Progetto di Ingegneria del Software 2

Travel Dream

Design Document

A.A. 2013/2014

Autori:
Gabriele Rovaris (759043), Rachele Sabatino (819191), Chengyu Zheng (820324)

Docente:
Prof.ssa Raffaela Mirandola

Versione 0.12 del 20/12/2013

Indice

1 Descrizione Generale

Travel Dream System e' nato per aiutare le persone a realizzare i loro desideri di viaggiare affidandosi ad una agenzia seria di viaggi che propone loro pacchetti vacanza organizzati che meglio si adattano alle loro esigenze. In tale ottica, ogni utente puo' accedere tramite interfaccia web al sito della TravelDream e prenotare con pochi e semplici passi, una vacanza da sogno.

1.1 Panoramica

Il Travel Dream System e' stato progettato con lo scopo di facilitare la pianificazione dei viaggi dei clienti, attraverso la scelta di pacchetti vacanza predefiniti che i clienti possono personalizzare. In tal modo si semplificano la scelta e la prenotazione di hotel e volo ed eventualmente l'utente può arricchire il proprio pacchetto attraverso l'aggiunta di escursioni organizzate.
A beneficiare della nuova piattaforma saranno anche gli impiegati della Travel Dream stessa che potranno gestire i pacchetti vacanza offerti ai clienti in modo efficiente ed efficace grazie alle funzionalità introdotte. Nel dettaglio, il presente documento di Design definisce la struttura concettuale e funzionale fornendo una precisa descrizione delle linee guida che saranno seguite nello sviluppo dell'applicazione. Il documento e' stato scritto attenendosi coerentemente ai requisiti presentati nel RASD. Il sistema e' utilizzato da utenti non registrati, clienti, impiegati e amministratori e un meccanismo di autenticazione permette di identificare a quale categoria appartiene l'utente nel momento del login.

1.2 Scopo del documento

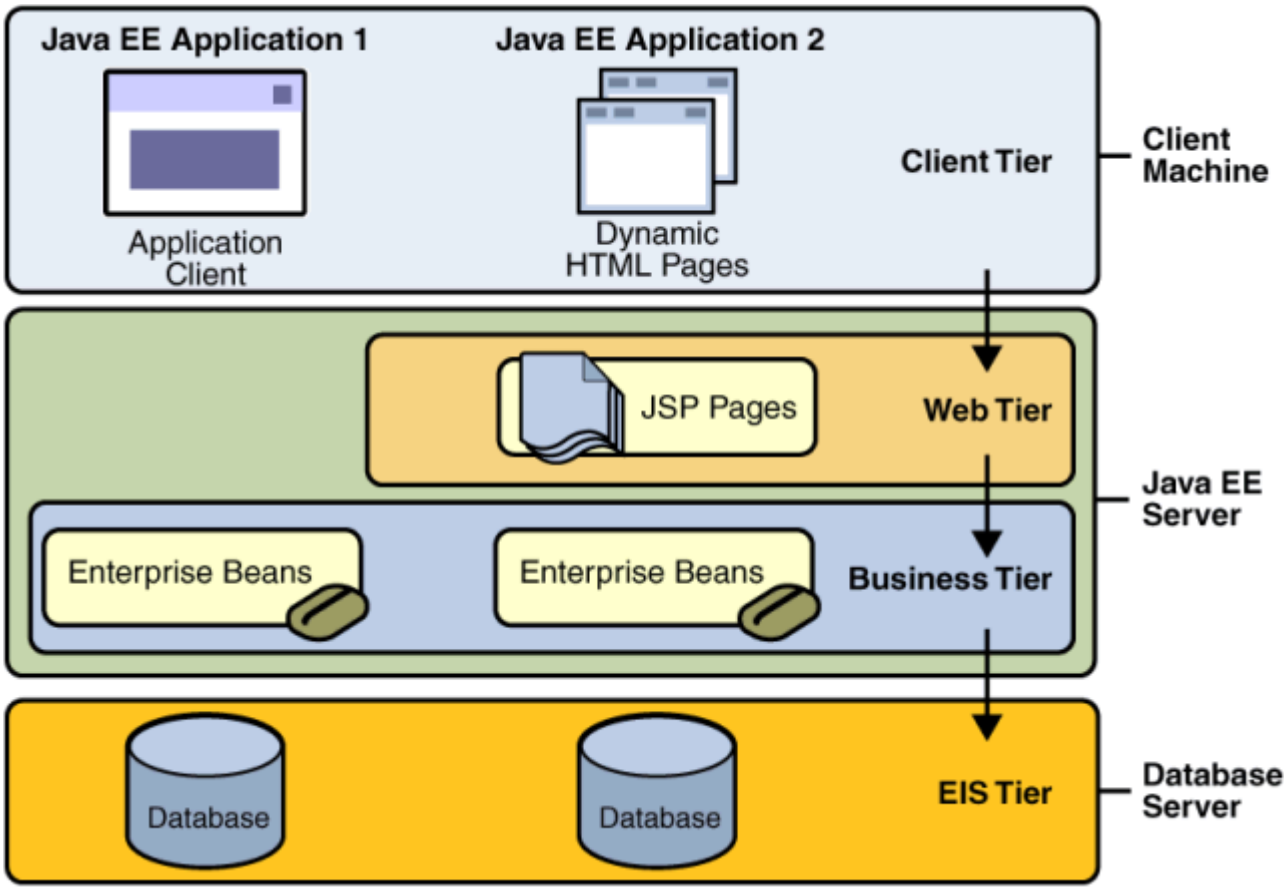
Lo scopo di questo documento e' quello di descrivere nel dettaglio, anche tramite l'ausilio di diagrammi ER, UML, UX, ecc. , lo scheletro del sistema, i suoi vari componenti (database, interfaccia utente e logica applicativa), le tecnologie utilizzate per comprendere al meglio come e' stata sviluppata TDS.

1.3 Divisione a tier

L'architettura a n-tier significa suddividere l'applicativo web in n moduli distinti o strati, assegnando ad ogni modulo compiti specifici. Tale architettura si presta bene nei casi in cui si vuole progettare un applicativo utilizzando il design pattern model view controller. Il sistema risulta composto da varie pagine htm con cui l'utente puo' interagire per reperire informazioni e/o richiedere un servizio. In ogni pagina viene utilizzata JavaScript, per poter recuperare e mostrare i dati all'utente. Tramite i Java Service e' poi possibile inviare richieste al business tier, che gestisce i dati e le operazioni necessarie per fornire le informazioni richieste dall'utente. Quando necessario, il server accede al database per recuperare i dati salvati o per inserirne di nuovi. La piattaforma risulta così un'applicazione three-tiered, composta da:

1. **Client tier:** e' composta dall'applicazione lato client gestita dai singoli utenti. Attraverso il browser, ogni utente invia richieste al server relativo.

2. **Business tier:** si occupa della logica applicativa. Sulla base dello standard J2EE deve essere composto da entita' che rappresentano informazioni durevoli raccolte in un database e session java beans, cioè oggetti che vengono associati con un singolo cliente. Il Business tier si occupa di ad esempio di rispondere alle richieste del client generando dinamicamente, raccogliere gli input forniti dagli utenti, mantenere le sessioni dati degli utenti
3. **Data tier:** include il database, accessibile dai componenti presenti nel business tier. Ha la funzione primaria di preservare i dati contenuti rispettando le proprietà ACID.



