Ingegneria del Software 2

POLITECNICO
DI MILANO

2013, Semestre invernale

TravelDream

Descrizione del progetto

Versione documento 1.0 (Dicembre 2013)

Dario Nesi # Leonardo Rossetti # Antonio Russo

Sommario

Introduzione

Scopo del documento

Viene descritto il progetto *TravelDream* nei termini utili alla comprensione delle dinamiche e delle strutture alla base del sistema in sviluppo. Viene riportata l'architettura del sistema, con la conseguente scomposizione in moduli e vengono mostrate le dipendenze interne. Il target di destinazione è un pubblico interno all'ambito di sviluppo del prodotto. Volendo essere più precisi, dato che si tratta di un progetto nato in ambito accademico, il target di riferimento è un pubblico composto da docenti e studenti del corso di *Ingegneria del Software 2* offerto dal *Politecnico di Milano* per il *Corso di Laurea Magistrale in Ingegneria Informatica (AA 2013/14)*.

Questo documento segue il documento di analisi dei requisiti (RASD) "TravelDream, Specifica dei requisiti"

Definizioni, acronimi e abbreviazioni

- *TravelDream*: applicativo web di e-commerce nato per supportare il processo di vendita dell'omonima agenzia di viaggi. E' il destinatario del documento e su di lui è concentrato tutto lo sforzo di analisi.
- *Prodotti Base* (abbr. PB): sono gli elementi base che compongono l'offerta dell'agenzia. Possono essere di tre tipologie: *biglietti per voli di linea*, *pernottamento in una struttura alberghiera/ostello* ed *escursioni*.
- *Pacchetti Viaggio* (abbr. PV): sono i prodotti che vengono realmente venduti dall'agenzia. Generalmente, sono composti da una combinazione di tre *PB*. Possono essere di tre tipologie: *pacchetti predefiniti*, *pacchetti personalizzati* e *"gift list"*. I primi sono i *PV* inseriti da parte dell'agenzia; i secondi sono i *PV* creati dagli utenti registrati al sistema secondo le loro richieste nel rispetto dei vincoli dello stesso; ed i terzi sono i *PV* che possono essere pagati da esterni nelle singole componenti.
- *Gift List* (abbr. GL): si tratta, come detto sopra, di *PV* speciali. La loro particolarità sta nel fatto che possono essere pagati da esterni nelle singole componenti.
- *Prodotto*: qualsiasi tipologia di *PV*.
- *Sistema*: l'applicativo web che stiamo progettando.

Contesto

Il progetto di *TravelDream* prevede la realizzazione di un applicativo web di e-commerce nato per supportare il processo di vendita dell'omonima agenzia di viaggi. Gli obiettivi che il sistema si pone sono riportati di seguito.

Codice	Descrizione
OB-1	Permettere, al pubblico, la consultazione dell'offerta di *TravelDream*.
OB-2	Permettere, agli *Impiegati* dell'agenzia, la gestione dell'offerta.
OB-3	Permettere la registrazione di un'utenza e la gestione del proprio profilo nel sistema.
OB-4	Permettere, ai soli *Utenti Registrati*, la personalizzazione di un *PV* offerto.
OB-5	Permettere, ai soli *Utenti Registrati*, la creazione di una *GL*.
OB-6	Permettere, ai soli *Utenti Registrati*, l'acquisto di un *PV predefinito* o *personalizzato*.
OB-7	Permettere, al soli *Utenti Registrati*, il pagamento di singole voci componenti una *GL*.
OB-8	Permettere, agli *Amministratori di Sistema*, la gestione dell'utenza che ha accesso all'applicativo e l'assegnazione dei loro privilegi.

Struttura del documento

Il presente documento vuole essere conforme allo standard IEEE Standard for Information Technology (1016-2009) Systems Design – Software Design Descriptions[1].

Si compone di due sezioni:

- Introduzione: qui si riprendono le nozioni base per la comprensione del contesto applicativo, già presentate nel documento "TravelDream - Specifica dei requisiti" e si illustrano le sigle che vengono usate internamente al documento.
- Descrizione del progetto: in questa sezione si descrive il progetto dai punti di vista principali necessari a comprendere quali sono le dinamiche del sistema, l'architettura e la scomposizione in moduli e le strutture dati che rappresentano il modello di dominio.

