata dividendo la distanza totale percorsa per il tempo.

Secondo alcuni studi, la velocità media in bicicletta nei centri abitati di città come Bologna o Milano, è compresa tra i 7 e i 15 km/h, per cui potremmo utilizzare anch'essa come metro di valutazione, per capire se si riesce a viaggiare in modo scorrevole o meno. Evaluation è la valutazione estetica sul percorso effettuato, quindi sul panorama o sul paesaggio, e che inizialmente è impostata a "not". Una volta raccolti questi dati, viene creato il file vero e proprio che avrà come nome la data precisa dell'inizio della corsa seguito da un numero che ne indica la parte, in quanto, come detto in precedenza, la registrazione avviene ogni 50 secondi e nel caso di corse più lunghe, sarà necessario dividerle in più parti.

AGGIORNAMENTO STATISTICHE

Ogni dispositivo avrà un file **stats**, che manterrà informazioni aggiornate su

"totDistance_averageSpeed_percentageGood_percentageDiscrete_percentageBad", seguite da un identificativo del dispositivo, fornito alla prima installazione dell'applicazione dalla funzione randomUUID(). totDistance ed averageSpeed riguardano rispettivamente la distanza totale e la velocità media considerando tutti i percorsi svolti dall'utente, mentre i tre percentage riguardano le percentuali dei tre livelli di asfalto su cui l'utente ha pedalato.

DATABASE REMOTO

Come detto in precedenza, alcune informazioni vengono caricate su un server remoto, in modo da essere utilizzate per altri scopi ed essere visibili ad altri utenti. In questo caso è stato utilizzato il **RealTime Database** offerto da **Google Firebase**. Sono state

archivate informazioni riguardo le corse nel formato
`rides : { data : xxx, start : xxx, stop :
xxx },`

e riguardo le statistiche degli utenti nel formato
`users : { deviceId : { avgSpeed : xxx,
totKms : xxx } }.`

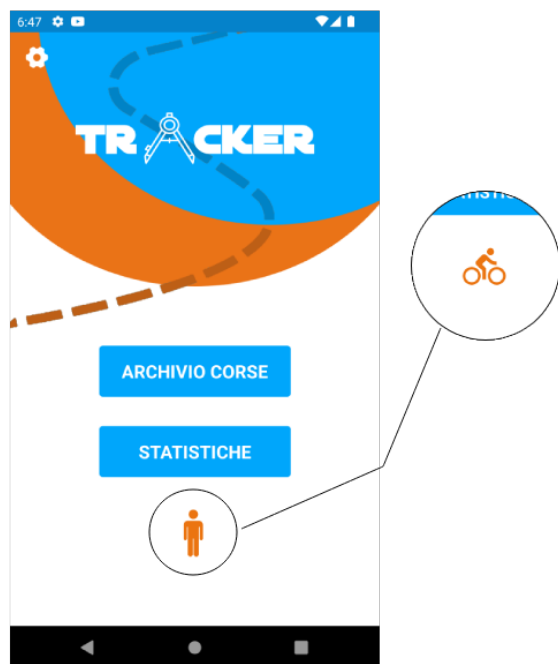
La console è disponibile al link

<https://console.firebase.google.com/u/0/project/tracker-f56ac/database/tracker-f56ac-default-rtdb/data/~2F> e contiene tutti i dati raccolti durante la fase di testing.

Su codice, la comunicazione con FireBase viene effettuata tramite la classe `FirebaseApp`, con la quale viene inizializzata la comunicazione con la funzione `.initializeApp()`. Dopodiché viene richiesta un'istanza del database tramite un apposito link, e da lì è possibile comunicare sia in scrittura, come effettuato nei casi di aggiornamento di statistiche e di caricamento di corse, sia in lettura, nel caso dei confronti svolti nell'activity stats.

Importante notare come venga usata una callback che implementa l'interfaccia `myCallback` quando viene effettuata una lettura dal database, in modo da permettere alla funzione asincrona di ricezione dei dati di completarsi prima di continuare la normale esecuzione del programma, in modo da poter utilizzare tutte le informazioni necessarie.



SCHERMATA HOME



La schermata home, oltre ai pulsanti di navigazione per entrare nelle altre aree dell'applicazione, mostra un'icona che indica l'attuale attività rilevata dall'app. Nel momento in cui mostra l'omino in bici, l'applicazione sta registrando i dati.

