




ANNO ACCADEMICO 2019/2020

Relazione Elaborato

Advanced Programming Languages

DAVIDE DIGENTI – 055000372
GABRIELE RUIZ – 055000351



Sommario

Obiettivi dell’elaborato 2

Architettura..... 3

Scelte Implementative 4

Casi D’Uso 5

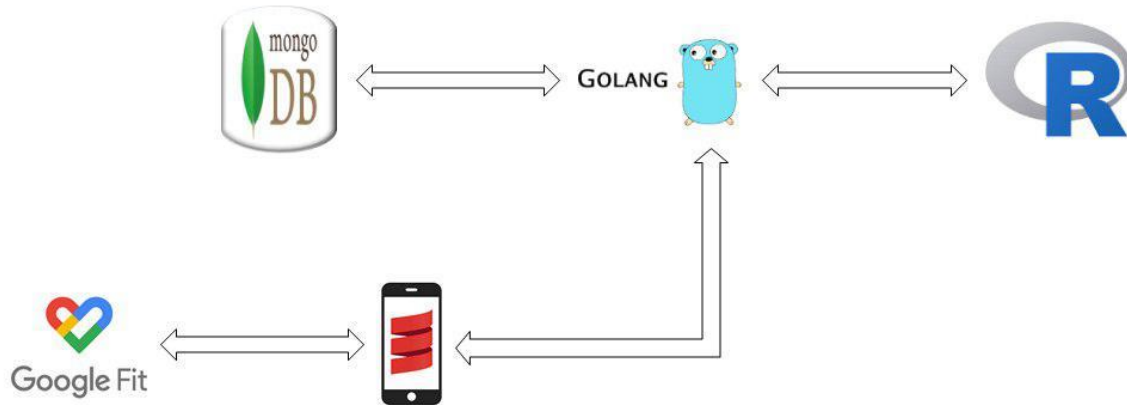
Obiettivi dell'elaborato

L'obiettivo dell'elaborato riguarda lo sviluppo di una piattaforma in grado di tracciare i percorsi effettuati dagli utenti durante le loro sessioni di jogging. Vengono raccolte quindi le coordinate relative ad una sessione avviata e terminata dall'utente, in modo da tracciare sulla mappa i percorsi più comuni con relativo orario di percorrenza. Oltre a ciò verrà mostrato all'utente lo storico dei passi effettuati.

Ciò viene realizzato sfruttando l'accesso alla sensoristica dello smartphone Android, che è in grado di restituire longitudine e latitudine. I dati raccolti vengono inviati ad un server che inizialmente si occuperà esclusivamente di memorizzarli.

I dati raccolti dal server verranno inviati periodicamente ad un ulteriore server che si occuperà di delineare i percorsi.

Architettura



L'architettura proposta è composta da diverse entità: l'applicazione Android, il server in Go ed infine un ulteriore server in R.

L'applicazione Android rappresenta un client che raccoglie dati all'interno di una sessione. I dati espressi sotto forma di **latitudine, longitudine e timestamp** vengono inviati al server principale il quale espone delle **REST API**. Oltre a ciò, l'applicazione si occuperà anche di richiedere i dati relativi ai percorsi più vicini alla sua posizione, in modo da poter avere una visione dei luoghi più frequentati vicini all'utente per lo svolgimento di attività fisica.

Il server in Go si occuperà di memorizzare i dati raccolti in un database. Periodicamente inoltre, si occuperà di raccogliere i dati presenti sul database e inviarli al server R che a sua volta esporrà delle REST API.

Il server in R, una volta ricevuti i dati, si occuperà di analizzare quanto ricevuto e tramite un algoritmo di clustering fornirà un certo numero di percorsi, che verranno inviati al server Go il quale a sua volta si occuperà di memorizzare il tutto sul database.

