TravelDream Management System

Design Document

Omar Maltoni, Enrico Ghirardi

Documento di Design in merito al progetto del corso di Ingegneria del Software 2 - A.A. 2013/2014
Docente di Riferimento: Mirandola Raffaella

Indice

1. Introduzione

1.1 Obiettivo del documento

Il presente testo rappresenta il documento di Design per il software TravelDream Management System (in seguito TDMS). La sua finalità è quella di presentare in modo chiaro e completo il sistema che verrà realizzato per soddisfare i requisiti e le funzionalità definiti dal precedente documento di specifica dei requisiti. A tale scopo faremo ampio utilizzo di diagrammi per illustrare in modo più schematico l'architettura del sistema.
Il documento è rivolto in particolar modo al team che avrà il ruolo di implementare, testare e manutenere il software.

1.2 Definizioni e acronimi

Per una migliore comprensione di questo documento è necessario fare riferimento al documento di specifica dei requisiti. Aggiungiamo alcune nuove definizioni che renderanno più chiara la lettura:

1.2.1 Definizioni

Parola	**Definizione**
Diagramma Entity-Relationship	Modello per la per la rappresentazione concettuale dei dati ad alto livello di astrazione
Model View Controller	Uno dei pattern più utilizzati per la progettazione software. Divide il programma in 3 diversi componenti: il Model che si occupa di gestire i dati dell'applicazione, il Controller che contiene la logica e fa da tramite tra le view e il model, la View che è l'interfaccia con cui si interagisce con il programma
Boundary Control Entity	Pattern molto simile ad MVC, utilizzato principalmente durante la fase di analisi.
Java Persistence API	Un'API di Java che si occupa della gestione dei dati relazionali
Framework	Insieme di librerie per semplificare lo sviluppo software
Java Server Faces	Framework Java per semplificare lo sviluppo delle interfacce utente tramite pagine web in applicazioni Java EE
Enterprise JavaBeans	Componenti software che implementano, lato server, la logica di business all'interno dell'architettura/piattaforma Java EE espletando servizi a favore della parte di front-end ovvero per la logica di presentazione di un'applicazione web.

1.2.2 Abbreviazioni

Sigla	Significato
TDMS	TravelDream Management System
JSF	Java Server Faces
JEE	Java Enterprise Edition
MVC	Model View Controller
BCE	Boundary Control Entity
DD	Design Document
JPA	Java Persistence API
EJB	Enterprise Java Bean

1.3 Riferimenti

1. TravelDream - Requirements Analisys and Specification Document: https://is2-travel-dream.googlecode.com/git/Deliveries/RASD%20-%20TravelDream.pdf

1.4 Struttura del documento

Come già detto in questo Design Document descriveremo la progettazione dell'architettura di TDMS.
Il presente testo si divide nelle seguenti quattro sezioni:

1. **Introduzione**: è la sezione introduttiva del documento, illustra lo scopo del documento, i riferimenti, le definizione e gli acronimi utilizzati, così da rendere più chiara la lettura;
2. **Design Architettura e Applicazione**: questa sezione da prima descriverà l'architettura in generale, partendo dai livelli da cui è composta. In seguito verranno illustrate le tecnologie che si sono decise di adottare e in che modo queste s'integreranno con il design scelto. Verrà inoltre fornito uno schema BCE che mostra in dettaglio il design delle singole componenti, seguendo il paradigma MVC;
3. **Design Base di Dati**: tutta l'analisi dei dati gestiti dall'applicazione sarà contenuta in questa sezione. Per primi saranno illustrati i passaggi che si sono seguiti per la costruzione del diagramma E-R. Successivamente verrà mostrata la traduzione dello stesso in uno schema logico adatto al database relazionale che sarà utilizzato da TDMS;
4. **Interazione con l'Utente**: qui verrà descritta l'interazione dell'utente con l'applicazione attraverso l'utilizzo di diagrammi che mostreranno gli Screen che la compongono;
5. **Appendice**: questa sezione raccoglierà informazioni utili come le note di versione, per seguire i cambiamenti applicati al documento durante le fasi successive del progetto.

2. Design Architettura e Applicazione

2.1 Descrizione Architettura

Il design dell'architettura software che si vuole realizzare ha l'obiettivo di rispettare le più utilizzate e consigliate pratiche. Lo stile che si è voluto usare è perciò un approccio multi-tier, molto comune nel caso di applicazioni enterprise complesse.
I principali tiers che la compongono sono i seguenti:

1. **Client Tier**
2. **Web Tier**
3. **Business Logic Tier**
4. **Persistence Tier**

Ognuno di questi livelli sarà implementato con tecnologie diverse.

2.1.1 Client Tier

Il livello più alto dove verrà eseguita l'applicazione, sarà infatti situato direttamente sulla macchina dell'utente, sia esso uno User, AU o un Admin.
Questo livello sarà composto dalle pagine WEB interpretate dal browser del client e restituite dal Web Tier a seguito di una richiesta HTTP.
L'applicazione è progettata per essere utilizzata principalmente da clienti che visitano il sito di TravelDream per prenotare un pacchetto viaggio. Essendo perciò l'utente comune il maggior utilizzatore, risulta necessario implementare un'interfaccia il più immediata possibile, senza lo sviluppo di Applet che potrebbero risultare scomodi sia nell'installazione che nella manutenzione.

2.1.2 Web Tier

Le pagine web con cui l'utente andrà ad interagire saranno prodotte tramite la tecnologia JSF, Java Server Faces, che permetterà agli sviluppatori di rispondere alle richieste dell'utente. Questo strato riceverà perciò delle richieste HTTP e genererà la risposta, comunicando, se necessario, con la logica business.

