Progetto: “Maledetta TreEst”

Insegnamento di Mobile Computing

Versione 1.2

Differenze rispetto alla versione precedente: a) il nome utente non è univoco; b) il campo did è specificato per la stazione di arrivo, non per quella di partenza. Le modifiche sono evidenziate in blu.

Introduzione

TreEst è una compagnia di trasporto ferroviario (chiaramente TreEst è un nome inventato). Purtroppo i treni di TreEst sono spesso in ritardo, affollati o addirittura soppressi. Oltre a questi problemi, i passeggeri (soprattutto i pendolari che usano tutti i giorni TreEst) lamentano il fatto che TreEst fornisca scarse informazioni sullo stato dei treni.

Per affrontare questi problemi, i passeggeri si sono organizzati e hanno realizzato il sistema “Maledetta TreEst”. Lo scopo del sistema è permettere ai passeggeri di condividere tra di loro le informazioni sullo stato dei treni.

Dati del sistema

- Linea: TreEst gestisce varie **linee** (*line*) ferroviarie. Ciascuna linea è composta da una lista di **stazioni** (*station*) ciascuna con un nome (*sname*) e coordinate geografiche (*lat* e *lon*). La prima e l’ultima stazione (i due capolinea) definiscono il nome della linea. Es: una linea che ha come stazioni [“Abbiategrasso”, “Baggio”, “Como”, “Domodossola”] è la linea “Abbiategrasso-Domodossola”. Per ogni linea esistono due **tratte** (*direction*) identificate dal un id (*did*). Nell’esempio precedente, per la linea “Abbiategrasso-Domodossola” esistono le tratte “Abbiategrasso-Domodossola direzione Abbiategrasso” e “Abbiategrasso-Domodossola direzione Domodossola”. Nota: alcune stazioni possono far parte di più linee.
- **Utente** (*user*): è identificato da un nome (*uname*) e contiene un’immagine di profilo opzionale (*upicture*), che deve essere codificata in Base64 e il numero di versione dell’immagine di profilo (*pversion*) che all’inizio (quando l’utente non ha ancora un’immagine) è uguale a zero. Ogni utente ha altri due identificatori: un numero di sessione (*sid*) che è noto solo all’applicazione dell’utente stesso e funge da credenziali per l’accesso e da un identificato utente (*uid*) che è noto agli altri utenti.

• Per ciascuna tratta di ciascuna linea è definita una **bacheca** (*board*) di **post** (*post*). Ciacun post è identificato da un **identificatore** numerico (*pid*), è creato da un utente (*author,* lo uid dell'utente che ha creato il post), e contiene informazioni sul **ritardo** (*delay*, 0: in orario, 1: ritardo di pochi minuti, 2: ritardo oltre i 15 minuti, 3: treni soppressi) e **stato** del viaggio (*status*, 0: situazione ideale, 1: accettabile, 2: gravi problemi per i passeggeri) e un **commento** (*comment*). Le tre informazioni (ritardo, stato e commento) sono tutte e tre opzionali, ma almeno una di queste deve essere riportata (altrimenti avremmo un post vuoto).

• Tra gli utenti è definita una relazione di "**segue**" (*follow*): la relazione è asimmetrica (l'utente A può seguire l'utente B senza che B segua A) e A può iniziare a seguire B senza che B fornisca la propria autorizzazione.

Descrizione del sistema

Il sistema è composto da un server (fornito dal docente), e da un client, che dovrà essere realizzato dagli studenti (sia in android sia in cross-platform). Le funzionalità descritte nel seguito dovranno essere accessibili via applicazione mobile.

• **Registrazione implicita.** Ogni utente dispone di un numero di sessione che lo identifica rispetto al server. Al primo avvio l'applicazione richiede un numero di sessione al server e poi lo memorizza in modo persistente. In tutte le comunicazioni tra client e server, il client indica il proprio numero di sessione. L'utente non inserisce mai a mano il proprio numero di sessione (l'utente non sa neanche cosa sia un numero di sessione) né effettua mai un'azione esplicita di registrazione. L'utente non sa nulla di login o registrazione, l'app fa tutto senza chiedere nulla all'utente.

• **Profilo.** L'utente ha la possibilità di specificare un nome utente e un'immagine di profilo. Le immagini di profilo sono associate ad un numero di versione, che viene incrementato ogni volta che un utente cambia la propria immagine. L'utente non vede le informazioni relative al numero di versione.