ARCHIVIO CORSE

Per elencare le corse è stato utilizzato un CardAdapter che riempie dinamicamente una lista. Ognuna di esse è individuata da una Card che, oltre al nome del file, mostra informazioni in breve sulla corsa e due icone sui due lati : a sinistra il cerchio indica la qualità generale del percorso, ottenuta in base al livello dell'asfalto; a destra è invece presente un bottone che dà accesso ad un BottomSheetDialogFragment, nel quale compare anche una RatingBar che permette di valutare l'aspetto panoramico del percorso.

| Archivio corse | |
|---|---|
|  Strada percorribile in modo scorrevole, con velocità media di 47.12 km/h Condizioni asfalto : Ottime |  |
| 11-11-2022 04:18:59_4 | |
|  Strada percorribile in modo scorrevole, con velocità media di 10.54 km/h Condizioni asfalto : Ottime |  |
| 11-11-2022 04:18:59_3 | |
|  Strada percorribile a velocità sotto la media : 6.980 km/h Condizioni asfalto : Ottime |  |

Archivio corse

12-11-2022 03:30:47_0

○

Strada percorribile a velocità sotto la media : 4.240 km/h

Condizioni asfalto : Ottime

12-11-2022 03:30:34_0

○

Strada percorribile in modo scorrevole, con velocità media di 35.96 km/h

LUOGO DI PARTENZA

44.5004, 11.3235

LUOGO DI ARRIVO

44.5004, 11.3235

VELOCITÀ MEDIA

4.240 km/h

IRREGOLARITÀ ASFALTO RILEVATE

NON GRAVI

6.0%

GRAVI

0.0%

VALUTAZIONE VISIVA DEL PERCORSO

★ ★ ★ ★ ★

BottomSheetFragment che si apre al click sul bottone a destra e che mostra i dettagli della corsa

Quando viene settata una valutazione visiva, comparirà a schermo un Toast che confermerà l'avvenuta registrazione del voto e in seguito verrà riaggiornata l'Activity, in modo da rendere immediatamente visibili le modifiche effettuate. Anche la valutazione visiva potrà poi essere utilizzata come metro di valutazione dei percorsi effettuati.

STATISTICHE

In questa pagina viene mostrato il contenuto del file stats. In alto sono state utilizzate delle ProgressBar per mostrare dettagli sulle qualità delle strade percorse.

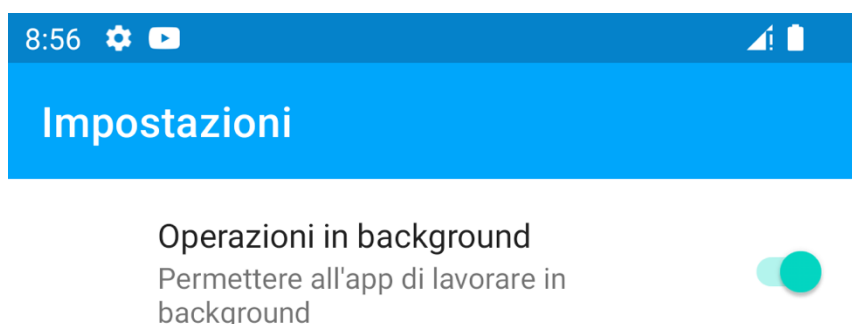
In più vengono confrontate le proprie statistiche con quelle degli altri utenti, scaricando le informazioni necessarie con Firebase.



IMPOSTAZIONI

La PreferencesActivity fa uso del PreferencesFragment per la gestione delle preferenze dell'applicazione. In

questo caso è presente soltanto uno SwitchPreferenceCompat che riguarda la preferenza sull'utilizzo dell'applicazione in background. Se essa sarà inattiva, allora ogni qual volta l'utente uscirà dall'Activity principale, il servizio MyService verrà fermato.



TEST

L'applicazione è stata testata sia su AVD che tramite un dispositivo fisico montato sul manubrio di una bicicletta sportiva.

I dispositivi in questione sono :

- AVD Pixel 2 – Android 12 – API 31
- Samsung Galaxy S6 – Android 7 – API 24

L'applicazione è risultata capace di raccogliere dati in maniera sufficientemente precisa e di distinguere determinati percorsi irregolari da altri più regolari.