2.1.3 Business Logic Tier

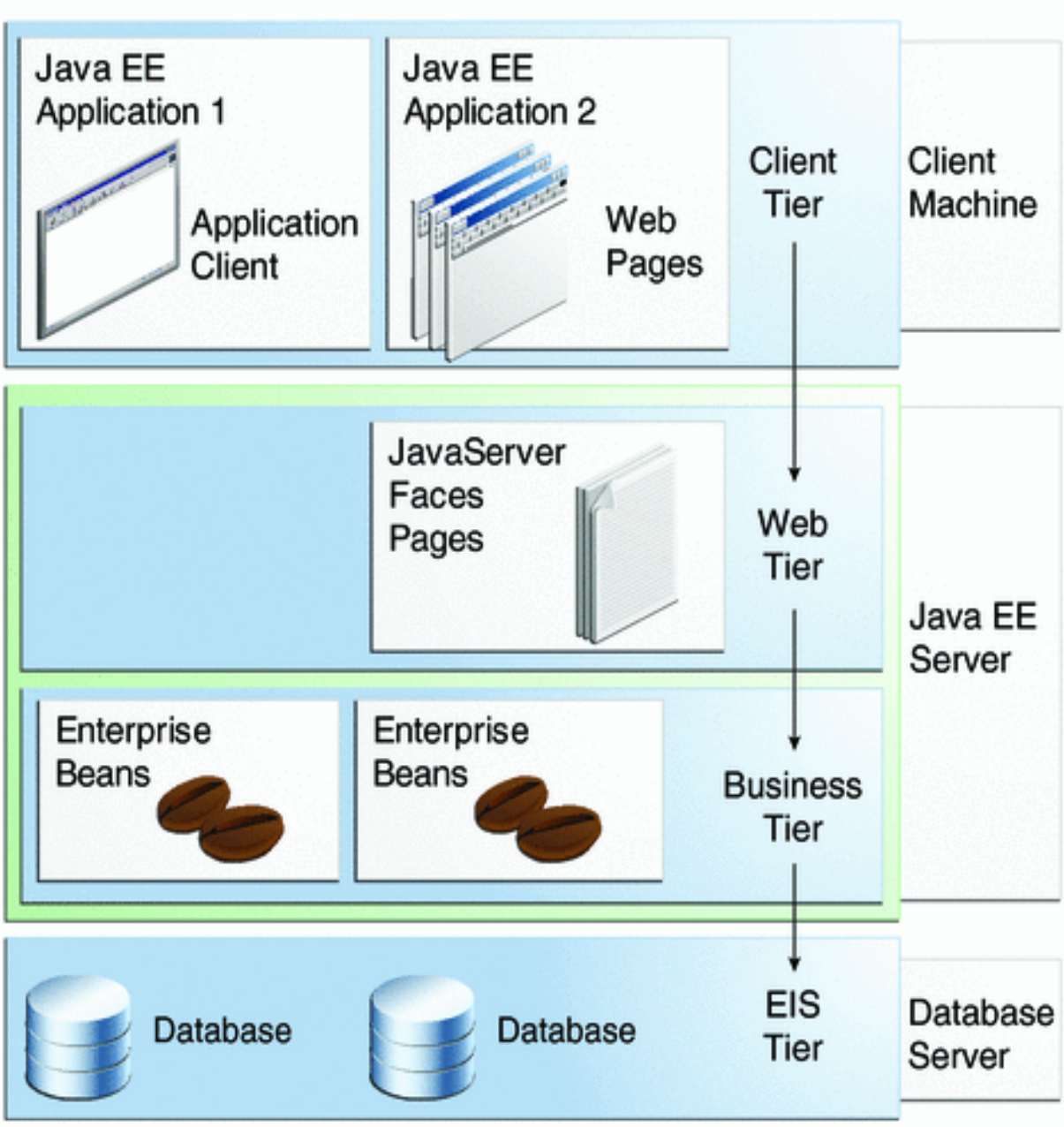
La logica business, quindi tutta la logica applicativa, si troverà in questo strato e verrà realizzata tramite Enterprise Java Beans, in particolare tramite Session Beans.

2.1.4 Persistence Tier

Per ultime invece sono definite le Entity, implementate con dei Persistence Beans. Questi forniranno una rappresentazione logica delle entità del sistema da mappare sul Database; questo livello sarà infatti in diretta comunicazione con un DBMS relazionale. La comunicazione con il DB sarà possibile tramite le Java Persistence API, implementate da Hybernate 2.1, che ci forniranno, oltretutto, la funzionalità di Object Relational Mapping.

2.1.5 Schema Architettura

Il primo tier, come già detto, sarà eseguito direttamente sulla macchina dell'utente.
I restanti tre invece saranno eseguiti dal Java EE Application Server scelto (Glassfish 4.0).
Quest'ultimo andrà ad interfacciarsi con la base di dati su cui verranno mappate le Entity, definite al livello superiore. Come database si è deciso di utilizzare MySQL.



Schema standard architettura applicazione Java EE

2.2 Design Applicazione

Come definito nelle precedenti sezioni, l'architettura che si andrà a realizzare sarà composta principalmente da 4 livelli. Poiché però abbiamo deciso di non realizzare un'applicazione lato Client, lo sviluppo del software verterà sui tier che andranno eseguiti sull'AS.

2.2.1 Java Packages

Risulta naturale dividere il progetto in 3 packages corrispondenti ai livelli da implementare:

- **Client package**: questo package contiene l'implementazione dell'interfaccia utente ed è perciò responsabile dell'interazione con essi. Esso dovrà quindi interagire con la business logic in modo da svolgere le funzioni richieste dall'utente e presentare il risultato.

- **Business Logic package**: questo package contiene la logica vera e propria dell'applicazione. Questo package ha il compito di interpretare le richieste proveniente dal client package e fornire le risposte interfacciandosi con il package della persistenza se necessario

- **Persistence package**: questo package implementa le entity che sono state definite per rappresentare il sistema. Dovrà interfacciarsi con il business package e rispondere alle richieste di questo.

Tutti gli utenti di TDMS, sia User, Advanced User, Administrator (ma anche i guest) si interfacceranno unicamente con il client package, al quale faranno richiesta per accedere a tutte le funzionalità offerte dal sistema.