[1] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5167255

Descrizione del progetto

Parti interessate e requisiti progettuali

Si riporta la classificazione dell'utenza prevista del sistema emersa in fase di analisi dei requisiti. A seguire sono riportati i requisiti che il sistema deve soddisfare, suddivisi per utenza coinvolta.

Classificazione parti interessate

Amici (sottocategoria degli utenti non registrati): identificano l'utenza base che può avere accesso all'applicazione. Possiedono la sola facoltà di consultazione dell'offerta tramite catalogo online o ricerca.

Utenti Registrati: tutti coloro che han richiesto le credenziali di accesso al sistema per poter usufruire dei servizi di prenotazione e personalizzazione dei PV.

Impiegati: è il livello base di utenza. Ha facoltà di modificare l'offerta dell'agenzia al pubblico, ovvero, creare nuovi PV, disabilitare PV esistenti e modificare PV predefiniti o personalizzati dagli Utenti Registrati (clienti TravelDream). Tali utenze ricevono aggiornamenti dal sistema nel caso in cui delle procedure automatiche varino una nuova offerta. Inoltre, hanno una visione completa dell'elenco dei PV ed accesso alle statistiche del sistema.

Amministratore di sistema (unico): è il livello dai maggior privilegi, creato per poter gestire opportunamente le utenze. Difatti, è in grado di modificare la tipologia di una certa utenza o di inserirne una nuova rimanendo al di fuori delle logiche di sistema.

Requisiti per livello utenza *Amministratore di sistema*

Codice requisito	Descrizione	Use case collegati
REQU-21	Il sistema deve permettere agli utenti di livello Amministratore di sistema di modificare il livello di altri utenti.	CU-17
REQU-22	Il sistema deve permettere agli utenti di livello Amministratore di sistema di creare utenze di uno specifico livello.	CU-15
REQU-23	Il sistema deve permettere agli utenti di livello Amministratore di sistema di disabilitare l'accesso ad altri utenti.	CU-16

Requisiti per livello utenza *Impiegato*

Codice requisito	Descrizione	Use case collegati
REQU-16	Il sistema deve permettere di aggiungere un PV all'offerta dell'agenzia.	CU-2
REQU-17	Il sistema deve permettere di modificare un PV tra quelli offerti dall'agenzia.	CU-8
REQU-18	Il sistema deve permettere di disabilitare un PV tra quelli offerti dall'agenzia.	CU-9
REQU-19	Il sistema deve permettere l'aggiunta di un PB all'offerta dell'agenzia.	CU-13
REQU-20	Il sistema deve permettere di disabilitare l'offerta di un PB.	CU-14

Requisiti per livello utenza *Utente Registrato*

Codice requisito	Descrizione	Use case collegati
REQU-5	Il sistema deve permettere di aggiungere un prodotto base ad un PV.	CU-3, CU-4
REQU-6	Il sistema deve permettere di rimuovere un prodotto base da un PV.	CU-3, CU-4
REQU-7	Il sistema deve permettere di creare un PV come GL	CU-5
REQU-8	Il sistema deve permettere di pagare un PV predefinito	CU-10
REQU-9	Il sistema deve permettere di pagare un PV personalizzato	CU-4
REQU-10	Il sistema deve permettere di pagare una voce di una GL	CU-7
REQU-11	Il sistema deve permettere di condividere una GL con altri utenti tramite web-link.	CU-5
REQU-12	Il sistema deve permettere la composizione di PV personalizzato.	CU-3
REQU-13	Il sistema deve avvertire l'utente in caso di composizione di PV non corretta (*2).	-
REQU-14	Il sistema deve permettere di rimuovere la propria utenza dal sistema	CU-12

Requisiti per livello utenza *Amico*

Codice requisito	Descrizione	Use case collegati
REQU-1	Il sistema deve permettere la ricerca di PV per città.	CU-1
REQU-2	Il sistema deve permettere la ricerca di PV per nazione.	CU-1
REQU-4	Il sistema deve permettere la creazione di una utenza privata	CU-11
REQU-3	Il sistema deve permettere la consultazione del dettaglio di una GL	CU-6

Vista contestuale

Il diagramma riportato sotto fornisce informazioni su quali sono i servizi che il sistema fornisce all'utente, suddivisi in 10 macro-aree. Tale diagramma è orientato agli obiettivi.