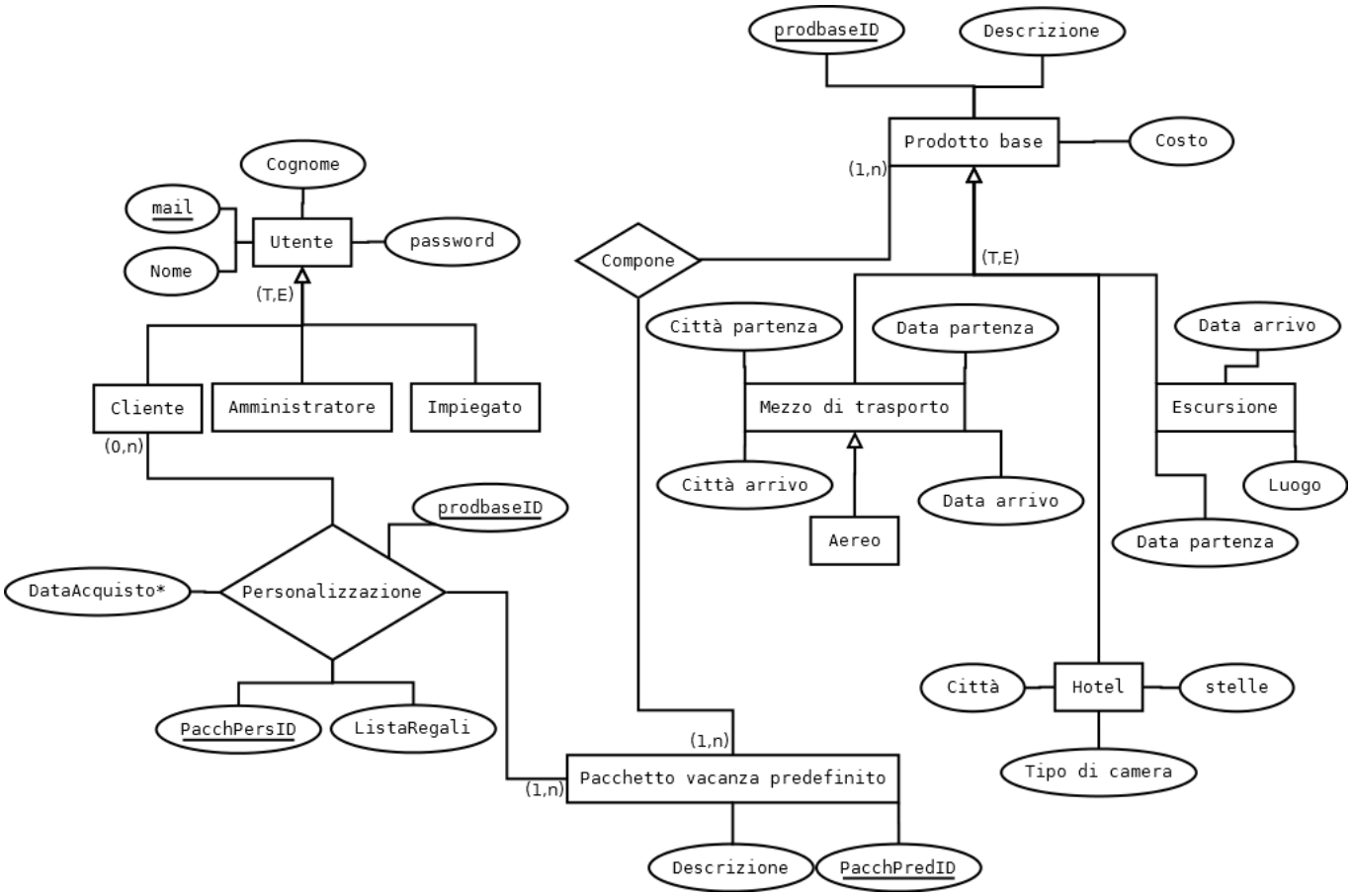
2 Progetto dei Dati

Vengono qui presentate le scelte progettuali nella creazione e gestione della base di dati. E' basata sul modello relazionale ed usa MySQL.

2.1 Progettazione concettuale

la progettazione concettuale serve a fornire una rappresentazione semplificata della realta' che al meglio rappresenti il sistema. I dati vanno individuati in modo tale da rispondere allo scopo per il quale il modello viene creato e per le successive fasi di design. Nell'ottica di realizzare una base di dati semplice ma completa, abbiamo individuato entità e relazioni di interesse per l'applicazione coerentemente con le specifiche del RASD.

Si è scelto di non modellare come eventuale entità gli inviti mandati da un cliente ad amici, poichè non si ritengono le e-mail degli amici, dati di interesse per l'applicazione.
Di seguito il diagramma ER.



2.1.1 Entità

- **Utente:** rappresenta un utente registrato al sistema, identificato dalla email e con attributi nome, cognome e password. E' suddiviso in Cliente, Amministratore e Impiegato. La partizione è totale ed esclusiva.

- **Prodotto Base:** rappresenta l'unità atomica dei servizi offerti, identificato da un codice identificativo prodBaseID ed ha come attributi comuni una descrizione e il costo. E' suddiviso in tre entità figlie.

  1. **Mezzo di Trasporto:** ha come attributi la città di partenza, la data e ora di partenza, la città di arrivo, data e ora di arrivo. La data di partenza e arrivo viene inserita perchè si ipotizza che l'agenzia Travel Dream prenoti in anticipo un numero stabilito di posti per quella tratta in tali giorni. Al momento l'unico mezzo di trasporto previsto dall'applicazione è l'aereo, ma non si esclude in futuro la possibilità di aggiungere altri mezzi di trasporto.
  2. **Hotel:** ha come attributi la città in cui si trova, il numero di stelle e il tipo di camera, ossia doppia, tripla.
  3. **Escursione:** è un'attività organizzata che ha come attributi la data di partenza e arrivo, per tenere conto di attività che durano più di una giornata, e il luogo dove si svolge.

- **Pacchetto Vacanza Predefinito:** rappresenta un insieme di prodotti base da personalizzare da parte di un cliente. Esso è identificato da pacchPredID ed ha come attributo una descrizione che contiene i dettagli della vacanza.

2.1.2 Relazioni

- **Personalizzazione:** rappresenta le scelte effettuate da un cliente a partire da un pacchetto vacanza predefinito, identificato da pacchPersID, avente come attributi prodBaseID, che indica i prodotti base scelti. L'attributo listaRegali, indica se il cliente ha creato di far diventare il suo pacchetto personalizzato, una lista regali. L'attributo data di acquisto viene inserito o al momento dell'acquisto del pacchetto vacanza o nel momento in cui un amico compra un prodotto base dalla lista regali del pacchetto.

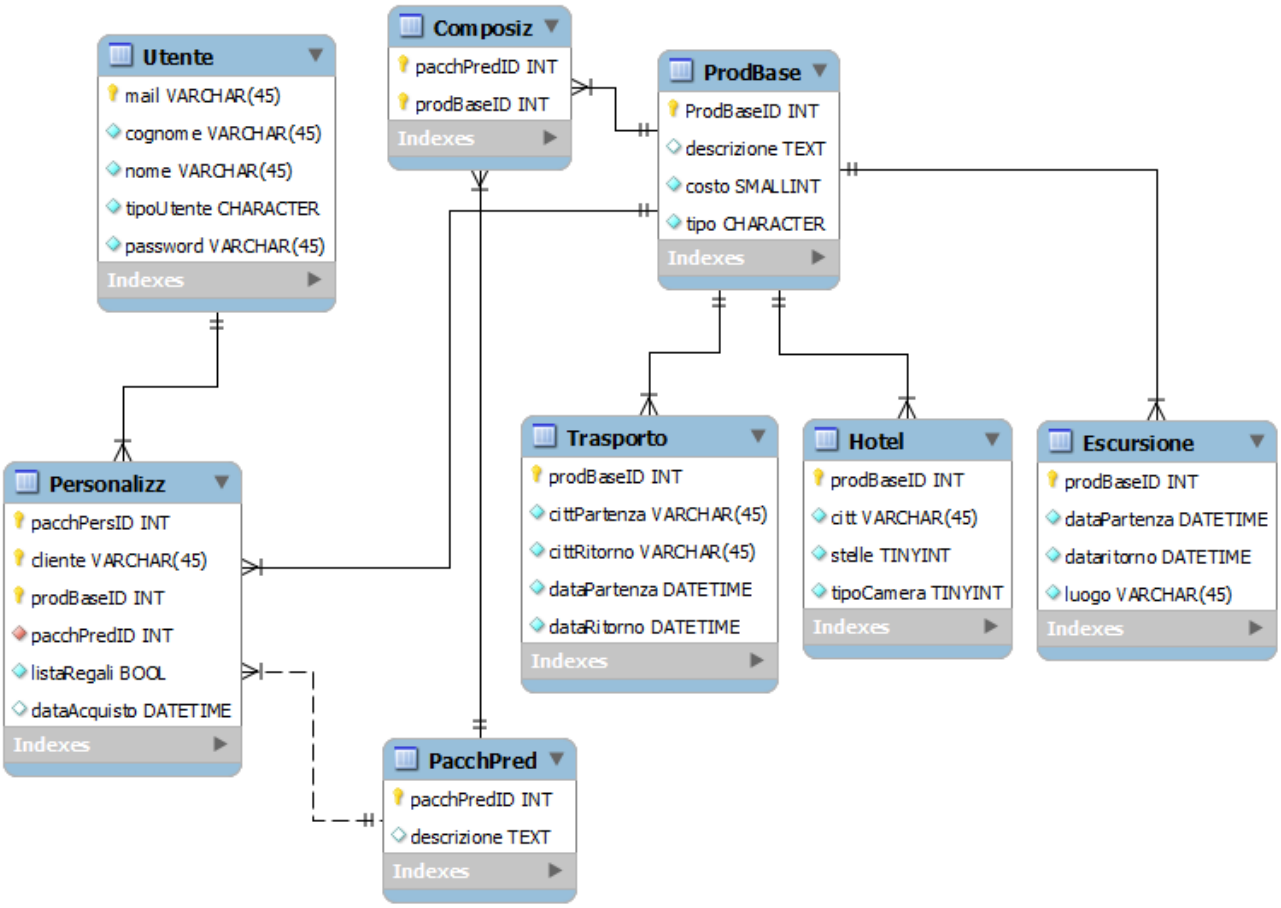
- **Compone:** rappresenta la composizione dei pacchetti vacanza predefiniti, ossia quali prodotti base compongono tale pacchetto.

