Introduzione

Descrizione del progetto

L'obbiettivo di questo progetto è quello di fornire un servizio che permetta a vari utenti di cercare e visualizzare sulla mappa il miglior tragitto da percorrere tra due località. Per un utente che sia registrato nel sistema è anche possibile salvare le proprie località preferite e se il tragitto da percorrere prevede l'utilizzo di servizi pubblici della Trentino Trasporti e possibile comprare il biglietto addebitando una carta di credito od un equivalente metodo di pagamento. Il sistema tiene evidenza dei biglietti comprati e pagati.
Al servizio si può accedere sia tramite webBrowser che tramite un applicativo rilasciato in forma di "App" per iPad.

Analisi dei requsiti

System Users ("actors")

Gli attori di questo sistema sono tutti gli utenti che necessitano di conoscere il miglior tragitto da percorrere per raggiungere una determinata località, nota la località e l'orario di partenza. Se si intende utilizzare un mezzo pubblico viene proposta le combinazione di autobus o altri mezzi che conviene utilizzare per il tragitto specificato. Se il tragitto prevede l'utilizzo di uno o più mezzi pubblici si può acquistare il biglietto direttamente on line prima di salire sul mezzo. L'utente viene notificato dell'avvenuto pagamento con una mail di conferma eventualmente da mostrare al controllore.

System entities ("things")

Le entità del sistema sono le classi che forniscono il mapping del Database e che vengono utilizzate dai EJB per accedere al DB.
Le entita riguardanti il Database risultano essere: Account, Conto, CostoTariffaExtraUrbano, LocalitàPrivata, LocalitòPubblica, Movimento e Rc(classe di supporto utilizzate per la gestione dei errori e la sessione).
Un'altra entità utilizzata è la TripRequest che serve per il passaggio delle informazioni riguardanti il tragitto da ricercare tramite le google directions API.

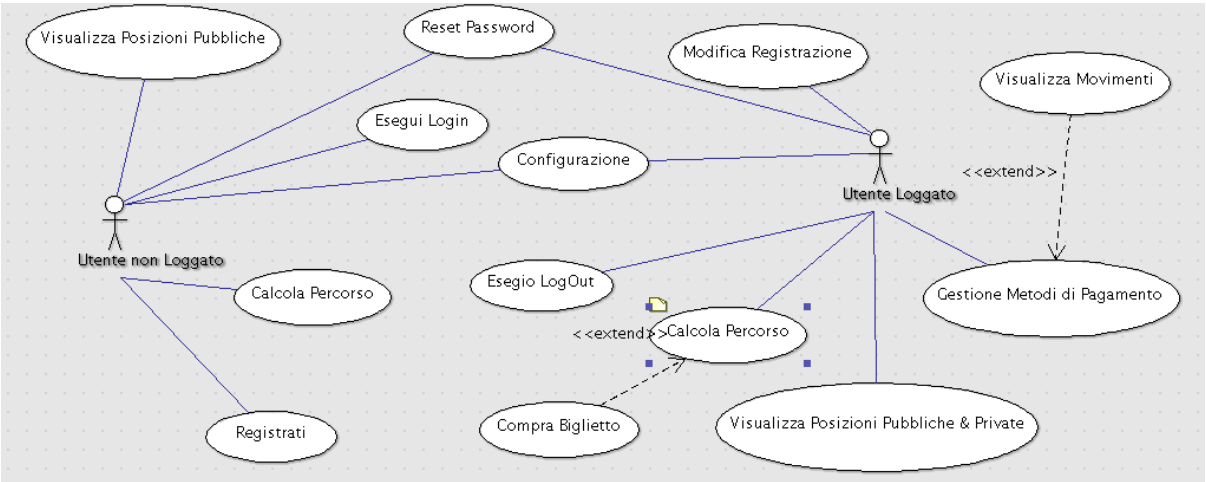
Possible user actions

L'utente oltre a compiere le "classiche" operazioni di LogIn, Registrazione, Reset della password e LogOut, può anche:

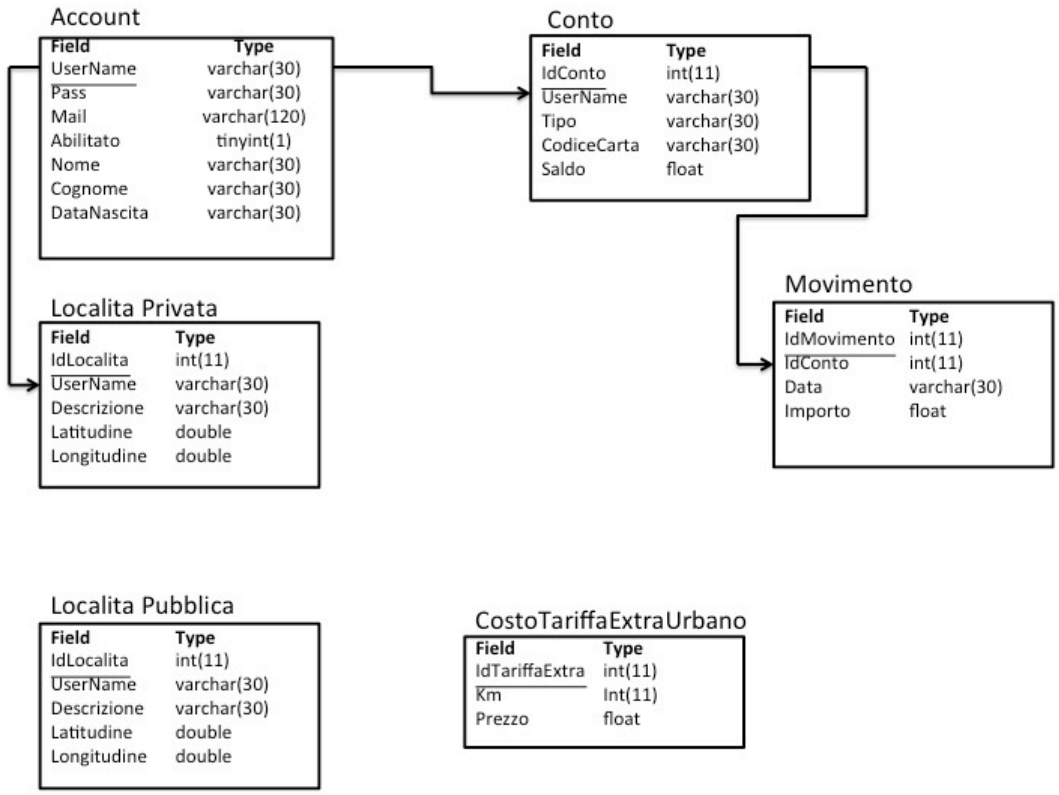
- Calcolare un percorso: questa azione permette di calcolare e di visualizzare il miglior tragitto da percorrere per raggiungere la destinazione desiderata da un punto di partenza.
  Se l'utente risulta essere registrato e accreditato, gli è permesso di selezionare come punti di partenza o arrivo le sue località preferite e, se richiesto, di comprare il biglietto.
- Località preferita: se l'utente è registrato ed utilizza un webBrowser l'utente può aggiungere e rimuovere le sue località preferite interagendo con una mappa.
- Metodo di pagamento: per un utente registrato è possibile definire e gestire le carte di credito o altri mezzi di pagamento dove far addebitare i vari pagamenti. Questo metodo permette anche la consultazione di tutte le transazioni per un determinato metodo di pagamento
- Comprare un biglietto: è previsto che l'utente registrato possa comprare un biglietto se la tratta desiderata prevede l'utilizzo di mezzi pubblici. A fronte di un pagamento viene inviata conferma dell'avvenuto pagamento tramite mail.