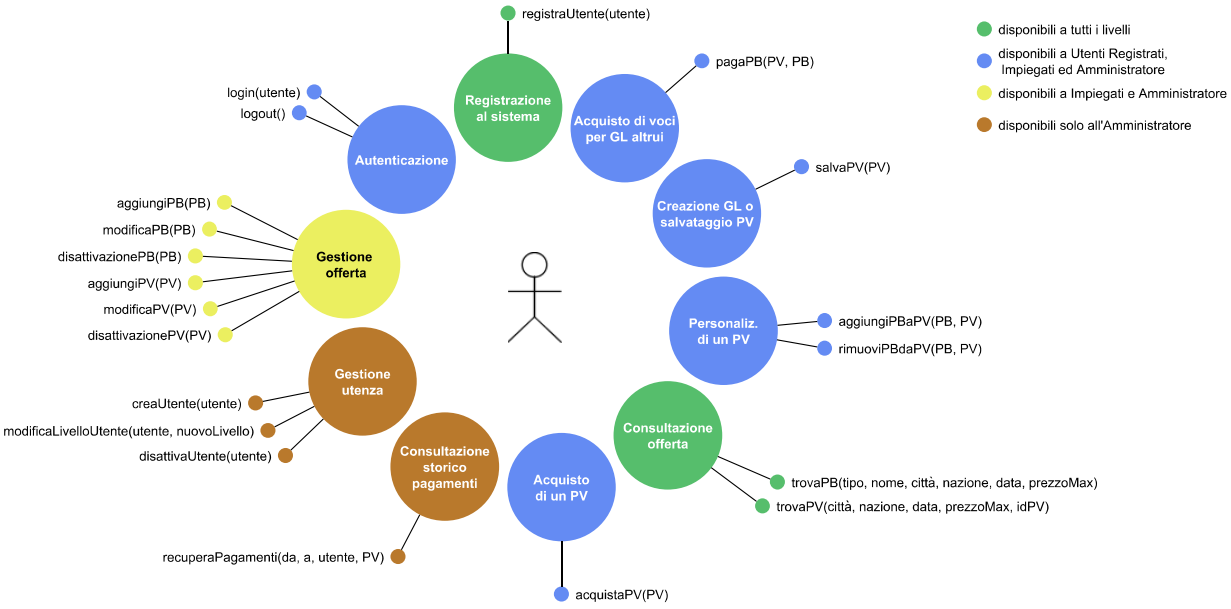

Diagramma 1 - Vista di contesto

Le aree ed i relativi servizi vengono riportati e descritti per maggiore chiarezza (i servizi che restituiscono Esito forniscono al chiamante informazioni sull'esito dell'operazione):

1. Consultazione offerta
    a. *trovaPV(città : String, nazione: String, data : Date, prezzoMax : currency, idPV : Integer) : PV []*: il sistema restituisce un elenco di PV (di descrittori di PV) filtrati in base ai parametri che sono avvalorati nella chiamata al servizio. Viene usato sia nelle ricerche che nel recupero delle informazioni di un singolo pacchetto.
    b. *trovaPB(tipo : TipoPB, nome : String, città : string, nazione : String, data : Date, prezzoMax : currency) : PB []*: il sistema restituisce un elenco di PB compatibili con i paramerti avvalorati nella chiamata al servizio.
2. Acquisto di un PV
    a. *acquistaPV(pv : PV) : Esito*: il sistema registra il pagamento di un PV.
3. Personalizzazione di un PV
    a. *aggiungiPBalPV(pb : PB, pv : PV) : Esito*: il sistema aggiunge un PB ad un PV esistente.
    b. *rimuoviPBdaPV(pb : PB, pv : PV) : Esito*: il sistema rimuove un PB da un PV esistente.
4. Creazione GL o salvataggio PV
    a. *salvaPV(pv : PV) : Esito*: il sistema registra il PV. Il tipo del PV (predefinito, personalizzato, GL) è contenuto nel descrittore del PV.
5. Acquisto di voci per i GL altrui
    a. *pagaPB(pv : PV, pb : PB) : Esito*: il sistema registra il pagamento di una sola voce appartenente ad un PV.
6. Registrazione al sistema
    a. *registraUtente(utente : Utente) : Esito*: il sistema registra una nuova utenza.
7. Autenticazione