2.2 Progettazione logica

In questa sezione lo schema del paragrafo precedente viene trasformato nel seguente schema logico corrispondente alla struttura effettiva della base di dati. Lo schema è il seguente:

- Utente(mail, cognome, nome, password, tipoUtente)
- ProdBase(prodBaseID, descrizione, costo, tipo)
- Hotel(prodBaseID, citt, stelle, tipoCamera)
- Trasporto(prodBaseID, cittPartenza, cittRitorno, dataPartenza, dataRitorno)
- Escursione(prodbaseID, dataPartenza, dataRitorno, luogo)
- PacchPred(pacchPredID, descrizione)
- Personalizz(pacchPersID, cliente, prodbaseID, pacchPredID, listaRegali, dataAcquisto*)
- Composiz(pacchPredID, prodBaseID)

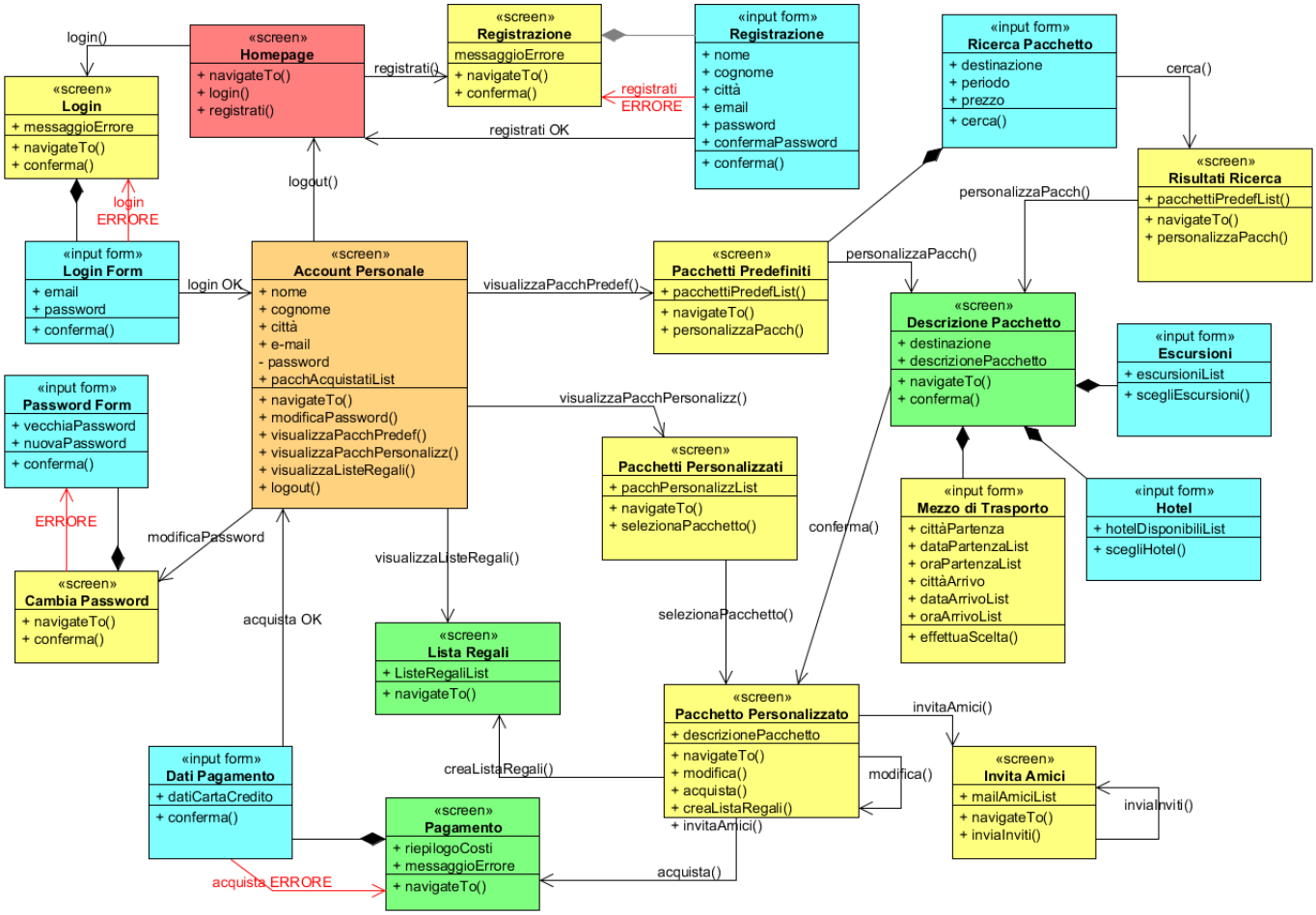
Lo schema di traduzione adottato è quello standard. Abbiamo collassato verso l'alto l'entità Utente ed introdotto l'attributo tipoUtente per differenziare gli utenti tra clienti amministratori ed impiegati. Per quanto riguarda l'entità prodotto base, abbiamo creato una tabella contenente gli attributi condivisi e tre tabelle deboli rispetto a ProdBase che contengono gli attributi specifici. Le relazioni Personalizzazione e Composizione sono molti a molti e perciò si rende necessario l'uso di una tabella apposita.