Design

Use case diagrams

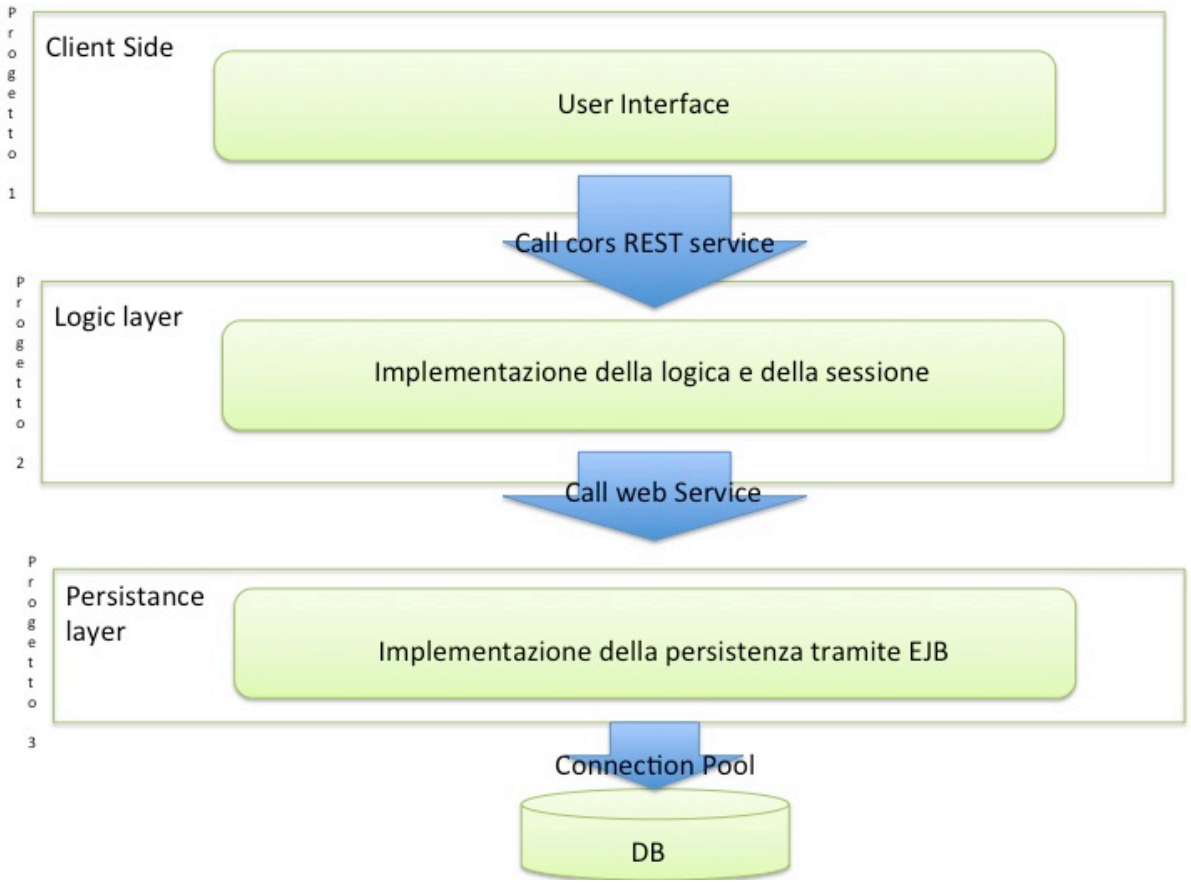


DB organization



Architectural considerations

Organization of the deployment on the various (logical) tiers



Used patterns

La comunicazione tra i vari progetti è basata interamente su un'architettura SOA.
Dato che la parte di User Interface è stata realizzata tramite jQuery e Backbone.js si è preferito utilizzare per la comunicazione tra i primi 2 progetti una architettura a servizi basata sul protocollo REST che trasporta messaggi in formato JSON. Le chiamate dal client verso il server utilizzano ajax in modalità asincrona e le risposte vengono identificate dalle opportune callback. Si è scelto un formato uniforme per i messaggi JSON di scambio con il server di front end.
La comunicazione tra il progetto 2 e 3 invece è stata basata sui servizi SOAP dato che la logica di interazione è omogenea per le due piattaforme (quella di front end e quella di back end).
Il progetto 3 fornisce uno stato di persistenza tramite l'utilizzo di EJB e di entità. La comunicazione con data base avviene tramite l'utilizzo di un connection pool e dell'EclipseLink.

Implementation and deployment

State management

Lo stato è mantenuto all'interno di una sessione nel secondo progetto (Server di front end) . In particolare lo stato relativo ad ogni utente connesso viene mantenuto tramite l'utilizzo di un EJB satefull. Il medesimo EJB contiene, oltre alle informazioni dell'unente, quelle necessarie per accedere ai servizi SOAP.
Dato che questo bean viene creato, gestito e mantenuto all'interno della servlet che gestisce il servizio REST, per salvare il bean si è scelto di generare un UUID identificativo di ogni utente e salvare tale bean nel contesto della servlet. Questo permette anche di creare una sessione differente per un utente che si collega contemporaneamente con 2 dispositivi separati.

Transactional behavior

Il comportamento del sistema risulta essere atomico in quanto le operazioni di merge, persist e remove nel EclipseLink sono per default eseguite in transazioni separate.

Technologies used

Le tecnologgie utilizzate nella parte client sono:
jQuery, Backbone.js e underscore.js per poter implementare in pattern a MVC basato interamente sul linguaggio Javascript e quindi poter essere integrato con Cordova-Phonegap e per il porting su iPad. In pratica la stessa code base viene utilizzata tramite un browser scaricandola da un server o viene gestita da Cordova-Phonegap come app residente sul dispositivo iPad (o iPhone o Android) dell'utente e quindi è potenzialmente distribuirli tramite AppStore o Andriod Market. Il codice javascript locale è stato adattato per poter gestire opportunamente i diversi gradi di risoluzione dei browsers e dei device portable e per poter gestire il cambio di orientamento (portrait-landscape).

La User Interface si compone di pagine statiche HTML, stili CSS, codice Javascript, e immagini. Il contenuto delle pagine HTML è popolato a runtime tramite codice javascript che utilizza le primitive della libraria jQuery. Oltre ad un framework di base (index.js) si è fatto uso della libreria Backbone.js e Underscore.js per gestire la user interface tramite un pattern di tipo MVC (Model View Controller) interamente governato sul client.

Il livello di Logica risulta essere un server Rest che interagisce con la parte di User Interface e contemporaneamente un client SOAP per poter accedere al DataBase. Tutto il sistema REST è basato sulle specifiche JAX-RS e sull'utilizzo delle annotazioni JAXB,

mentre la gestione dello stato risiede interamente su in EJB statefull. Il protocollo REST implementato aderisce allo stadard CORS (Cross Origin Resource Sharing) per permettere che le chiamate originate da pagine html scaricate da un server (o dal file system nel caso di app per device mobile) possano interagire in ajax con servizi REST residenti si altri server.
Il livello di persistenza è gestito tramite EJB stateless e l'utilizzo di classi di entità. I vari EJB poi possono eseguire operazioni sul DataBase tramite l'utilizzo di EclipseLink e un jdbc/MySQL basato su di un connection pool. Ad EclipseLink è stato impedito di mantenere una cache in modo tale che tutti i valori vengano scritti e letti direttamente dal DB(questo riduce le performance, ma permette una gestione più chiara del sistema nella fase di sviluppo e testing).
Il DataBase risiede interamente su una piattaforma MySQL gestite da mamp