   a. *login(utente : Utente) : Esito*: il sistema effettua il login di un utente.
   b. *logout() : Esito*: il sistema effettua il logout dell'utenza.
8. Gestione offerta
   a. *aggiungiPB(pb : PB) : Esito*: il sistema aggiunge un PB all'offerta.
   b. *modificaPB(pb : PB) : Esito*: il sistema modifica un PB dell'offerta.
   c. *disattivaPB(pb : PB) : Esito*: il sistema disattiva l'offerta di un PB.
   d. *aggiungiPV(pv : PV) : Esito*: il sistema aggiunge un PV all'offerta.
   e. *modificaPV(pv : PV) : Esito*: il sistema modifica un PV in offerta.
   f. *disattivaPV(pv : PV) : Esito*: il sistema disattiva l'offerta di un PV.
9. Gestione utenza
   a. *modificaLivelloUtente(utente : Utente, nuovoLivello : LivelloUtenza) : Esito*: il sistema modifica il livello di una utenza.
   b. *creaUtente(utente : Utente) : Esito* : il sistema registra una utenza.
   c. *disattivaUtente(utente : Utente) : Esito*: il sistema disattiva una utenza.
10. Consultazione storico pagamenti
   a. *recuperaPagamenti(da : Date, a : Date, pv : PV, utente : Utente)*: il sistema restituisce l'elenco dei pagamenti filtrato in base ai parametri avvalorati nella chiama del servizio.

Architettura del sistema

Il sistema da un punto di vista architetturale si compone di 5 livelli, raggruppate in tre unità architetturali:

- Sul browser web lato client:
  - il client-tier è composto di due parti fortemente coese, un set di pagine HTML ed una applicazione Javascript, la quale si occupa di fornire correttamente le funzionalità del sistema, di comunicare con il server e di gestire la visualizzazione dei frammenti HTML.
- Sul server applicativo:
  - Il web tier è composto dalle servlet SpringWebMVC che ricevono le richieste HTTP e le reindirizzano verso la logica di business implementata nello strato sottostante.
  - Il business logic tier è composto dall'insieme degli Enterprise Java Beans che implementano la logica di business del sistema. Essi manipolano il modello dati del sistema e restituiscono i risultati in base alle richieste delle servlet nel web tier.
  - Il persistence tier è composto dagli oggetti che rappresentano il modello dei dati e dagli oggetti che si occupano del recupero e dell'aggiornamento dei dati sul database.
- Sul database server:
  - Il database è realizzato da un solo schema SQL all'interno del quale trovano posto le entità e le relazioni che si applicano al modello di dominio del sistema.