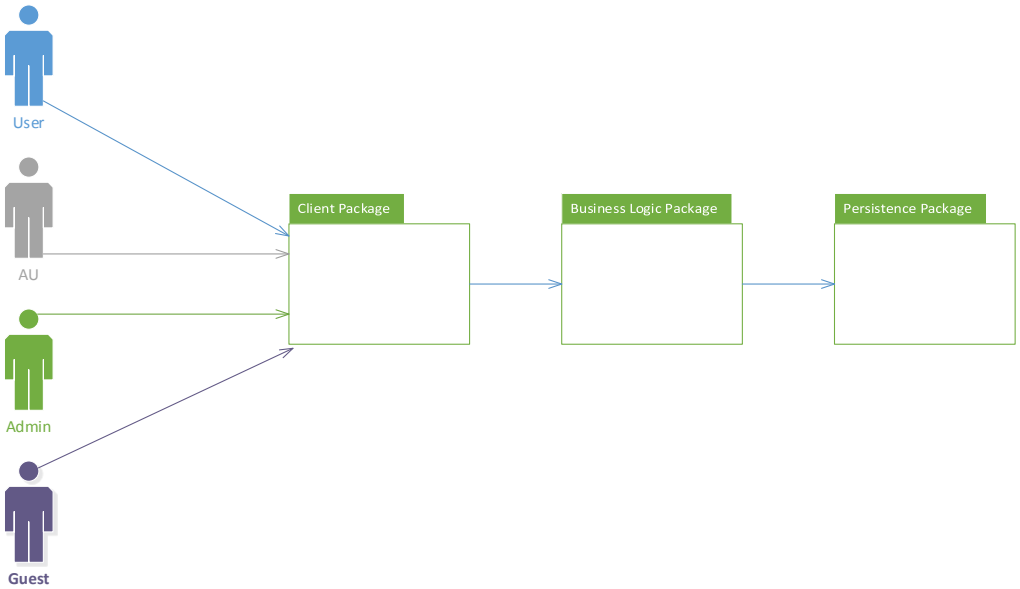
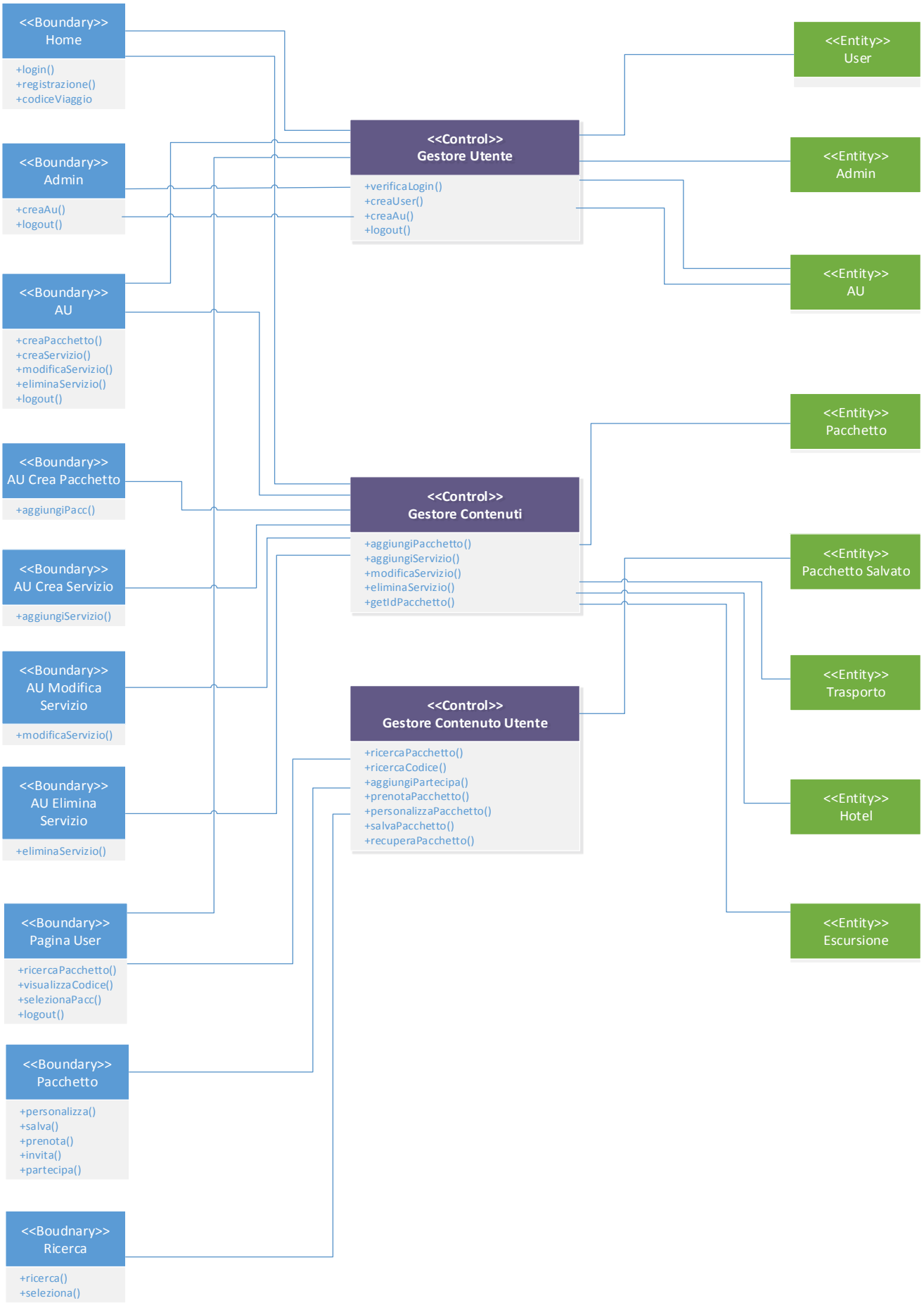


Diagramma dei packages

2.2.2 Componenti Applicazione

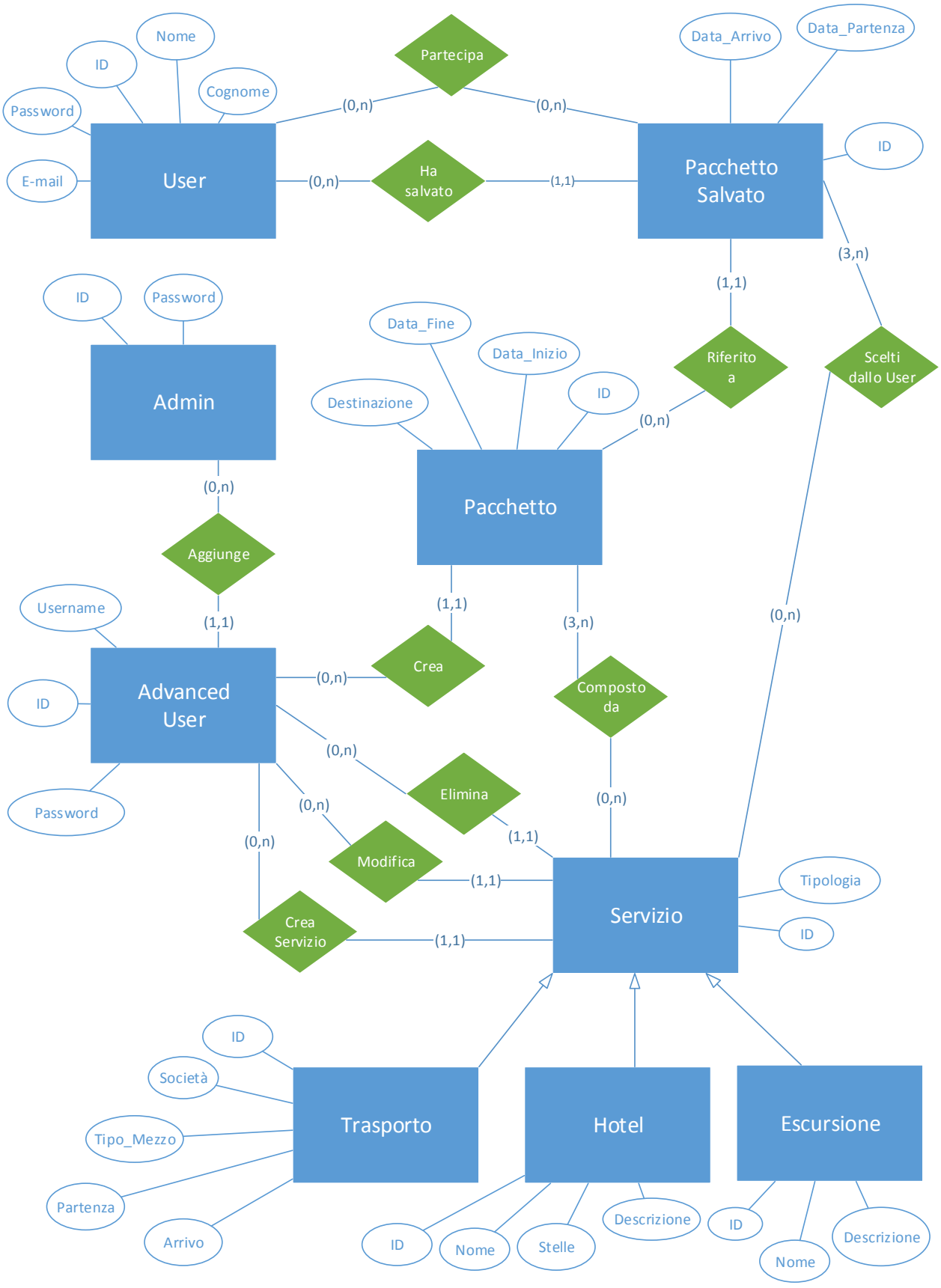
Di seguito è illustrato, tramite uno schema di tipo Boundary Control Entity, il design dei componenti che andranno a comporre l'applicazione:


<<Boundary>>
Home
+login()
+registrazione()
+codiceViaggio
<<Boundary>>
Admin
+creaAu()
+logout()
<<Boundary>>
AU
+creaPacchetto()
+creaServizio()
+modificaServizio()
+eliminaServizio()
+logout()
<<Boundary>>
AU Crea Pacchetto
+aggiungiPacc()
<<Boundary>>
AU Crea Servizio
+aggiungiServizio()
<<Boundary>>
AU Modifica Servizio
+modificaServizio()
<<Boundary>>
AU Elimina Servizio
+eliminaServizio()
<<Boundary>>
Pagina User
+ricercaPacchetto()
+visualizzaCodice()
+selezionaPacc()
+logout()
<<Boundary>>
Pacchetto
+personalizza()
+salva()
+prenota()
+invita()
+partecipa()
<<Boudnary>>
Ricerca
+ricerca()
+seleziona()
<<Control>>
Gestore Utente
+verificaLogin()
+creaUser()
+creaAu()
+logout()
<<Control>>
Gestore Contenuti
+aggiungiPacchetto()
+aggiungiServizio()
+modificaServizio()
+eliminaServizio()
+getIdPacchetto()
<<Control>>
Gestore Contenuto Utente
+ricercaPacchetto()
+ricercaCodice()
+aggiungiPartecipa()
+prenotaPacchetto()
+personalizzaPacchetto()
+salvaPacchetto()
+recuperaPacchetto()
<<Entity>>
User
<<Entity>>
Admin
<<Entity>>
AU
<<Entity>>
Pacchetto
<<Entity>>
Pacchetto Salvato
<<Entity>>
Trasporto
<<Entity>>
Hotel
<<Entity>>
Escursione

3. Design Base di Dati

3.1 Modello Concettuale

Durante il processo di astrazione che si è seguito per determinare le entità riportate nel diagramma E-R della pagina seguente, ci si è riferiti al diagramma delle classi definito nel RASD. Le entità principali così individuate sono: User, AU, Admin, Pacchetto, PacchettoPersonalizzato ed infine Servizio.
Le prime tre entità corrispondono agli attori del sistema, cioè quelle che interagiranno in maniera attiva con l'applicativo.
User identifica il cliente che si collega al sito web di TravelDream per comprare e personalizzare i pacchetti viaggio. Occorrerà dunque salvare i suoi dati personali, un indirizzo e-mail ed una password. Inoltre sono necessarie anche due relazioni che legano lo User ai pacchetti viaggio da lui salvati e/o personalizzati e ai pacchetti per cui ha confermato la partecipazione, su invito di un altro utente. Non abbiamo ritenuto necessario salvare una grande quantità di dati personali poiché non è prevista l'implementazione di un sistema di pagamento.
Advanced User, invece, avrà solo gli attributi username e password, e sarà creato dall'entità Admin. Queste due entità svolgono i ruoli di gestione dei contenuti e di gestione degli account dipendenti. L'AU, infatti avrà la possibilità di creare pacchetti viaggio e aggiungere, eliminare e modificare i servizi che li compongono.
Servizio è una delle entità più importanti del sistema. Infatti è necessaria per la costruzione dei pacchetti, siano questi personalizzati o quelli predefiniti creati dagli AU. Da Servizio derivano tre sotto entità: Trasporto, Hotel ed Escursione.
Trasporto identifica il mezzo di trasporto per il viaggio. Apparterrà ad una tipologia ed avrà un riferimento alla società che lo fornisce oltre alle località di partenza e arrivo.
Hotel avrà gli attributi nome, descrizione e stelle, necessari per identificare l'hotel ed indicare i servizi da esso offerti e la loro qualità.
Escursione avrà semplicemente una descrizione dell'attività offerta al cliente. Non è stato previsto nessun attributo che indichi la disponibilità di posti in quanto non è prevista la gestione dei limiti di partecipazione ai viaggi.
Questi servizi andranno a comporre il Pacchetto viaggio creato dagli AU che perciò saranno legati all'entità AU tramite opportune relazioni. Inoltre vogliamo mantenere salvate le date di validità del pacchetto, in modo da rendere la ricerca da parte dell'utente più semplice.
Gli User potranno personalizzare questi pacchetti di base selezionando alcuni servizi fra quelli proposti. Sarà necessaria perciò l'entità PacchettoPersonalizzato, in cui avremo un riferimento al pacchetto base da cui è stato creato, le date di partenza e ritorno scelte dall'utente e relazioni con i servizi personalizzati selezionati.



Schema modello E-R

3.2 Traduzione al modello logico

Il passo successivo comporta la traduzione del diagramma E-R in uno schema logico che sia possibile "mappare" sulla base di dati. Per far ciò è necessario progettare come rappresentare, in un database relazionale, le diverse relazioni tra le entità cercando di avere meno ridondanza possibile, mantenere l'espressività e con la massima chiarezza possibile.
Seguono le spiegazioni dei punti critici riscontrati durante questo processo:

- Come già detto lo User è legato ai pacchetti da lui creati e a quelli per cui ha confermato la partecipazione. Entrambe sono relazioni 1-n che si è deciso di rappresentare diversamente nella base di dati. Infatti, poiché il legame tra il creatore e il pacchetto è piuttosto forte, si è preferito inserire un riferimento all'id dell'utente creatore direttamente fra gli attributi del PacchettoSalvato. La relazione dei partecipanti andrà invece ad occupare una tabella separata cosicché, nel caso si decida in futuro di implementare la possibilità rimuovere la propria partecipazione, non si vada ad intaccare l'entità che descrive il pacchetto stesso.

- Il secondo problema da affrontare è la rappresentazione dell'entità padre Servizio e delle sue sotto-entità. Se dapprima era sembrato conveniente mantenere un'unica entità di riferimento Servizio in modo da avere un unico identificativo e un'unica tabella che inglobava anche quelle delle entità figlie, per il design attuale il Servizio è un'entità troppo "piccola" e superflua per rimanere a sé stante. Si è perciò deciso di inglobarla nelle diverse entità figlie (Trasporto, Hotel, Escursioni).

- Le relazioni rispettive fra gli AU e i servizi/pacchetti e quella tra Admin ed AU non sono state rappresentate poiché sono a livello applicativo non è di alcun interesse il mantenimento di dati su chi ha inserito un particolare servizio nel pacchetto o ha creato quest'ultimo.

- Il resto della traduzione è stato fatto in maniera sistematica per mantenere il design il più semplice possibile. Le relazioni mancanti, cioè quelle che legano i Pacchetti e i PacchettiPersonalizzati ai servizi, sono state tutte tradotte con tabelle dedicate.