• **Scelta della bacheca.** L'utente deve poter scegliere quale bacheca visualizzare (ricordati, c'è una bacheca per ogni tratta). La scelta avviene così: se l'utente in precedenza ha già visualizzato una bacheca, quella stessa bacheca deve essere visualizzata di default. Deve essere possibile cambiare rapidamente tratta sulla stessa linea. Es: se l'utente visualizza la bacheca della tratta "Abbiategrasso-Domodossola direzione Abbiategrasso" deve poter

rapidamente passare alla tratta “Abbiategrasso-Domodossola direzione Domodossola”. Al primo avvio, o comunque quando l’utente lo desidera, l’utente può selezionare una tratta diversa scegliendo tra tutte le tratte disponibili mostrate in una lista.

• **Visualizzazione della bacheca.** Quando l’utente visualizza una bacheca vede l’elenco degli ultimi post. Per ciascuno viene visualizzata l’immagine, il nome dell’autore la data e ora di pubblicazione e, se disponibili, i seguenti dati: commento, ritardo, stato. È compito del server decidere quali siano gli “ultimi post”. Il client deve mostrare tutti i post inviati dal server. I post degli utenti seguiti sono mostrati in evidenza. Da ogni post, l’utente può scegliere se iniziare a seguire o smettere di seguire l’autore del post. L’utente può inserire un nuovo post.

• **Dettagli tratta.** Dalla pagina della bacheca l’utente può visualizzare i dettagli di una tratta: una mappa mostra le stazioni e, se possibile, la posizione corrente dell’utente.

È richiesto agli studenti di progettare la UX e UI in modo coerente con il contesto applicativo dell’applicazione, che può essere definito dallo studente stesso. Per esempio lo studente può dare un look and feel più o meno formale al sistema, adattandolo a quello che pensa sarà il suo uso principale. È comunque necessario personalizzare l’interfaccia grafica.

Comunicazione con il server

Le API rese disponibili dal server sono documentate nel seguito.

URL base: https://ewserver.di.unimi.it/mobicomp/treest/

Tutte le chiamate indicate nel seguito devono essere seguite da “.php” e restituiscono:

• Il codice di stato HTTP 200 in caso di richiesta corretta;

• Il codice di stato HTTP 400 nel caso in cui non siano forniti i parametri specificati nel seguito oppure se i parametri non sono corretti (es: manca un parametro);

• Il codice di stato HTTP 401 nel caso in cui sia fornito un numero di sessione non valido.

• Il codice di stato HTTP 413 nel caso in cui i dati passati come parametro siano troppo lunghi.

Chiamata	Input*	Output e descrizione
register		Il numero di sessione. Viene creato un nuovo utente senza nome e senza immagine di profilo.

Chiamata	Input*	Output e descrizione
getProfile	sid	Ritorna i dati dell'utente (uid, name, picture, pversion). L'immagine di profilo è codificata in base64.
setProfile	sid, name, picture	Non ritorna nulla. Aggiorna i dati di profilo. Sia "name" che "picture" sono opzionali ma almeno uno dei due deve essere specificato. Se viene cambiata l'immagine, il numero di versione viene incrementato. ~~Se viene indicato un nome già usato da un altro utente, viene ritornato un errore 400~~.
getLines	sid	Ritorna l'elenco di linee, ciascuna con l'elenco di stazioni. Per le stazioni di capolinea viene ritornato anche il *did* della tratta che termina in quella stazione. Esempio: {"lines":[{ "terminus1":{"sname":"Milano Celoria","did":1}, "terminus2":{"sname":"Milano Rogoredo","did":2}}, { "terminus1":{"sname":"Milano Lambrate", "did":3}, "terminus2":{"sname":"Sesto San Giovanni","did":4}}]}
getStations	sid did	Ritorna l'elenco ordinato delle stazioni per la direzione indicata. Esempio: {"stations": [{"sname":"Milano Celoria","lat":"45.47669","lon":"9.23217"}, {"sname":"Milano Lambrate","lat":"45.4847","lon":"9.23688"}, {"sname":"Milano Forlanini","lat":"45.4643","lon":"9.2387"}, {"sname":"Milano Rogoredo","lat":"45.43331","lon":"9.241"}]}
getPosts	sid, did	Ritorna un array di post. Esempio: {"posts":[{"delay":1,"status":1,"comment":"prova","followingAuthor":true,"datetime":"2021-11-02 13:42:36.215414", "authorName":"Pluto", "author": 21, "pversion":"0"}, {"comment":"ciao a tutti","followingAuthor":false,"datetime":"2021-11-02 13:43:22.501043", "authorName":"Pippo", "author": 27, "pversion":"2"}]}
addPost	sid, did, delay, status, comment	Inserisce un nuovo post nella bacheca indicata. Delay, status e comment sono tutti opzionali, ma almeno uno deve essere indicato. Delay è un intero in [0,3]. Status è un intero in [0,2]. Non ritorna nulla.
getUserPicture	sid, uid	Ritorna, per l'utente con lo uid idicato: l'immagine di profilo codificata in base64, lo uid dell'utente e la versione dell'immagine. Esempio: {"uid" : "456", "pversion" : "4" , "picture" : "5asdfJHKJHG…"}
follow	sid, uid	Indica al server che l'utente associato al sid indicato inizia a seguire l'utente con lo uid indicato. Non ritorna nulla
unfollow	sid, uid	Indica al server che l'utente associato al sid indicato smette di seguire l'utente con lo uid indicato. Non ritorna nulla