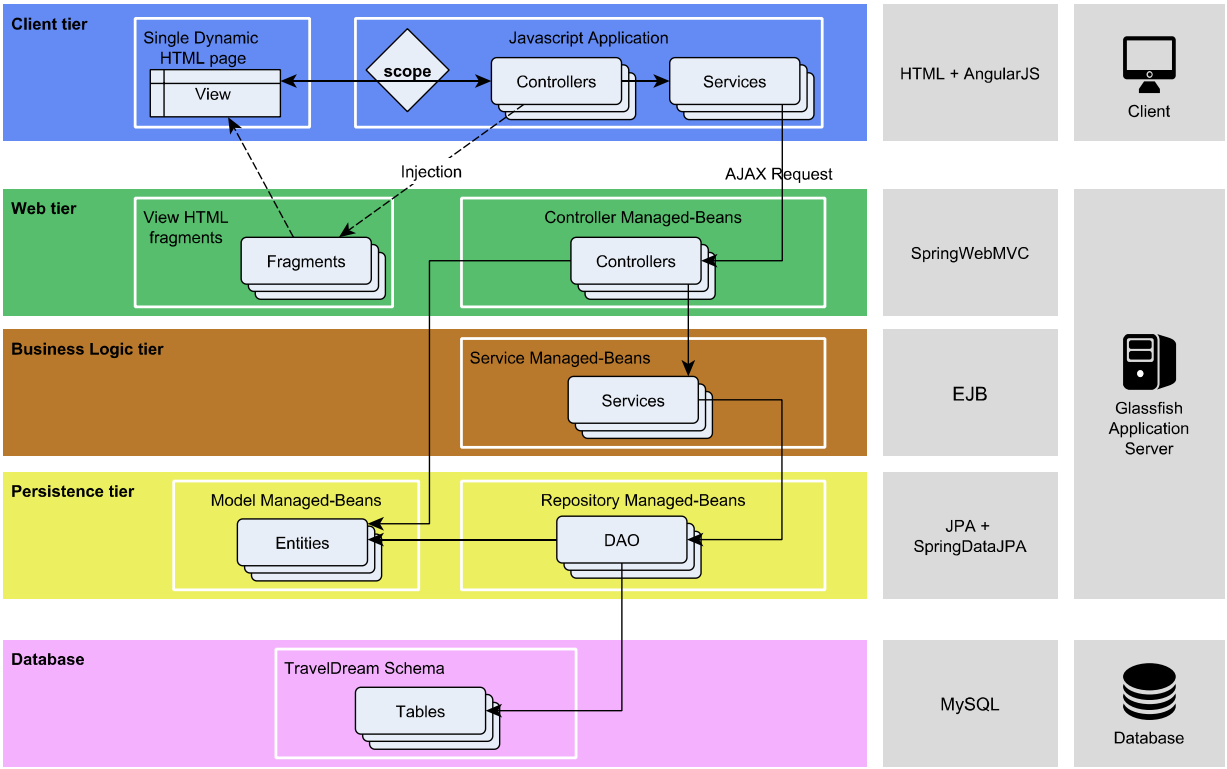

Diagramma 2 - Architettura del sistema

Struttura del sistema e dipendenze tra moduli

Il progetto è scomposto su tre livelli,

- client side, dove l'applicativo sfrutta le tecnologie AngularJS e HTML5
- server side, dove Spring3, EJB3 fanno da framework alla web-application in esecuzione su un server Glassfish 4
- db side, dove il DBMS mySql gestisce la base di dati relazionale

Nel livello client side si distinguono tre sottolivelli:

- View, che raggruppa tutte le viste dell'applicazione definite in linguaggio HTML5 e CSS3 e associa ad ogni elemento una routine per la gestione degli eventi definita nello strato dei controller
- AngularJS controller, dove sono definite le routine che hanno la responsabilità di rispondere agli eventi sollevati dalle viste. Ad ogni vista dell'applicazione è associato un controller.
- AngularJS service, che ospita i servizi che si occupano di comunicare col server per il recupero e l'invio dei dati. Tali servizi sono chiamati dai controller del livello superiore.

Lo strato server side si suddivide in quattro sottolivelli:

- SpringWeb Controller, dove sono collocate le servlet Java attivate da Glassfish al momento della richiesta di un particolare url. Tali servlet mappano i servizi definiti nello strato AngularJS service lato client.
- EJB Service, dove trova implementazione la logica di business, raggruppata per funzionalità, che risponde alle richieste effettuate dalle servlet. Per l'interfacciamento con lo strato della persistenza questo livello dialoga con il livello sottostante di JPA repository.

- JPA repository, definisce gli oggetti che hanno il compito di recuperare i dati dal database ed espone le funzionalità CRUD.
- Entity, dove sono definite le classi per la rappresentazione del modello dei dati, altrimenti dette Entity Beans.

