Slide 1

Generocity è un applicazione mobile sviluppata su sistemi ios e android. In particolare il mio lavoro è stato fatto sulla versione android. L’applicazione ha come scopo principale quello di permettere a un utente di cercare un parcheggio e scambiarsi con chi desidera lasciarlo. Inoltre, l’utilizzo di questa applicazione ha come obiettivi quello di ridurre il tempo alla ricerca di un parcheggio, e di conseguenza i costi del carburante, riducendo così anche il traffico. Tra le funzionalità principali ne abbiamo due: gestione delle automobili e scambio di parcheggio.

Slide 2

Per gestione di automobili si intende la possibilità di creare un auto da poter utilizzare all’interno dell’applicazione aggiungendola o con la targa oppure collegando un dispositivo bluethoot. Dopo di che sarà possibile inserire tutte le informazioni sulla auto come si vede in figura: nome auto, marca ecc… Sarà dunque possibile poi modificare o eliminare una auto quando si vuole

Slide 3

Lo scambio di parcheggio avviene tra due utenti: l’utente A decide che vuole lasciare il parcheggio in una determinata zona, in cui ha parcheggiato la auto, a un determinato orario; l’utente B decide di voler cercare un parcheggio allo stesso orario e stessa zona e a questo punto il sistema creerà quello che chiameremo match tra i due utenti. Successivamente i due procederanno allo scambio.

Slide 4

LeaveAndSearchParkScheduleFragment è una classe che gestisce l’interfaccia mostrata in figura. Questa interfaccia mostra in alto un tasto “adesso” che permette all’utente di lasciare il parcheggio al primo schedule disponibile. Uno schedule è sostanzialmente un orario a scaglioni di 15 minuti. Sotto il tasto adesso ci sarà una lista di schedule inizializzati dall’orario corrente (ad esempio nell’immagine sono le 13.50 e quindi gli schedule vanno dalle 14.00 in poi). Affianco a ogni schedule sarà possibile visionare un elenco di utenti che hanno deciso di cercare il parcheggio a quell’ora cosi da facilitare la scelta dell’ora in cui lasciare, sapendo appunto che ci sarà uno scambio.

Slide 5

Che cos’è un giver… Un utente che ha deciso di lasciare il posto di parcheggio per il sistema diventerà un Giver. Per diventare giver ci sono diversi modi: uno di questi è cliccare su uno schedule e a questo punto l’app farà una chiamata API di createGiver() per comunicarlo al sistema

Slide 6

Un altro modo per diventare Giver è cliccare sul tasto adesso e analogalmente a come succede nella slide precedente l’utente diventa Giver per il primo schedule disponibile.

Slide 7

Per realizzare l’elenco di schedule cliccabili è stato inizializzato un array di valori presi dall’ora corrente, mi spiego meglio: nel codice vengono presi ora e minuti correnti; uno schedule è valido solo se è nel futuro, motivo per il quale viene fatto un arrotondamento al primo quarto d’ora successivo; dopodiché viene fatto un normale ciclo while sulle ore, da ora corrente calcolata a 23, e di minuti a scaglioni di 15. Ognuna di queste coppie forma uno schedule che viene inserito nell’array che mostra gli schedule

Slide 8

All’interno di ogni elemento della lista, come detto precedentemente, è presente la lista di utenti che hanno deciso di cercare per quell’ora e quella destinazione. Questo viene realizzato attraverso la chiamata API GetTakerList (un Taker è la controparte del giver, ovvero chi sta cercando parcheggio). Questa chiamata ritorna una lista di Taker dai quali poi verranno estrapolati il nome e assegnati agli schedule rispettivi

Slide 9

In questa slide è presente un grafo che spiega tutto il meccanismo dei match. Si parte da due utenti che vogliono diventare take e giver rispettivamente. Essi hanno la possibilità di diventare Taker o Giver attraverso la chiamata API di createTaker e createGiver. A questo punto questi utenti saranno in uno stato di waiting nel quale sono in attesa di un match. Gli utenti da questo stato possono interrompere la ricerca in qualisasi momento tornando allo stato precedente. Dallo stato di waiting sarà possibile andare in due stati: expired e running. Lo stato expired è quello stato in cui un match in stato di waiting o anche di running sono scaduti da 15 minuti, a quel punto il match viene cancellato e si ritorna allo stato di partenza. Lo stato di running è lo stato in cui due utenti si sono trovati in un match. Da questo stato come in waiting possono annullare il match ma verranno creati altri due stati: unsuccess-giver e unsuccess-aker. Questi due stati non fanno altro che dire all’utente:”guarda il match che avevi in corso è fallito perché la controparte ha annullato quindi ti rimetto in ricerca in waiting”. Da Running poi sarà possibile completare un match e andare nello stato di success dove finalmente i due utenti hanno scambiato il posto e torneranno allo stato di partenza.

Slide 10

La situazione dello stato del match viene gestito con una grafica all’interno delle cards. Questo è stato realizzato utilizzando una serie di if a cascata per ogni macchina posseduta: se la macchina ha un match in stato di waiting allora mostrera la card di questo stato, stessa cosa per running e success. Per l’unsuccess-taker e giver quello che verrà fatto è l’eliminazione di tale match per crearne uno identico e tornare in stato di waiting. Con expired semplicemente match annullato. Altrimenti se gli altri if sono falsi, verrà, mostrata la card standard.

Slide 11

Qui viene mostrata la card standard nelle due forme per il lascia posto: se un utente non ha parcheggiato, esso non potrà giustamente lasciare il posto. Se è parcheggiata, cliccando verrà mostrata l’interfaccia degli schedule precedentemente spiegata.

Slide 12

Questa è la card in stato di waiting distinguendo quella del taker e del giver. Le informazioni scritte all’interno delle card sono salvate grazie all’utilizzo di una classe chiamata SchedulePositionUtils e il tasto annulla, annullerà il match con le chiamate API di removeTaker e Giver

Slide 13

Questa è la card per lo stato di running come prima distinta per taker e giver. I due possono annullare in qualsiasi momento come detto prima dal tasto annulla, le informazioni su chi sta arrivando o con chi ti sta scambiando di posto sono prede dalla chiamata API getCarByCid. Il taker cliccando sul tasto SONO QUI completerà il match avvisando il Giver che lui è arrivato e puo andarsene

Slide 14

Questa è la card in stato di success. È una card di notifica di scambio avvenuto con successo, cliccando ok torneranno alla card standard e vengono mostrati i generocoins ottenuti/spesi e i rimanenti. I generocoins, non sono stati spiegati ma sono il meccanismo principale per far funzionare l’applicazione. Come si puo vedere da queste immagini, chi ha cercato posto ha speso punti, chi lo ha lasciato ha guadagnato punti. Questo vuol dire che per permettere a un utente di cercare posto dovrà per forza lasciarlo a sua volta evitando comportamenti egoisti e facendo infine funzionare l’app.

Slide 15

Infine sono mostrati alcuni sviluppi futuri inerenti alla mia funzionalità quali: l’implementazione di un timepicker per la scelta dello schedule in linea con gli standard del material design android; mostrare sulla mappa il Taker e il Giver con il relativo percorso in tempo reale per permettere ai due di avere delle informazioni utili; l’update delle cards quando si passa da uno stato all’altro grazie alle notifiche firebase del server che attualmente non sono state ancora implementate.

La presentazione è finita. Grazie dell’attenzione.