Scelte Implementative

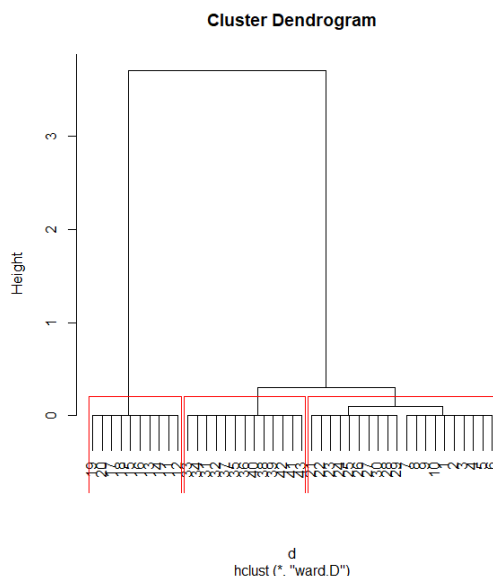
Per il seguente elaborato abbiamo realizzato delle applicazioni in Scala, GoLang e R, sfruttando inoltre MongoDB per il database NoSQL e le API di Google Fit per lo storico dei passi.

Per quanto riguarda l'applicazione **Scala**, si è scelto di integrarla in Android vista la semplicità di raccogliere dati da un dispositivo mobile. Inoltre, il contesto applicativo è pensato per svolgere attività in tutta comodità, con la necessità di dover semplicemente avviare la sessione dallo smartphone.

I dati relativi alla sessione dell'utente verranno incapsulati in un file **JSON** che verrà inviato al server Go tramite POST. Qualora l'utente volesse visualizzare i percorsi più vicini, verrà effettuata una GET al server Go, che restituirà i dati in formato JSON. Inoltre, per ciascun percorso, l'utente potrà richiedere informazioni dettagliate sugli orari di percorrenza, effettuando un'altra Get al server Go il quale restituirà un ulteriore JSON. Infine, sfruttando le API di **Google Fit**, sarà possibile per il client visualizzare lo storico dei passi effettuati durante il giorno, la settimana ed il mese corrente.

GoLang è stato usato per realizzare il server principale, che si occupa di memorizzare i dati degli utenti. Tali dati, visto la loro mole e l'elevata frequenza con cui vengono prodotti, possono essere considerati **big-data**, quindi la scelta più naturale è stata quella di utilizzare un database NoSQL, in particolare **MongoDB**. Questi big-data verranno periodicamente inviati al server R usando una *GoRoutine*. La comunicazione tra i due server avviene sempre tramite REST API.

R è stato infine sfruttato per la realizzazione di un ulteriore server, il quale si occupa esclusivamente dell'analisi dei dati. Per calcolare i percorsi a partire dai passi si è scelto di sfruttare un algoritmo di **clustering gerarchico**, in particolare si è sfruttato il criterio di *varianza minima di Ward*, il quale minimizza la varianza totale intra-cluster. Abbiamo sfruttato il package *NbClust*, che utilizza 30 diversi indici per determinare il numero ottimale di cluster al variare di parametri quali il numero minimo e massimo di cluster, criterio di misura della distanza (*Euclidea*), e metodo di analisi da utilizzare (*Ward*). Per determinare il numero massimo di cluster abbiamo considerato il numero di passi totali diviso una stima di passi minimi per sessione (500).



Questo è il risultato dell'analisi di clustering: i dati in ingresso vengono raggruppati, a partire da un numero di cluster pari al numero di passi, e via via l'algoritmo andrà a convergere verso il numero ottimale di cluster calcolato precedentemente.