Il diagramma con la scomposizione in moduli è riportato nella pagina successiva per chiarezza di lettura. Una versione PDF è recuperabile sul repository SVN del progetto.

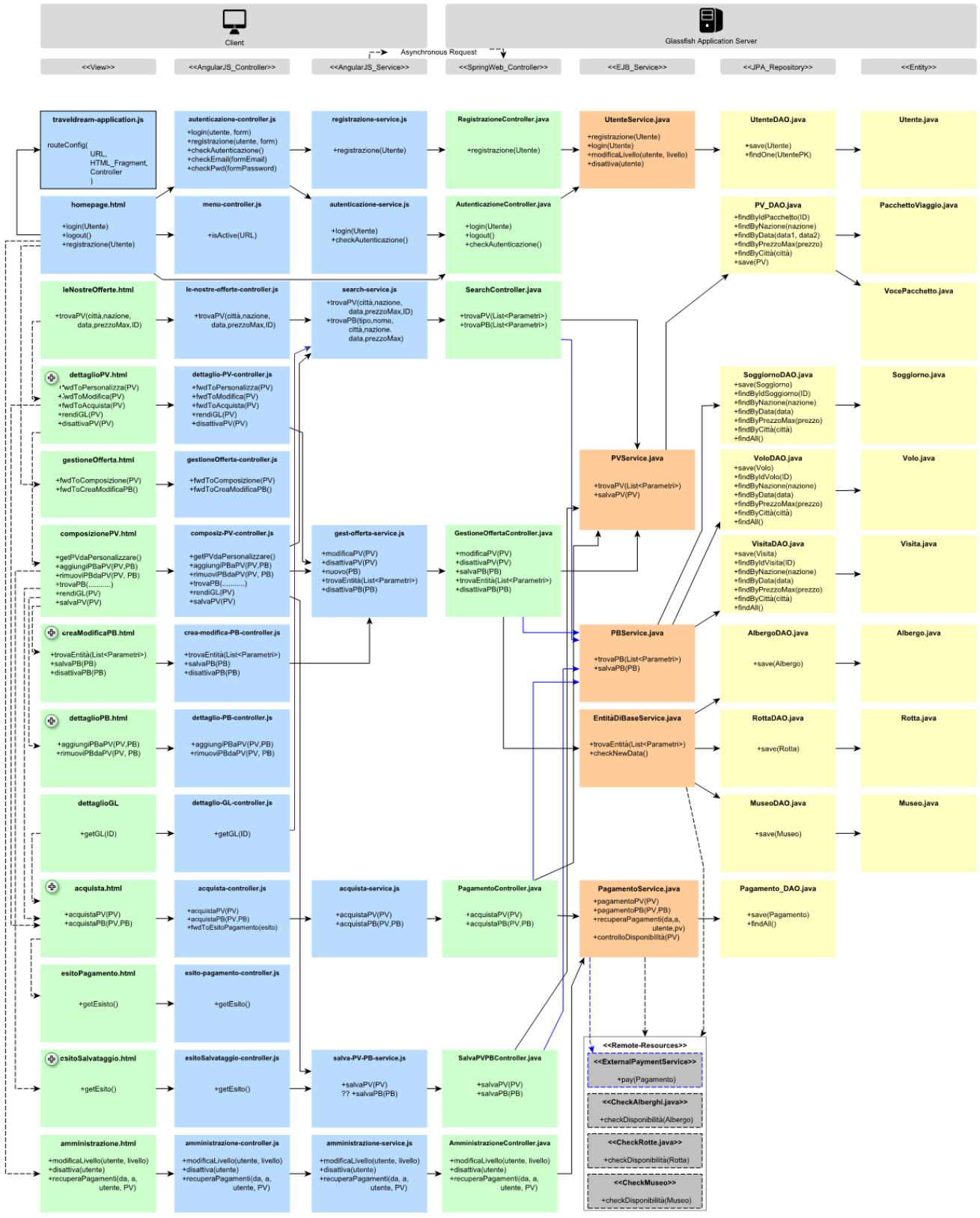

Diagramma 3 - Scomposizione del sistema in moduli e dipendenze

Homepage.html è l'unica pagina web dell'applicativo. E' direttamente controllata da TravelDream-App.js che inietta nel nodo etichettato *<ng-view>* i frammenti html (view) non modali, ogniqualvolta un url opportunamente mappato viene richiesto.