Utente
mail VARCHAR(45)
cognome VARCHAR(45)
nome VARCHAR(45)
tipoUtente CHARACTER
password VARCHAR(45)
Indexes
Composiz
pacchPredID INT
prodBaseID INT
Indexes
ProdBase
ProdBaseID INT
descrizione TEXT
costo SMALLINT
tipo CHARACTER
Indexes
Personalizz
pacchPersID INT
cliente VARCHAR(45)
prodBaseID INT
pacchPredID INT
listaRegali BOOL
dataAcquisto DATETIME
Indexes
Trasporto
prodBaseID INT
cittPartenza VARCHAR(45)
cittRitorno VARCHAR(45)
dataPartenza DATETIME
dataRitorno DATETIME
Indexes
Hotel
prodBaseID INT
citt VARCHAR(45)
stelle TINYINT
tipoCamera TINYINT
Indexes
Escursione
prodBaseID INT
dataPartenza DATETIME
dataritorno DATETIME
luogo VARCHAR(45)
Indexes
PacchPred
pacchPredID INT
descrizione TEXT
Indexes

3 Design dell'applicazione

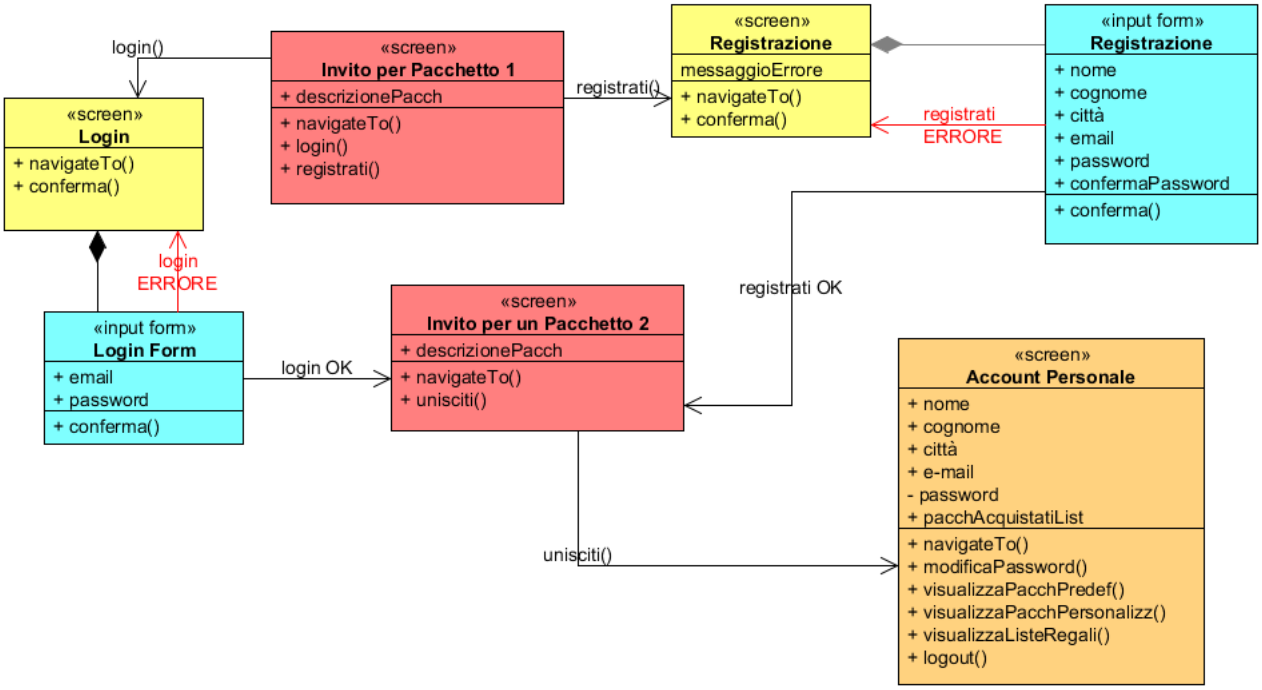
3.1 Modelli di navigazione

UX CLIENTE Il seguente diagramma ux rappresenta il modello di navigazione di un utente non registrato e di un cliente (un utente gia' registrato). Dalla home page, l'unica funzionalita' a cui puo' accedere un utente non registrato, e' la registrazione che gli permettera' in seguito di poter accedere al sistema tramite il login. Il cliente, una volta effettuato il login, viene indirizzato alla sua pagina personale in cui puo' modificare la password personale o effettuare varie operazioni come la ricerca di pacchetto vacanza predefinito e la sua personalizzazione. Si vuole fare notare che la creazione di una lista regalo o l'invio di inviti ad altri utenti, e' possibile solo a seguito della creazione di un pacchetto vacanza personalizzato.



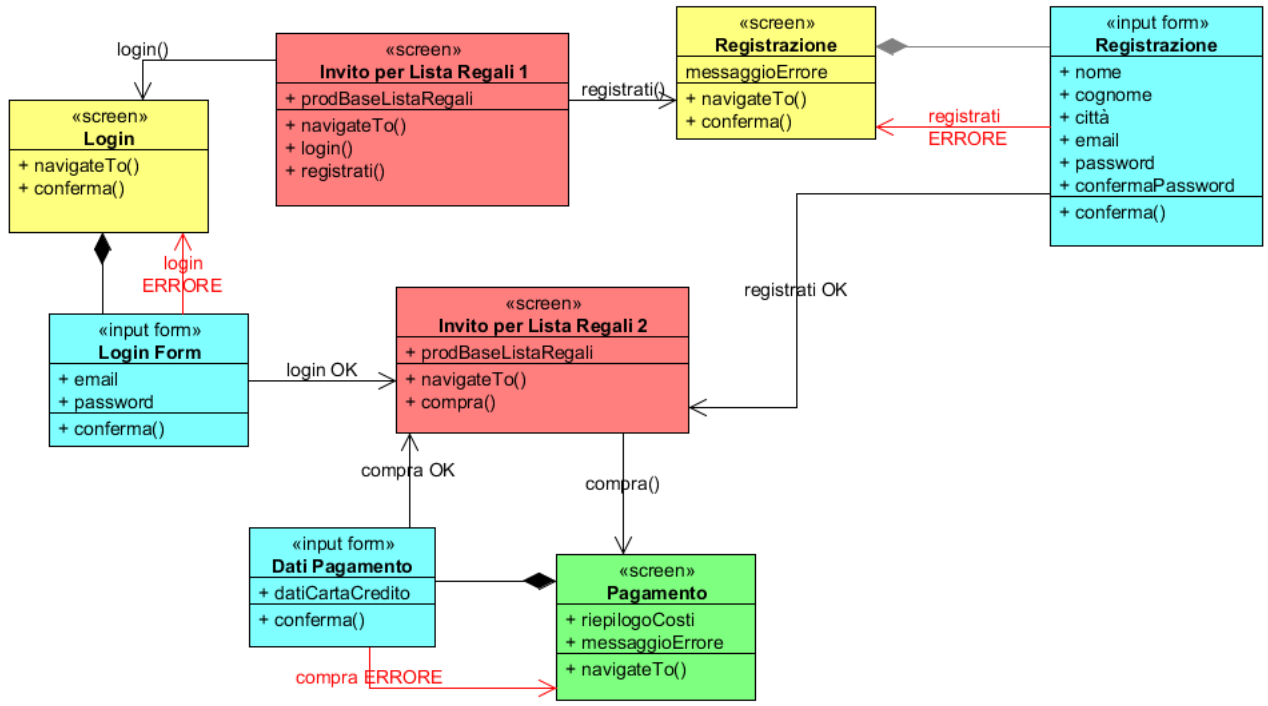
UX UTENTE RICEZIONE INVITO AD UNIRSI AD UN PACCHETTO VACANZA

Il seguente diagramma ux rappresenta il modello di navigazione di un utente non registrato o di un cliente nel momento in cui ricevono un invito ad unirsi ad un pacchetto vacanza personalizzato di un amico. Cliccando sul link contenuto nell'e-mail di invito, il sistema mostra all'utente una pagina che contiene la descrizione del pacchetto ed egli, se interessato all'offerta, potrà unirsi.
Attraverso il diagramma, si vuole evidenziare il fatto che tale azione di unione, è realizzabile solo se il cliente effettua il login o in caso, di utente non registrato, effettua la registrazione nel sistema. A termine del login o della registrazione, l'applicazione riporta loro nella pagina di invito, da cui possono concludere l'operazione. L'unione al pacchetto, significa che tale pacchetto viene salvato come pacchetto vacanza personalizzato nell'account dell'utente.