Applicando questi passaggi lo schema logico che si ottiene è il seguente:

Users(id_user, email, password, nome, cognome, data_registrazione)

UsersGroups(email, group_name)

Pacchetti(id_pacchetto, nome, descrizione, localita, inizioValidita, fineValidita)

PacchettiSalvati(id_pacchettoSalvato, id_userCreatore, id_pacchettoOriginale, dataPartenza, dataRitorno, prenotato)

Trasporti(id_trasporto, tipologiaTrasporto, societa, localitaPartenza, localitaArrivo)

Hotels(id_hotel, nome, descrizione, stelle)

Escursioni(id_escursione, nome, descrizione)

PartecipantiPacchettiPersonali(id_pacchettoSalvato, id_userPartecipante)

HotelsPacchettoPersonale(id_pacchettoSalvato, id_hotel)

EscursioniPacchettoPersonale(id_pacchettoSalvato, id_escursione)

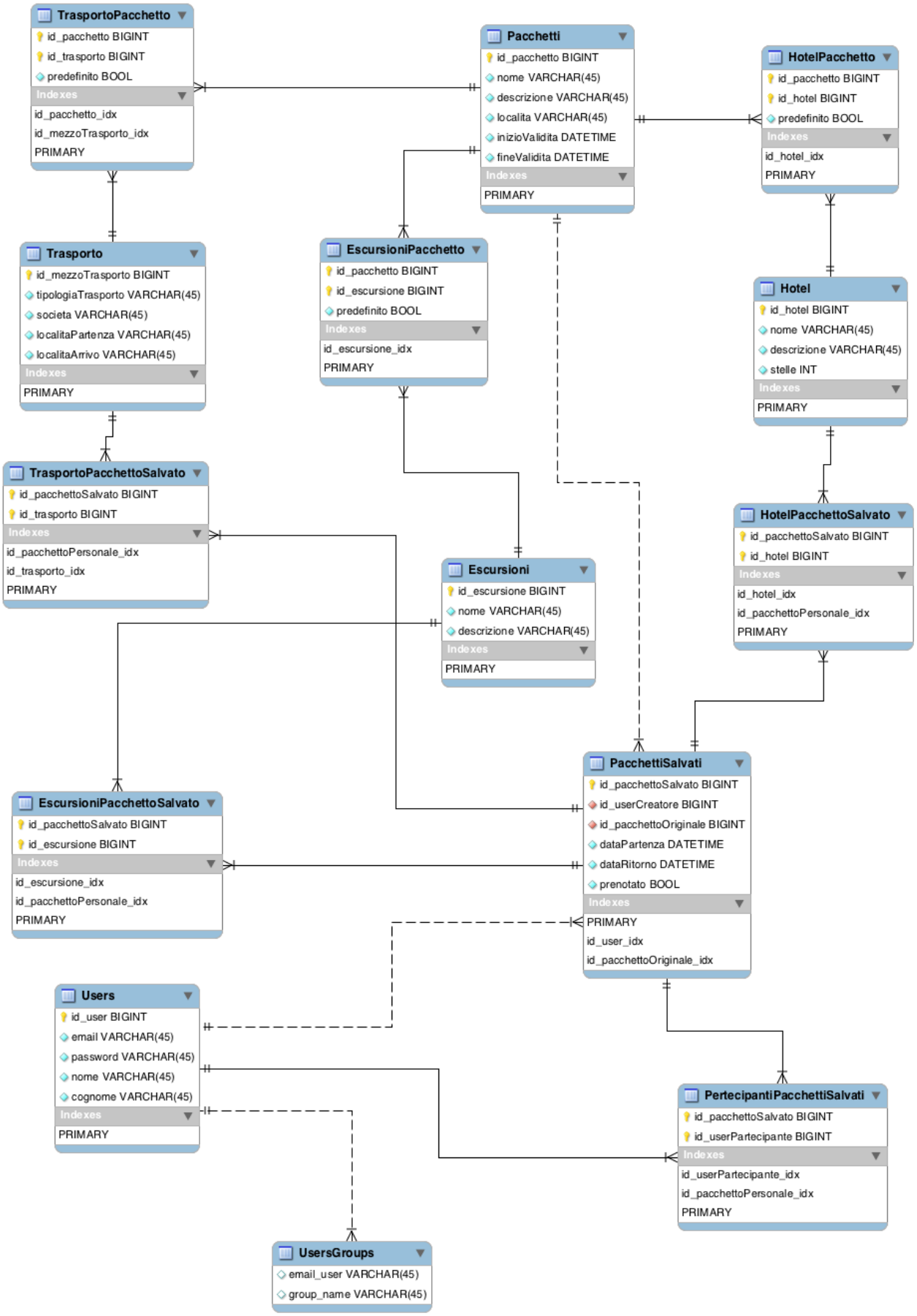
TrasportiPacchettoPersonale(id_pacchettoSalvato, id_trasporto)

HotelsPacchetto(id_pacchetto, id_hotel, predefinito)

EscrusioniPacchetto(id_pacchetto, id_escrusione, predefinito)

TrasportiPacchetto(id_pacchetto, id_trasporto, predefinito)

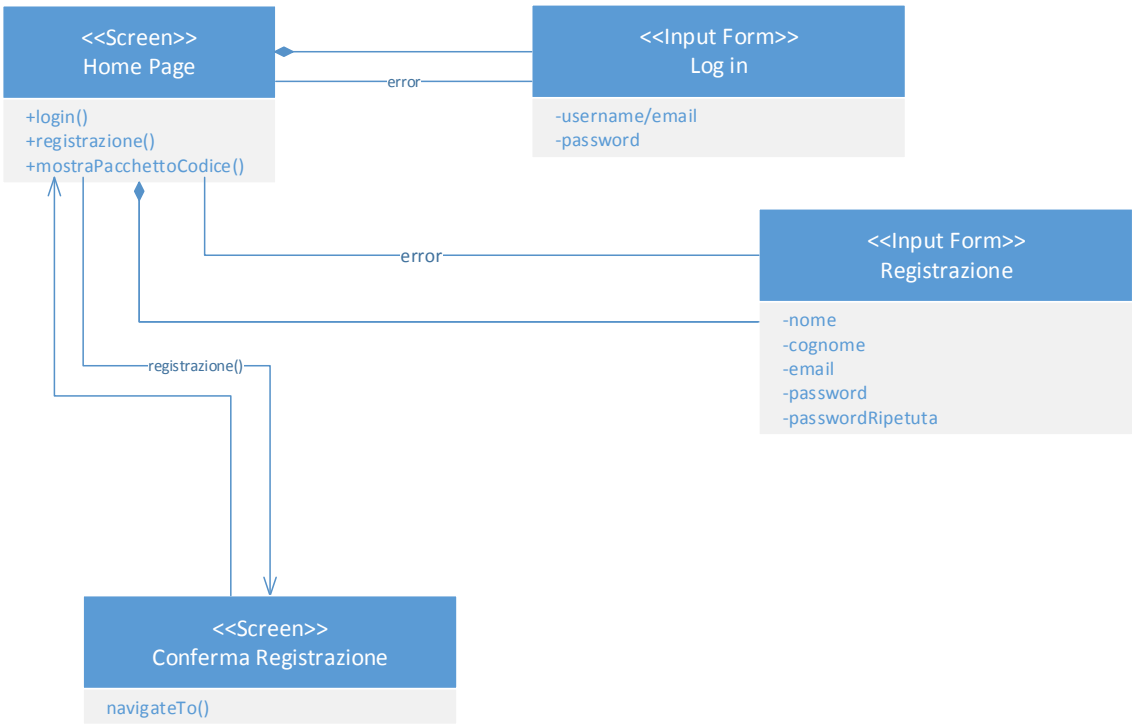
Segue un diagramma realizzato con MySQL Workbench che dovrebbe rendere più chiare le relazioni fra le chiavi utilizzate per le tabelle:



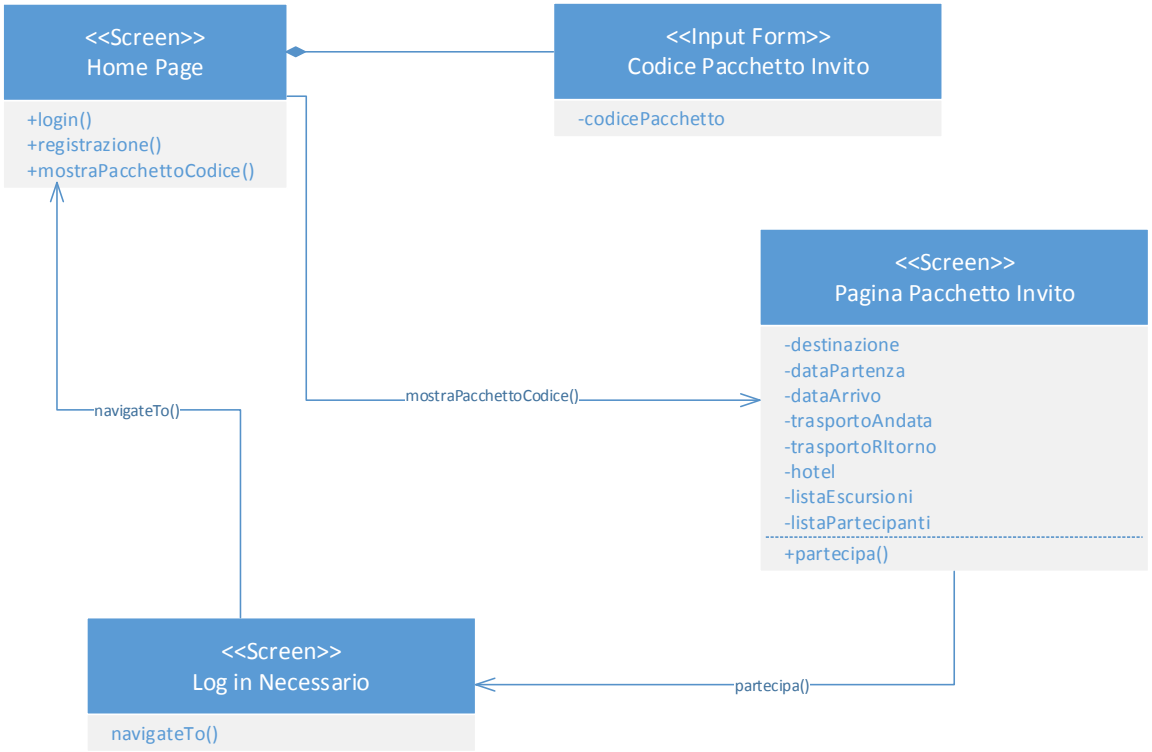
Rappresentazione schema logico ricavato dalla traduzione

4. Interazione con l'Utente

Di seguito sono riportati una serie di diagrammi che illustrano chiaramente l'interazione degli utenti con l'applicazione.