La funzione *isActive(url)* di *menu-controller.js* controlla in quale view l'utente sta navigando, in modo da rendere visibilmente attiva la corretta voce di menu.

La funzione *checkAutenticazione()* di *autenticazione-controller.js*, in fase di costruzione del DOM, controlla che l'utente che accede alla webApp sia già autenticato o meno: se la sessione sul server è attiva, in caso di aggiornamento dell'intera pagina, non viene richiesta una nuova autenticazione.

Le funzioni *fwdToView(object)* lato controller AngularJS mettono 'object' nel *rootScope* di TravelDream-App.js ed effettuano il forward alla 'view' attraverso l'url con la quale è mappata nell'applicazione Javascript. Sono utili nel caso in cui il forward a una specifica view debba avvenire solo in seguito alla ricezione di un esito proveniente dal Server a causa dell'evento che lo ha scatenato.

Interazione tra le componenti

Nel diagramma riportato sotto è illustrata una interazione esemplificativa tra tutti i moduli del sistema nella sequenza che vede l'utente creare un PV e salvarlo per poi condividerlo.

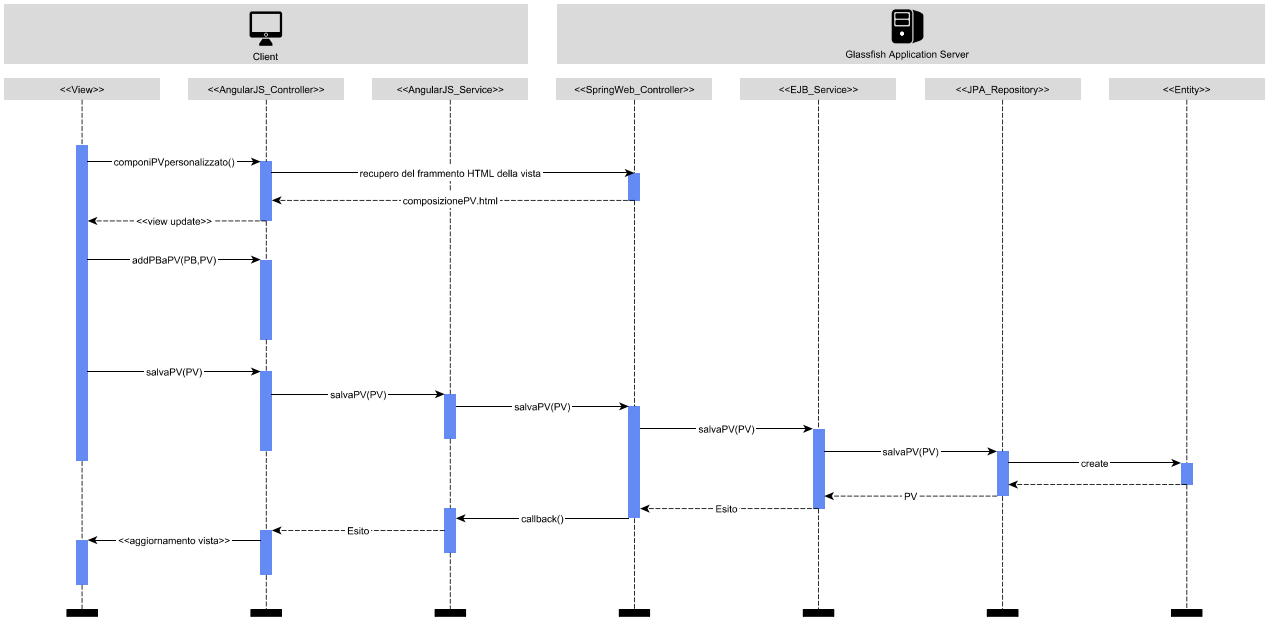

Diagramma 4 - Diagramma di sequenza

Modello dati

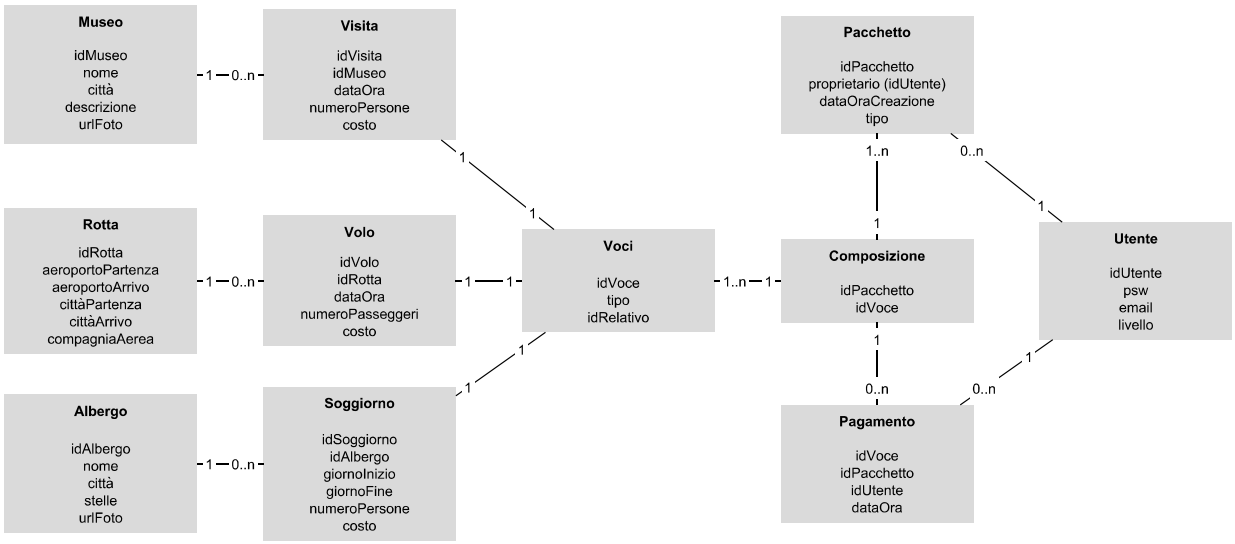

Diagramma 5 - Modello dati

Il modello dei dati è riportato nel diagramma sopra. Per popolare la tabella relative alle rotte vengono utilizzati i dati disponibili dal progetto OpenFlights[2] opportunamente elaborati per essere inserite nel DBMS mySql.

Pattern progettuali