UX UTENTE RICEZIONE INVITO A COMPRARE UN PRODOTTO BASE DALLA LISTA REGALI DI UN AMICO

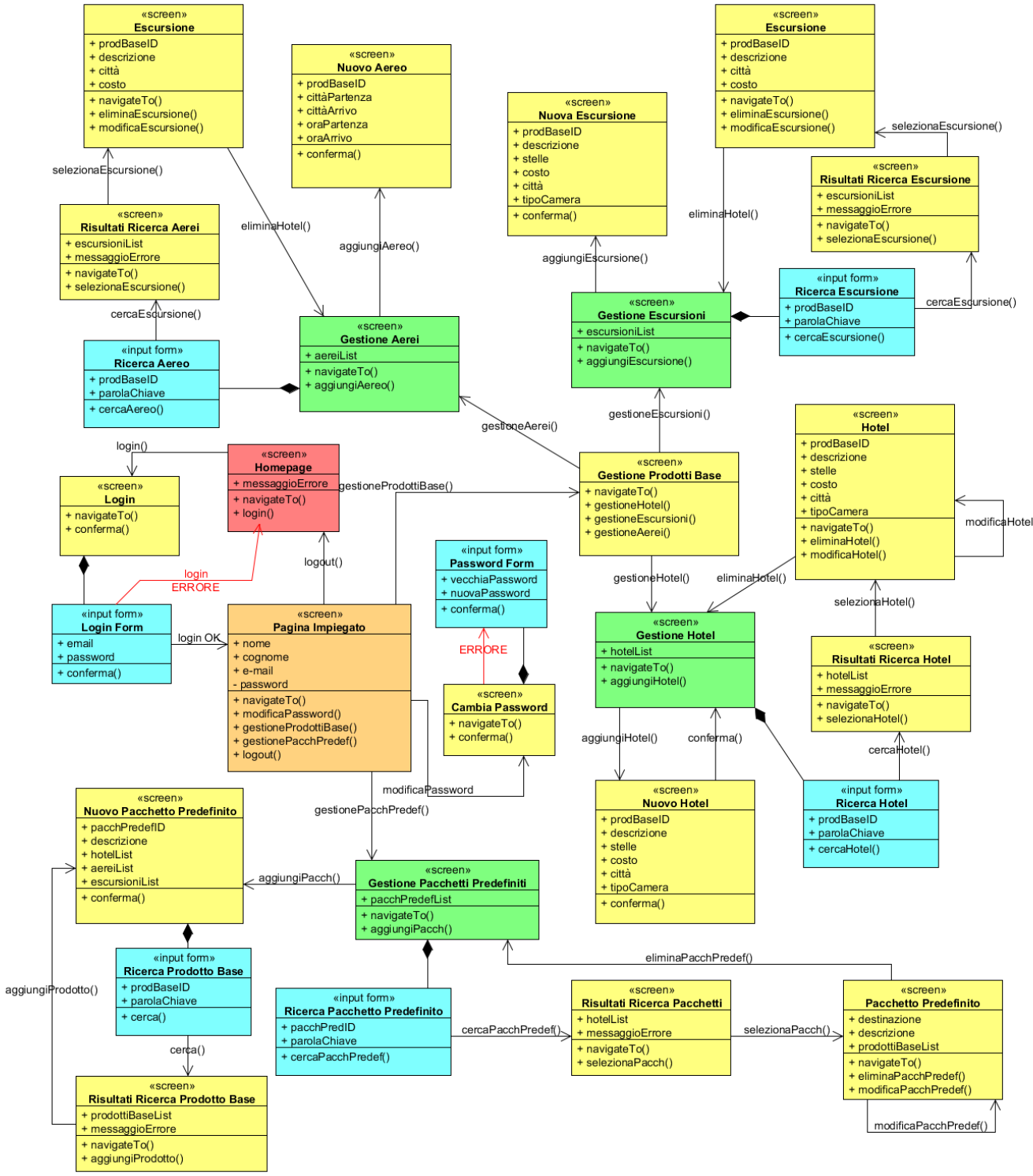
Il seguente diagramma ux mostra il modello di navigazione di un utente non registrato o di un cliente nel momento in cui ricevono un invito a comprare uno o più componenti di una lista regalo di un amico. Cliccando sul link contenuto nell'e-mail di invito, il sistema mostra loro una pagina che contiene l'elenco dei prodotti base che compongono la lista regali.
Come nel caso precedente, è obbligatorio il login o la registrazione prima di poter procedere ad un acquuisto. L'amico invitato può scegliere liberamente se comprare o meno un prodotto base per il suo amico. Nel caso in cui effettui un acquisto, deve inserire i dati della sua carta di credito.



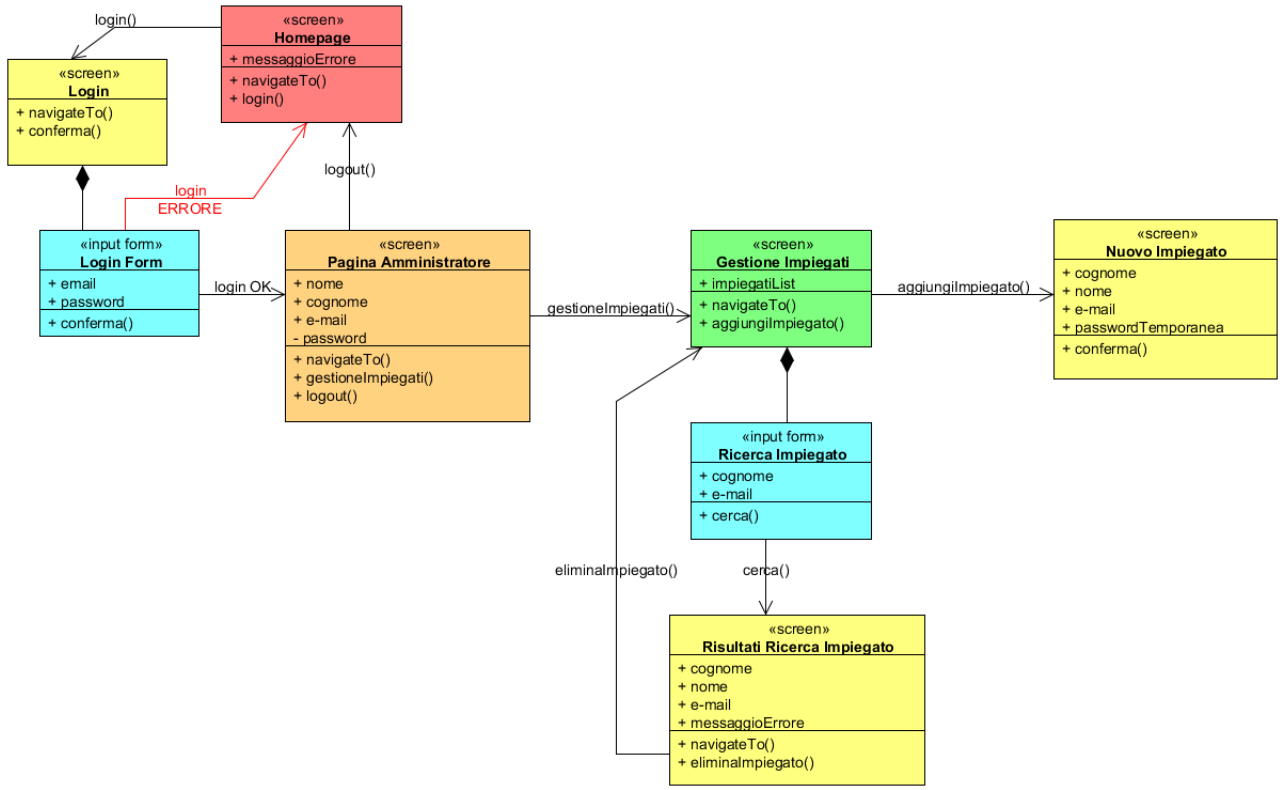
UX IMPIEGATO Il seguente diagramma rappresenta il modello di navigazione di un impiegato. Egli, una volta effettuato il login nel sistema, ha la facolta' di accedere alla gestione dei prodotti base e dei pacchetti vacanza predefiniti.

La gestione dei prodotti base e' divisa a sua volta in pagine diverse per permettere di gestire indipendentemente le tre tipologie di prodotti base quali hotel, aerei ed escursioni. Per ognuna delle tre tipologie e' possibile aggiungerne un nuovo prodotto base, ricercarne uno esistente per eventualmente modificarlo o eliminarlo dalla base di dati.