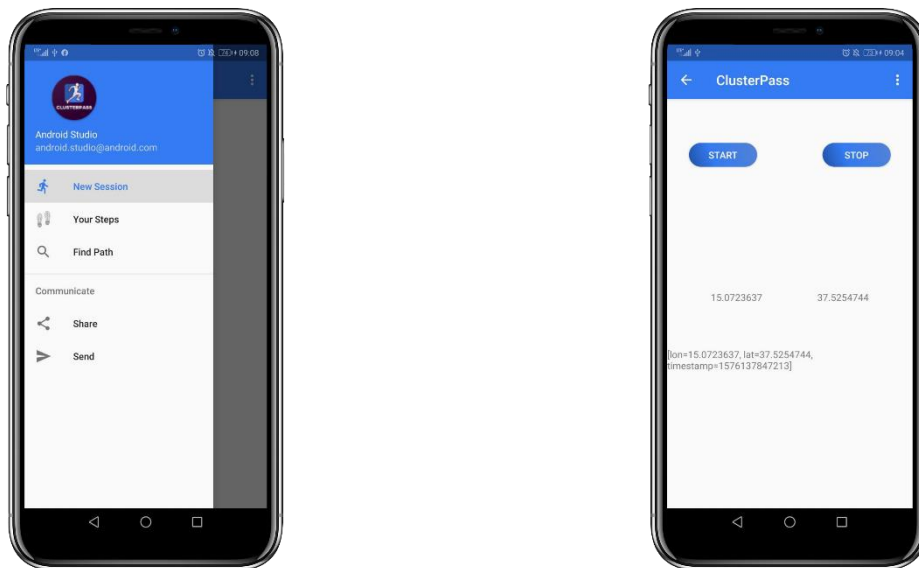
Nel nostro caso, il numero ottimale di cluster risulta essere tre, e di conseguenza in rosso vengono evidenziati i cluster effettivi, che in definitiva rappresentano i tre percorsi.

Casi D'Uso

Nei casi d'uso, mostriamo le tre funzionalità dell'applicazione:

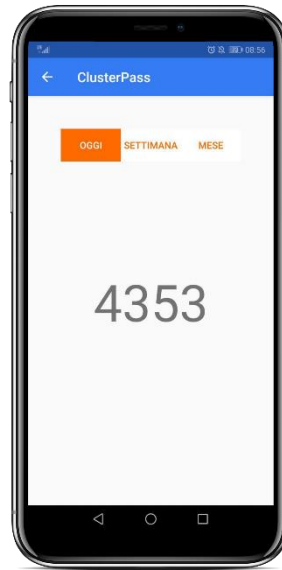
1. Avvio nuova sessione
2. Storico dei passi
3. Trova percorsi

Avvio Nuova Sessione



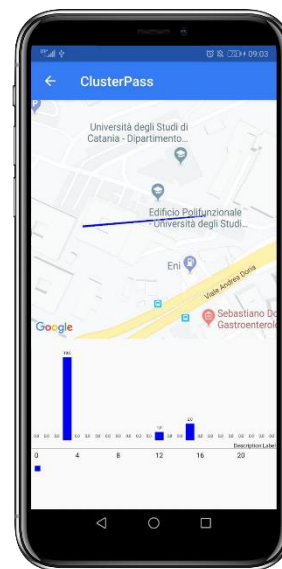
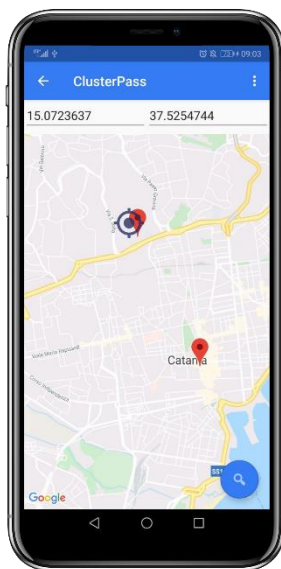
In questo caso viene mostrato l'avvio di una sessione, con conseguente raccolta dei dati inerenti alla latitudine, longitudine e timestamp da processare ed inviare al server. L'utente quando avrà intenzione di avviare una sessione dovrà premere sul pulsante Start, ed una volta finita la sessione dovrà cliccare sul pulsante Stop. I dati vengono raccolti a *spot* ed incapsulati e inviati alla fine della sessione.

Storico dei passi



In questo caso viene mostrata l'integrazione con Google Fit per avere accesso ad uno storico riguardante i passi effettuati nella giornata odierna, nella settimana o nel mese corrente.

Trova Percorsi



In questo caso viene mostrata la possibilità di vedere i percorsi vicini all'utente e cliccando sul marker di ciascuno viene mostrata una mappa in cui è tracciato il percorso ed un istogramma che mostra la distribuzione oraria delle percorrenze.