Per la realizzazione del sistema si sfruttano principalmente due pattern progettuali la cui descrizione è riportata di seguito.

	ModelViewController
Contesto applicazione	Client tier, implementato attraverso l'uso del framework AngularJS.
Descrizione	Le pagine HTML si interfacciano con i controller AngularJS per la gestione degli eventi. I controller invocano dei servizi che vanno ad interagire col server e quindi col modello dei dati. I dati elaborati dai servizi vengono resi immediatamente disponibili alla vista tramite il binding bidirezionale di AngularJS tra View e Controller in cui il Model è contenuto nello strato intermedio denominato Scope.
Conseguenze	I componenti del client tier per funzionare correttamente devono avere conoscenza dei metodi e degli attributi dei componenti con i quali interagiscono. Eventuali lavori di manutenzione del software devono fare in modo di mantenere la coerenza tra i vari moduli nel caso l'intervento non riguardi tutto il client tier.

[2] http://openflights.org/data.html

	ModelController
Contesto applicazione	Application server, web tier, business logic tier e persistence tier.
Descrizione	Le servlet attivate in base alle richieste HTTP asincrone dialogano con i servizi implementati negli EJB e quindi col modello dati. Al contrario del MVC la parte di view viene gestita lato client, infatti la risposta all'invocazione sui sevizi sul server non è un frammento HTML "renderizzabile" ma una stringa JSON.
Conseguenze	Le servlet devono restituire una stringa JSON nel formato corretto nel caso di interventi di manutenzione dei soli applicativi lato sever.