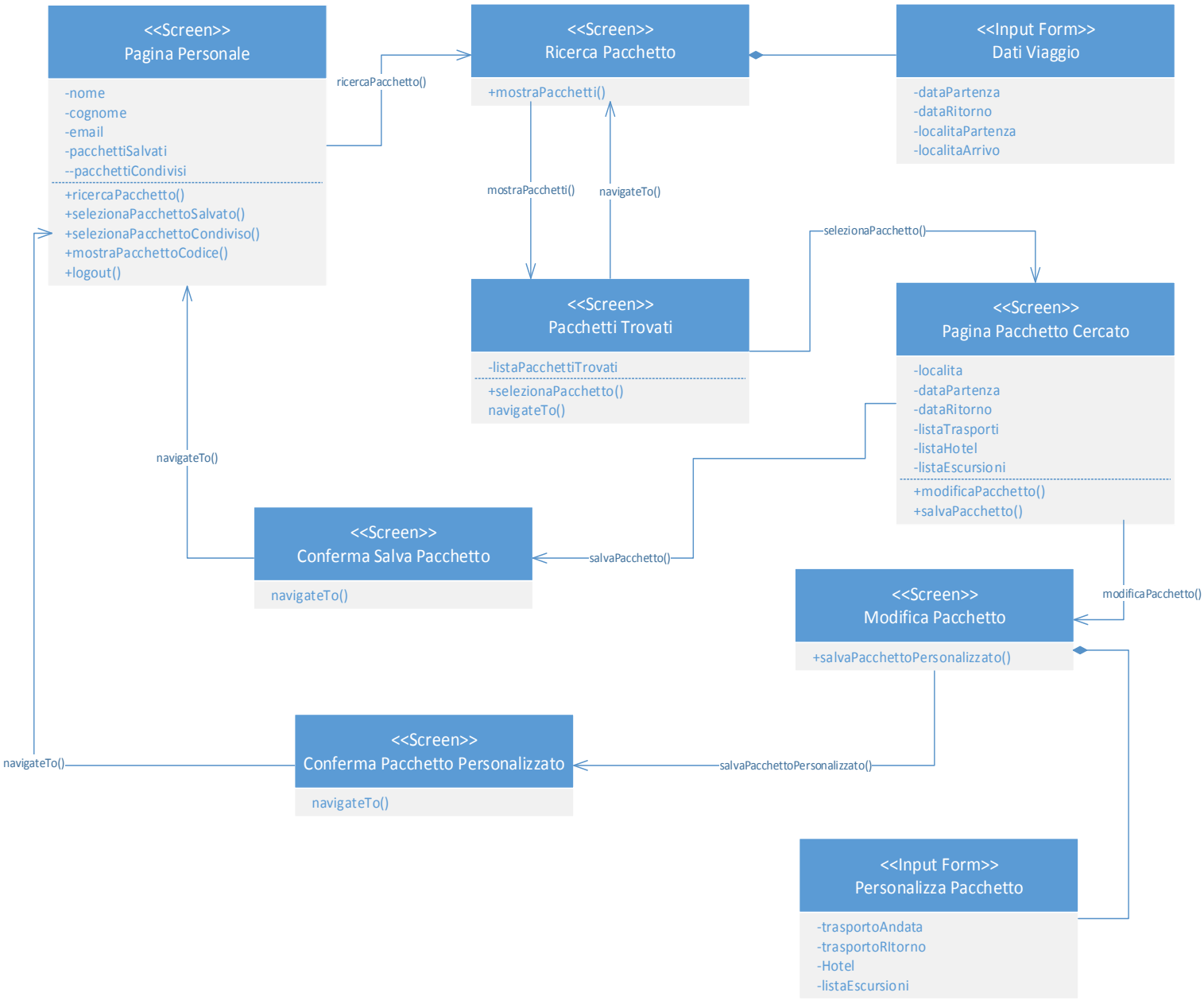


Registrazione Guest

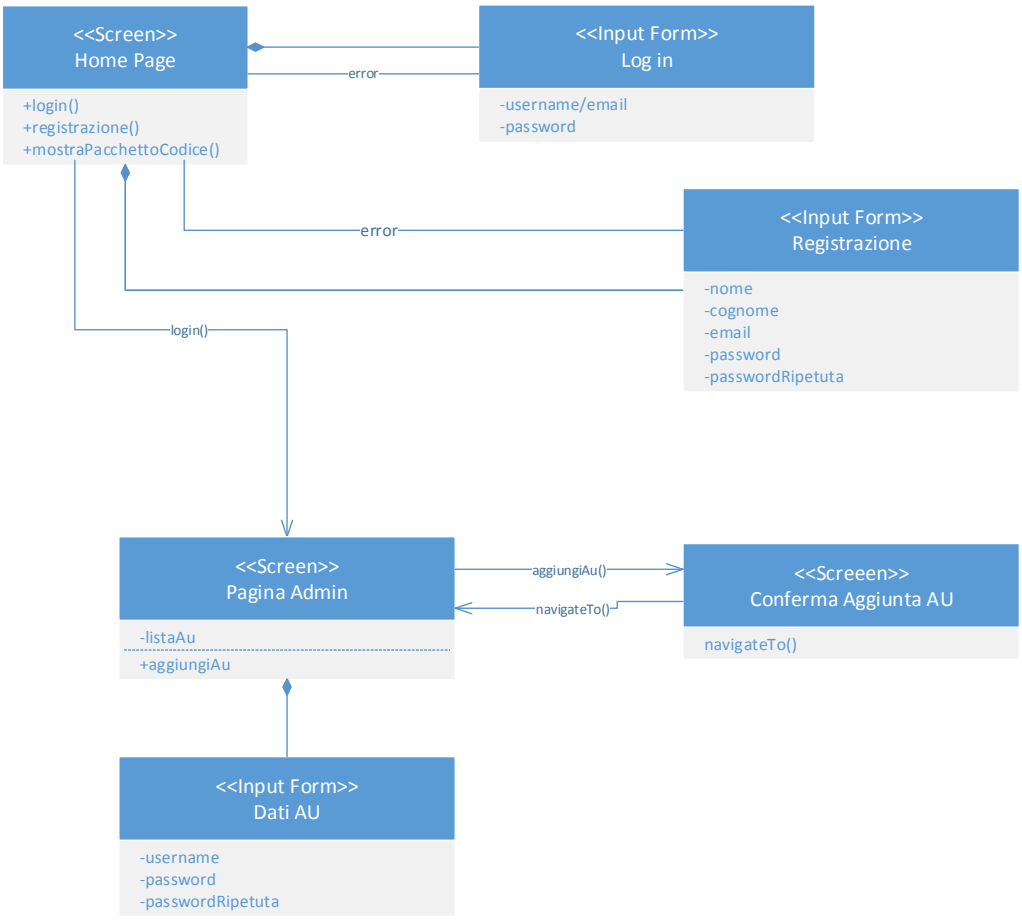


Guest Visualizza Pacchetto per Codice

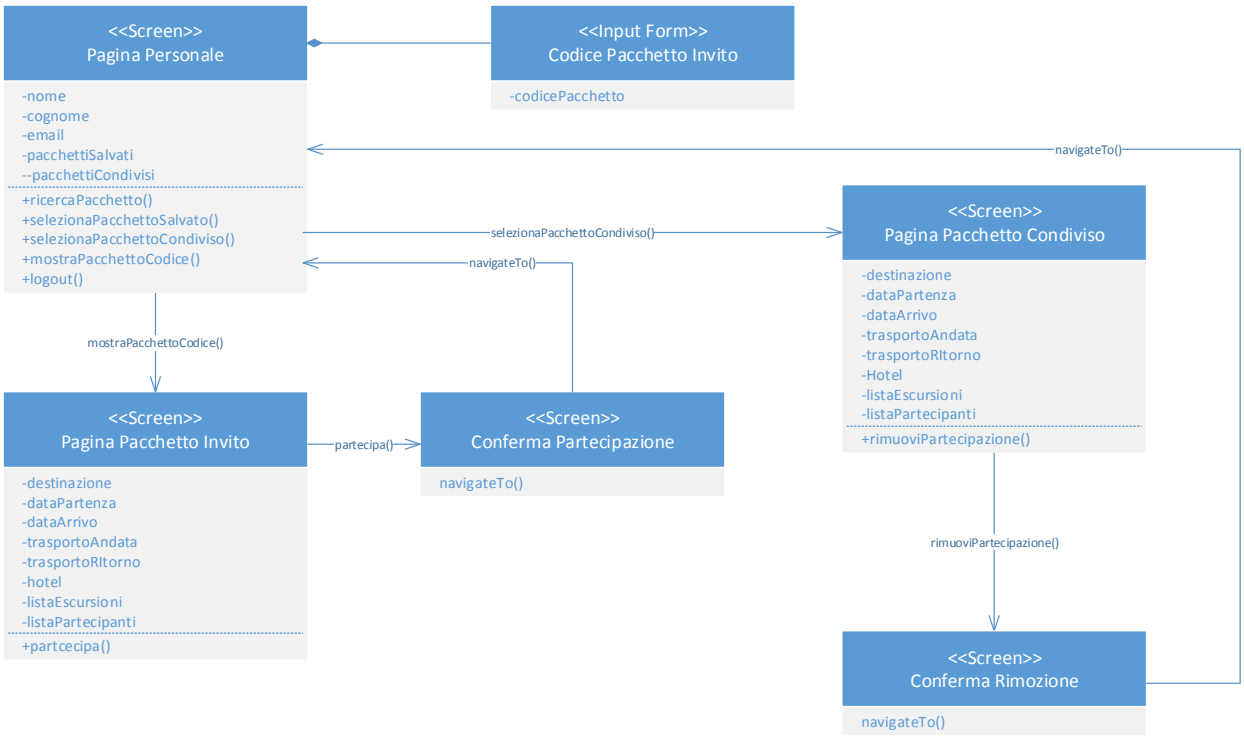
User modifica un pacchetto, invita degli amici e prenota