La gestione dei pacchetti vacanza predefiniti prevede la possibilita' di crearne uno nuovo o di ricercarne uno esistente per modificarlo o eliminarlo dal database. Poiche' un pacchetto e' formato da una lista di prodotti base esistenti, durante la creazione, l'impiegato ricerca nel database un prodotto base per aggiungerlo al pacchetto.

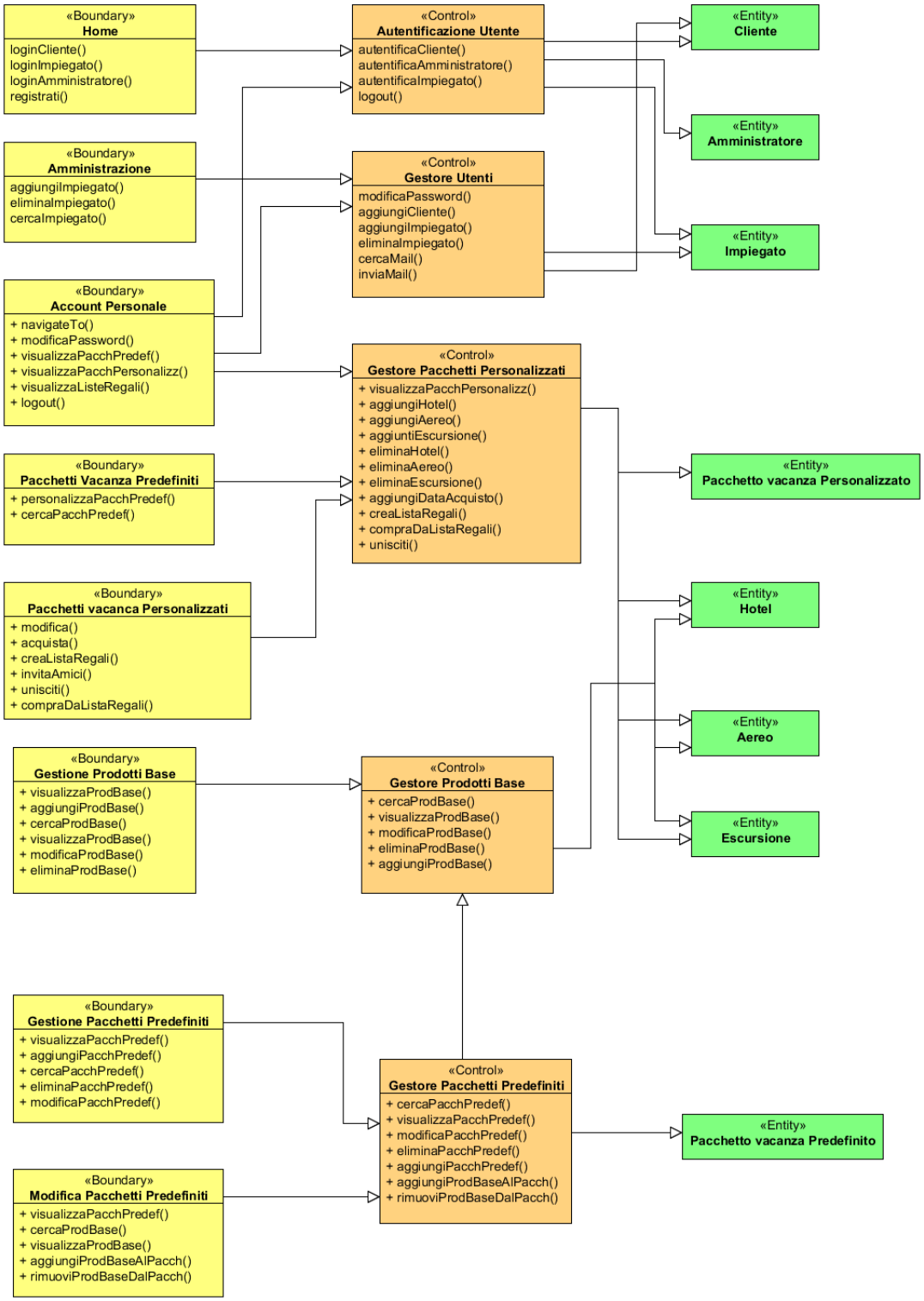

«screen»
Escursione
+ prodBaseID
+ descrizione
+ città
+ costo
+ navigateTo()
+ eliminaEscursione()
+ modificaEscursione()
«screen»
Nuovo Aereo
+ prodBaseID
+ cittàPartenza
+ cittàArrivo
+ oraPartenza
+ oraArrivo
+ conferma()
«screen»
Nuova Escursione
+ prodBaseID
+ descrizione
+ stelle
+ costo
+ città
+ tipoCamera
+ conferma()
«screen»
Escursione
+ prodBaseID
+ descrizione
+ città
+ costo
+ navigateTo()
+ eliminaEscursione()
+ modificaEscursione()
selezionaEscursione()
selezionaEscursione()
«screen»
Risultati Ricerca Escursione
+ escursioniList
+ messaggioErrore
+ navigateTo()
+ selezionaEscursione()
«screen»
Risultati Ricerca Aerei
+ escursioniList
+ messaggioErrore
+ navigateTo()
+ selezionaEscursione()
eliminaHotel()
aggiungiAereo()
aggiungiEscursione()
eliminaHotel()
«input form»
Ricerca Escursione
+ prodBaseID
+ parolaChiave
+ cercaEscursione()
cercaEscursione()
cercaEscursione()
«screen»
Gestione Escursioni
+ escursioniList
+ navigateTo()
+ aggiungiEscursione()
«input form»
Ricerca Aereo
+ prodBaseID
+ parolaChiave
+ cercaAereo()
«screen»
Gestione Aerei
+ aereiList
+ navigateTo()
+ aggiungiAereo()
gestioneAerei()
gestioneEscursioni()
«screen»
Hotel
+ prodBaseID
+ descrizione
+ stelle
+ costo
+ città
+ tipoCamera
+ navigateTo()
+ eliminaHotel()
+ modificaHotel()
login()
«screen»
Homepage
+ messaggioErrore
+ navigateTo()
+ login()
«screen»
Gestione Prodotti Base
+ navigateTo()
+ gestioneHotel()
+ gestioneEscursioni()
+ gestioneAerei()
gestioneProdottiBase()
modificaHotel()
«screen»
Login
+ navigateTo()
+ conferma()
logout()
«input form»
Password Form
+ vecchiaPassword
+ nuovaPassword
+ conferma()
gestioneHotel()
eliminaHotel()
selezionaHotel()
login
ERRORE
«input form»
Login Form
+ email
+ password
+ conferma()
login OK
«screen»
Pagina Impiegato
+ nome
+ cognome
+ e-mail
- password
+ navigateTo()
+ modificaPassword()
+ gestioneProdottiBase()
+ gestionePacchPredef()
+ logout()
ERRORE
«screen»
Cambia Password
+ navigateTo()
+ conferma()
«screen»
Gestione Hotel
+ hotelList
+ navigateTo()
+ aggiungiHotel()
«screen»
Risultati Ricerca Hotel
+ hotelList
+ messaggioErrore
+ navigateTo()
+ selezionaHotel()
aggiungiHotel()
conferma()
cercaHotel()
modificaPassword
gestionePacchPredef()
«screen»
Nuovo Hotel
+ prodBaseID
+ descrizione
+ stelle
+ costo
+ città
+ tipoCamera
+ conferma()
«input form»
Ricerca Hotel
+ prodBaseID
+ parolaChiave
+ cercaHotel()
«screen»
Nuovo Pacchetto Predefinito
+ pacchPredefID
+ descrizione
+ hotelList
+ aereiList
+ escursioniList
+ conferma()
aggiungiPacch()
«screen»
Gestione Pacchetti Predefiniti
+ pacchPredefList
+ navigateTo()
+ aggiungiPacch()
eliminaPacchPredef()
«input form»
Ricerca Prodotto Base
+ prodBaseID
+ parolaChiave
+ cerca()
aggiungiProdotto()
«input form»
Ricerca Pacchetto Predefinito
+ pacchPredID
+ parolaChiave
+ cercaPacchPredef()
cercaPacchPredef()
«screen»
Risultati Ricerca Pacchetti
+ hotelList
+ messaggioErrore
+ navigateTo()
+ selezionaPacch()
selezionaPacch()
«screen»
Pacchetto Predefinito
+ destinazione
+ descrizione
+ prodottiBaseList
+ navigateTo()
+ eliminaPacchPredef()
+ modificaPacchPredef()
cerca()
«screen»
Risultati Ricerca Prodotto Base
+ prodottiBaseList
+ messaggioErrore
+ navigateTo()
+ aggiungiProdotto()
modificaPacchPredef()

UX AMMINISTRATORE Il seguente diagramma rappresenta il modello di navigazione dell'amministratore. Egli, una volta effettuato il login, gestisce gli impiegati della Travel Dream. Egli puo' aggiungerne o eliminarne uno.