*Tutti i parametri in input e output sono oggetti JSON.

Semplificazioni

Si osservi che il sistema è un prototipo e dunque molti aspetti non strettamente necessari sono ignorati. Questo include:

- Le tecniche per garantire la sicurezza sono basiche e non sono adottate tecniche di protezione della privacy.
- Il sistema non è protetto contro client che hanno un comportamento malevolo.

• Si richiede di implementare solo le funzionalità descritte in questo documento; molte funzionalità generalmente necessarie non sono richieste (es: uso dello stesso account su più dispositivi).

• Si richiede di implementare il client in una sola lingua (Italiano o Inglese, a scelta dello studente) ma l'implementazione deve essere consistente (non una parte in italiano e una in inglese).

• Tutte le immagini di profilo devono essere quadrate.

• Non si deve assumere che i dati inseriti dagli altri utenti siano corretti. Ad esempio le immagini potrebbero essere codificate male, i dati di posizione errati. Il server non effettua controlli e trasmette quello che viene caricato dagli altri utenti.

• Nel progetto non sono adottate tecniche finalizzate a migliorare le prestazioni (se non la gestione delle immagini, vedi sotto). Per ovviare a questo ed ad altri problemi:

  • si possono caricare solo immagini (di profilo) inferiore a circa 100KB cioè stringhe in Base64 più corte di 137.000 caratteri.

  • I commenti devono essere più brevi di 100 caratteri.

  • I nomi utenti devono essere più corti di 20 caratteri.

• Le uniche tecniche di ottimizzazione delle prestazioni sono relative alla gestione delle immagini che devono essere salvate su memoria persistente e lette da rete solo quando non ancora disponibili sulla memoria persistente del dispositivo mobile.

• Nel progetto non sono considerati i problemi di sincronizzazione distribuita.

• È richiesto di realizzare l'app solo per smartphone, non è necessario progettare e realizzare l'interfaccia grafica per tablet.

Osservazioni importanti

È vietato:

• caricare immagini o altri dati personali degli studenti o di altre persone (tutti i dati devono essere inventati).

• caricare immagini e/o messaggi di testo inappropriati o protetti da copyright.

• Caricare dati con l'intento di causare comportamenti errati sul server o sui client degli altri studenti.

• Utilizzare numeri di sessione di altri utenti.

• Usare il server come strumento per la condivisione di soluzione durante la prova d'esame (a tale proposito, avrete notato che non è possibile cancellare i dati).

L'inosservanza delle disposizioni riportate sopra comporta l'impossibilità dello studente responsabile a partecipare alle prove d'esame per un numero di appelli deciso dal docente e l'eventuale segnalazione di quanto accaduto al consiglio di coordinamento didattico.

Tenete presente che la vostra applicazione riceverà dati forniti anche da altri utenti. Dovete ad esempio gestire il caso in cui l'immagine sia trasmessa in modo errato. Per convenzione, decidiamo che l'immagine Base64 inviata al server non deve contenere il prefisso HTML per le immagini.

Il docente cancella periodicamente i dati inseriti (es: tutte le bacheche). Se possibile, il docente avvisa preventivamente gli studenti. Non è necessario che l'applicazione gestisca il caso in cui i contenuti siano cancellati durante l'utilizzo dell'applicazione stessa.

Gli studenti possono creare i dati che servono loro per il test, ma sono invitati a non creare una quantità eccessiva di dati. Se uno studente, per errore, crea un grande quantitativo di dati, è invitato a contattare il docente, che li cancellerà.

Testing

Durante le prove d'esame, l'app sarà provata su simulatore (non su browser né su dispositivo fisico). Il modello di dispositivo simulato e la versione delle API è a discrezione degli studenti.

Nuove versioni del progetto

Nel caso in cui fosse necessario correggere il presente progetto, ne saranno rilasciate altre versioni. L'informazione sarà comunicata agli studenti. È compito degli studenti fare in modo che il proprio progetto aderisca all'ultima versione del presente documento.