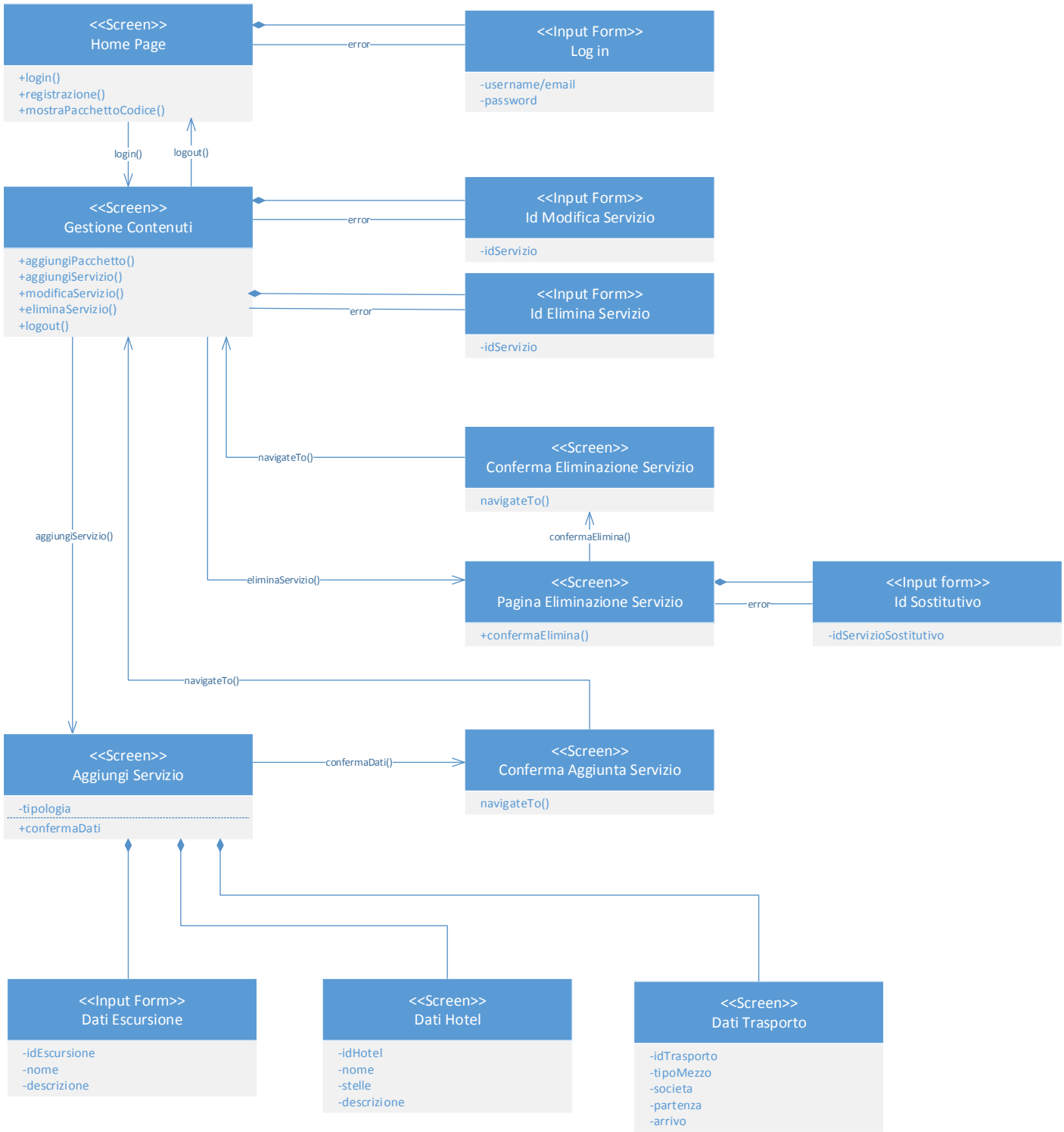
User ricerca un pacchetto, personalizzazione e salvataggio



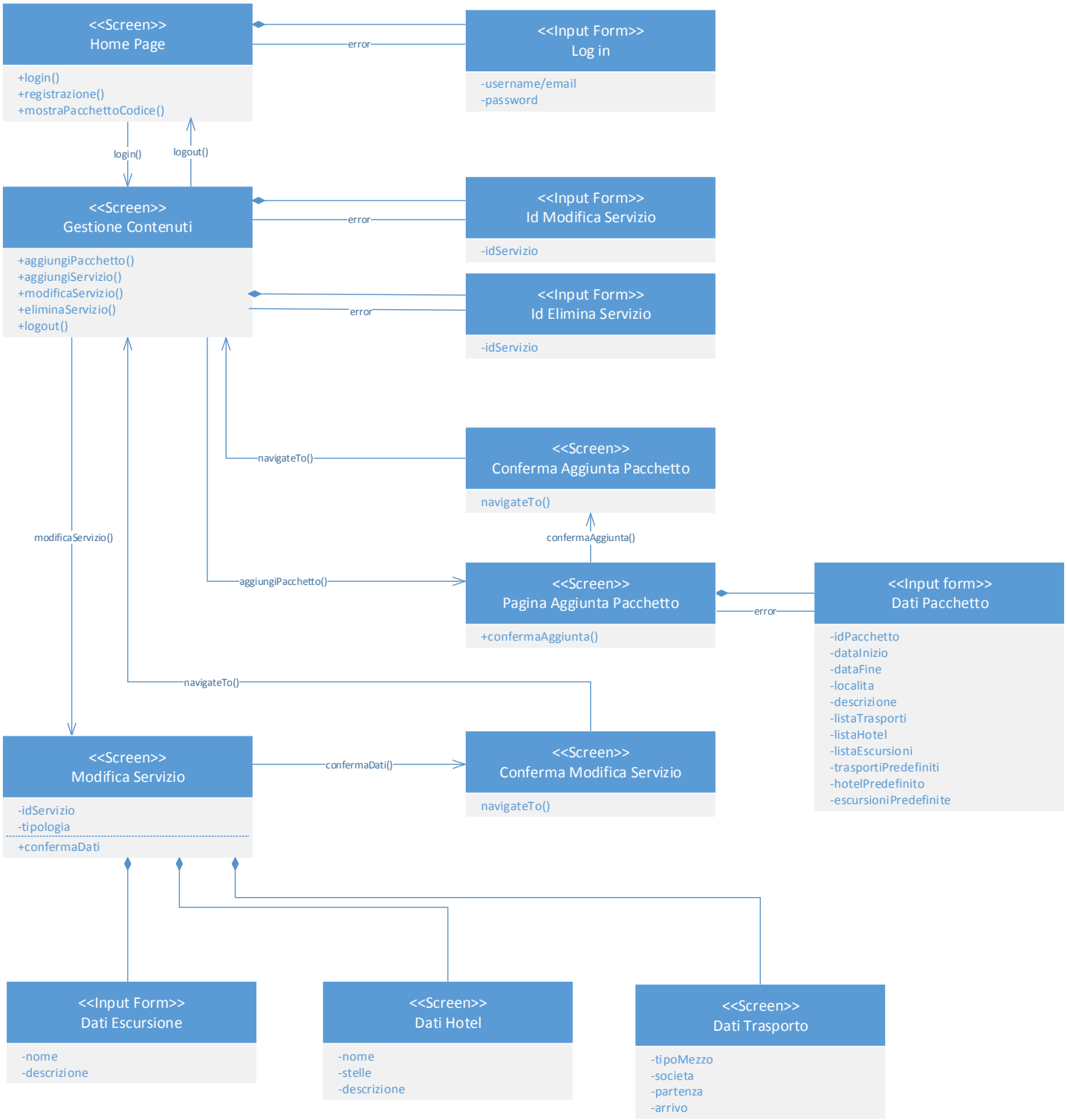
Admin aggiunge Advanced Users



User visualizza e partecipa ad un pacchetto tramite invito



Advanced User elimina ed aggiunge servizi



Advanced User modifica un servizio e crea un nuovo pacchetto

5. Appendici

5.1 Note di versione

Versione	Note
1.0	Il tempo dedicato sia alla preparazione, che alla stesura della prima versione del presente documento è superiore alle 40 ore, suddivise in percentuale uguale tra i due autori Omar Maltoni ed Enrico Ghirardi
1.1	Sono state effettuate delle piccole modifiche ad alcuni paragrafi e sono stati aggiornati lo schema BCE e lo schema logico.