3.2 Componenti

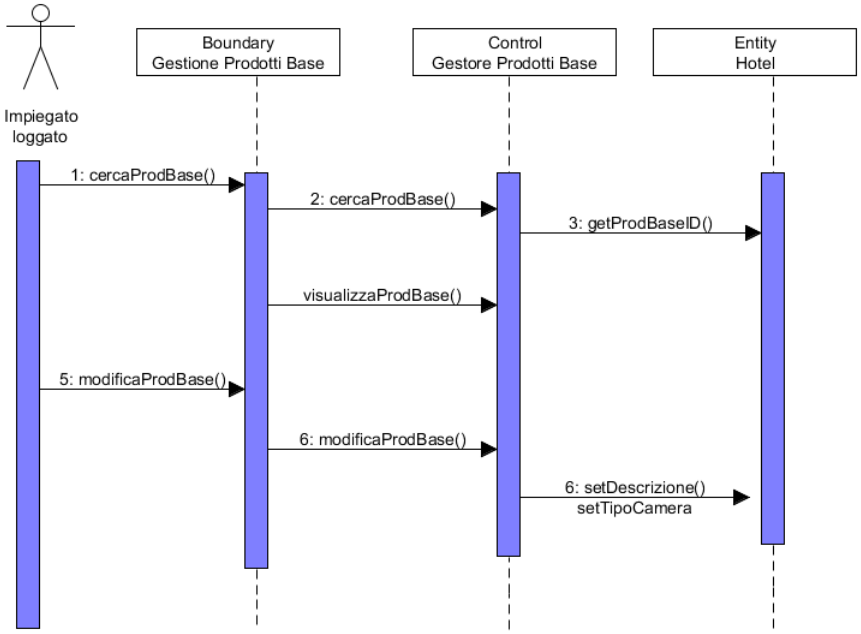
Al fine di definire i componenti da implementare per realizzare il sistema ci siamo basati sul pattern Model-View-Controller.
I vantaggi del modello MVC sono diversi e riguardano in particolare la riusabilità del codice e l'indipendenza dell'interfaccia utente dalla logica del sistema, consentendo la modifica di una delle due parti senza necessità di cambiamenti nell'altra. La parte Model incapsula le risorse dell'applicazione web, e scherma il resto dell'applicazione da eventuali cambiamenti. La parte View racchiude la parte di presentazione del sito, in questo modo è possibile cambiare il look-and-feel del sito agendo solo su questa parte. La parte Controller contiene racchiude la logica applicativa, cioè applicazioni che permettono all'utente di interagire col modello.
Il modello UML Boundary Control Entity mappa i componenti View, Model e Controller in rispettivamente Boundary, Entity e Control. Le Boundary rappresentano le funzionalità offerte all'utente. Gli oggetti Control svolgono un ruolo di mediazione tra Boundary ed Entity, implementano la logica richiesta per gestire i vari elementi e le loro interazioni. Le Entity rappresentano i dati del sistema. Le interfacce presentate all'utente, quindi le pagine HTML e JSP nel progetto finale, costituiranno l'implementazione delle Boundary. Gli elementi Control ed Entity saranno realizzati rispettivamente con SessionBeans e EntityBeans. I campi e i metodi degli oggetti Entity sono stati emessi per non appesantire troppo il diagramma. I campi si possono considerare gli stessi del diagramma ER presentato nella sezione riguardante la progettazione concettuale dei dati. Per i metodi nell'implementazione si avranno sicuramente metodi set per i campi modificabili, get per tutti i campi.


«Boundary»
Home
loginCliente()
loginImpiegato()
loginAmministratore()
registrati()
«Control»
Autentificazione Utente
autentificaCliente()
autentificaAmministratore()
autentificaImpiegato()
logout()
«Entity»
Cliente
«Entity»
Amministratore
«Boundary»
Amministrazione
aggiungiImpiegato()
eliminaImpiegato()
cercaImpiegato()
«Control»
Gestore Utenti
modificaPassword()
aggiungiCliente()
aggiungiImpiegato()
eliminaImpiegato()
cercaMail()
inviaMail()
«Entity»
Impiegato
«Boundary»
Account Personale
+ navigateTo()
+ modificaPassword()
+ visualizzaPacchPredef()
+ visualizzaPacchPersonalizz()
+ visualizzaListeRegali()
+ logout()
«Control»
Gestore Pacchetti Personalizzati
+ visualizzaPacchPersonalizz()
+ aggiungiHotel()
+ aggiungiAereo()
+ aggiuntiEscursione()
+ eliminaHotel()
+ eliminaAereo()
+ eliminaEscursione()
+ aggiungiDataAcquisto()
+ creaListaRegali()
+ compraDaListaRegali()
+ unisciti()
«Boundary»
Pacchetti Vacanza Predefiniti
+ personalizzaPacchPredef()
+ cercaPacchPredef()
«Entity»
Pacchetto vacanza Personalizzato
«Boundary»
Pacchetti vacanca Personalizzati
+ modifica()
+ acquista()
+ creaListaRegali()
+ invitaAmici()
+ unisciti()
+ compraDaListaRegali()
«Entity»
Hotel
«Entity»
Aereo
«Boundary»
Gestione Prodotti Base
+ visualizzaProdBase()
+ aggiungiProdBase()
+ cercaProdBase()
+ visualizzaProdBase()
+ modificaProdBase()
+ eliminaProdBase()
«Control»
Gestore Prodotti Base
+ cercaProdBase()
+ visualizzaProdBase()
+ modificaProdBase()
+ eliminaProdBase()
+ aggiungiProdBase()
«Entity»
Escursione
«Boundary»
Gestione Pacchetti Predefiniti
+ visualizzaPacchPredef()
+ aggiungiPacchPredef()
+ cercaPacchPredef()
+ eliminaPacchPredef()
+ modificaPacchPredef()
«Control»
Gestore Pacchetti Predefiniti
+ cercaPacchPredef()
+ visualizzaPacchPredef()
+ modificaPacchPredef()
+ eliminaPacchPredef()
+ aggiungiPacchPredef()
+ aggiungiProdBaseAlPacch()
+ rimuoviProdBaseDalPacch()
«Entity»
Pacchetto vacanza Predefinito
«Boundary»
Modifica Pacchetti Predefiniti
+ visualizzaPacchPredef()
+ cercaProdBase()
+ visualizzaProdBase()
+ aggiungiProdBaseAlPacch()
+ rimuoviProdBaseDalPacch()

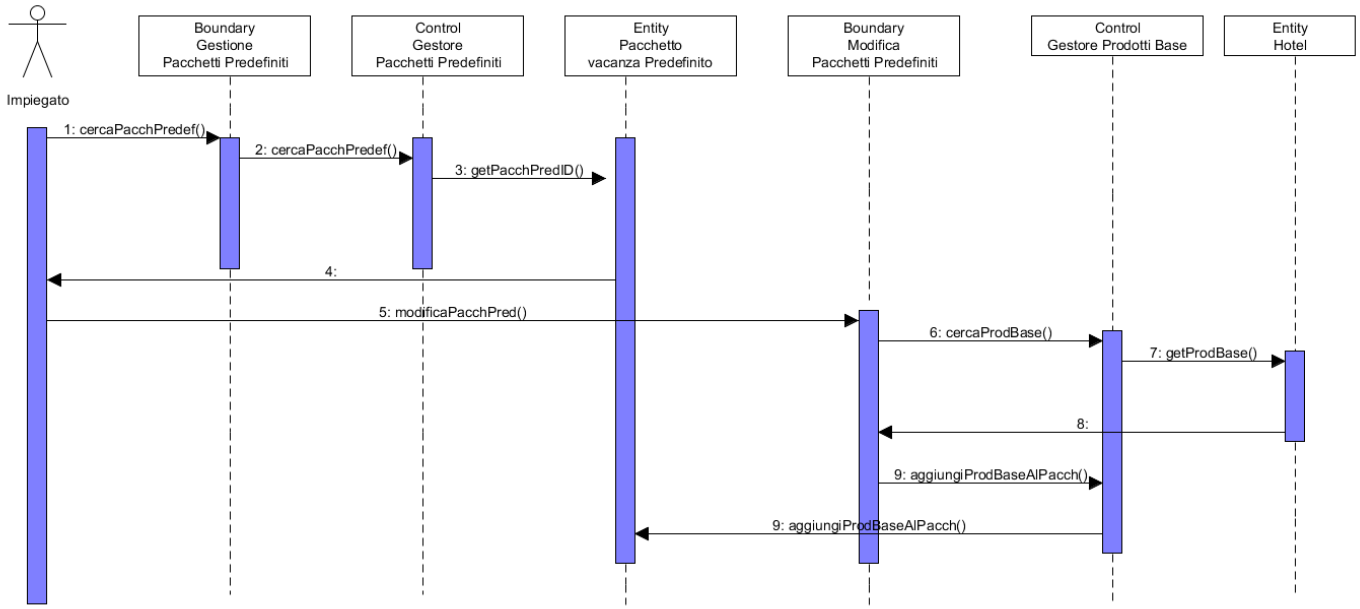
3.3 Diagrammi di sequenza

In questa sezione vengono presentati i sequence diagram delle funzionalità più significative del sistema.

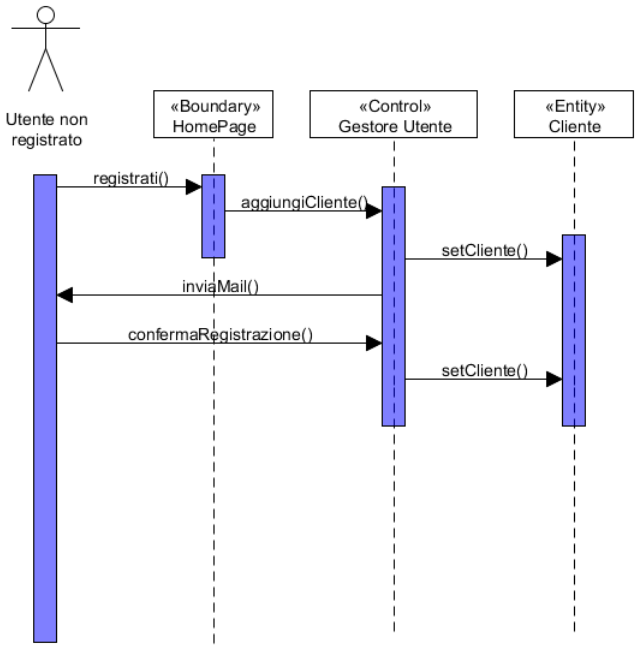
Nel seguente diagramma viene mostrata la sessione di lavoro di un impiegato in precedenza già loggato nel sistema. Egli effettua una ricerca di un prodotto base, ad esempio un hotel, dalla Boundary Gestione Prodotti Base, richiesta che viene trasmessa al relativo Control che a sua volta effettua un'interrogazione all'Entity per ottenere i dati richiesti.
A questo punto l'utente ha la facoltà attraverso la Boundary Risultati Ricerca Prodotto Base di modificarlo, cambiando qualche attributo come la descrizione. Tale modifica viene inserita nel database.



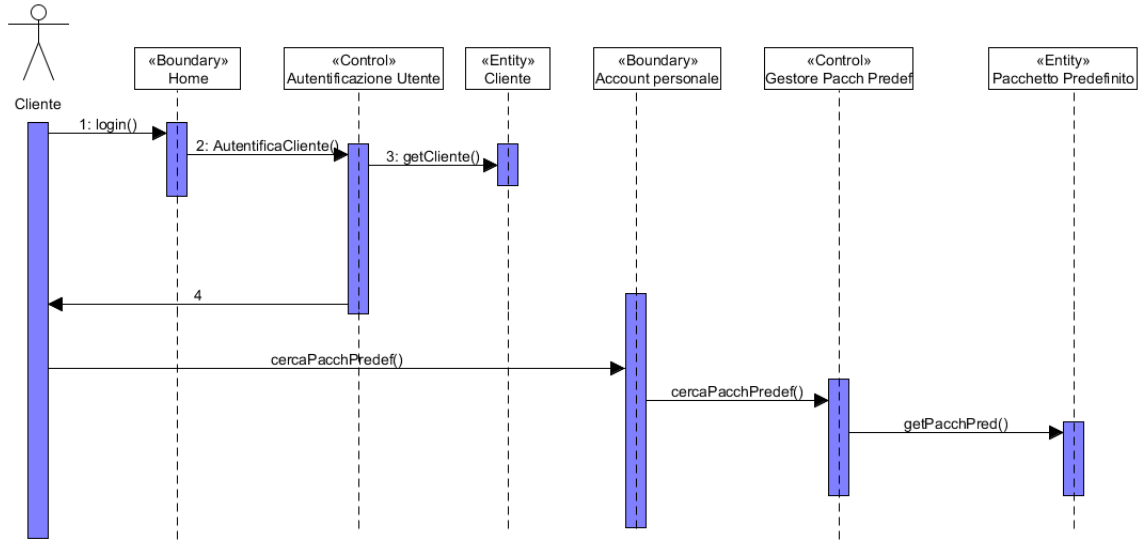
Nel seguente diagramma viene mostrata la sessione di lavoro di un impiegato in precedenza già loggato nel sistema, impegnato nella modifica di un pacchetto personalizzato esistente. Tramite la Boundary Gestione Pacchetti Predefiniti, l'impiegato effettua la ricerca di un pacchetto predefinito, richiesta che viene soddisfatta attraverso la relativa Control che effettua una query sulla Entity.
A questo punto, l'impiegato ricerca un prodotto base da aggiungere al pacchetto vacanza attraverso il metodo cercaProdBase() della Boundary Modifica Pacchetti Predefiniti. L'operazione si conclude con la Control Gestione Pacchetti Predefiniti che aggiorna la base di dati, modificando il pacchetto predefinito.



Nel seguente diagramma viene mostrata la registrazione di un utente al sito. Un utente fa partire la procedura di registrazione dalla Boundary di Homepage, quindi la Control di gestione Utenti viene invocata mediante aggiungiCliente. Questa si occupa di inserire il cliente nella Entity e di inviare una mail all'indirizzo dell'utente, il quale conferma la registrazione cliccando sul link apposito. La control quindi conferma il tutto



Nel seguente diagramma viene mostrato come un cliente può cercare un pacchetto vacanza predefinito. La procedura di login prevede l'invocazione del metodo relativo presente nella Boundary Home, che richiama la Control Autentificazione Utente che effettua un'interrogazione sull'Entity Cliente per accertare che l'e-mail e la password inserite dall'utente siano presenti in database.
Se l'autentificazione ha esito positivo, il cliente viene reindirizzato alla Boundary Account Personale. Da qui, ricerca un pacchetto vacanza predefinito, inserendo ad esempio come parametro di ricerca, la destinazione desiderata. La ricerca viene effettuata dalla Control Gestore Pacchetti Predefiniti attraverso la entity Pacchetti vacanza Predefiniti.



4 Ore di Lavoro

Il gruppo ha svolte le seguenti lavorative:

1. **Gabriele Rovaris:** 17 ore
2. **Rachele Sabatino:** 18 ore
3. **Chengyu Zheng:** 16 ore