Politecnico Di Milano

Scuola Di Ingegneria Dell'Informazione
Corso Di Laurea Magistrale In Ingegneria Informatica

Progetto di Ingegneria del Software 2
TravelDream

Design Document

Autori:
Francesca Garziera
Matricola 818925

Stefano Gianelli
Matricola 817398

Responsabile:
Prof. Luca Mottola

Anno Accademico 2013–2014

Indice

1 Introduzione **2**

1.1 Obiettivo del documento . . . 2
1.2 Panoramica del sistema . . . 2
1.3 Rettifiche al RASD . . . 2
1.4 Definizioni, acronimi e abbreviazioni . . . 3
1.5 Fonti . . . 4
1.6 Struttura del documento . . . 4

2 Architettura del sistema **5**

2.1 Architettura e tecnologia del sistema . . . 5
2.2 Moduli funzionali . . . 7

3 Progettazione della base di dati **9**

3.1 Progettazione concettuale . . . 9
3.1.1 Analisi delle entità . . . 10
3.1.2 Analisi delle relazioni . . . 12
3.2 Progettazione logica . . . 14
3.2.1 Schema logico . . . 14
3.2.2 Vincoli di integrità referenziale . . . 17
3.3 Schema database . . . 18

4 Descrizione dettagliata del design **19**

4.1 Functional View . . . 19
4.1.1 Diagrammi di analisi . . . 19
4.1.2 Diagrammi di sequenza . . . 22
4.2 Module View . . . 24
4.3 Deployment view . . . 25
4.4 Runtime view . . . 25
4.5 Modelli di navigazione . . . 26

1 Introduzione

1.1 Obiettivo del documento

L'obiettivo principale del documento è definire l'architettura del sistema e progettare lo schema logico della base di dati di TravelDream, sito di e-commerce creato e prossimamente implementato nell'ambito del corso di Ingegneria del Software 2.
La struttura concettuale del documento è coerente con quella precedentemente definita nel RASD, di cui il Design Document riprende e approfondisce alcuni aspetti funzionali.
È rivolto al team di sviluppo del software, infatti compie un'analisi più specializzata rispetto al RASD e si preoccupa di fornire ai programmatori le ultime specifiche necessarie per passare alla successiva fase di implementazione del prodotto.
Il documento è ricco di diagrammi e schemi che permettono di semplificare e modellizzare l'esposizione di determinati concetti: dalla progettazione della base di dati (diagramma ER e schema logico del database) all'interazione tra moduli funzionali (diagrammi BCE e di sequenza), fino alla navigazione all'interno delle pagine del software (diagrammi UX).

1.2 Panoramica del sistema

TravelDream permette agli utenti di acquistare pacchetti vacanza direttamente on-line, tramite un'interfaccia web di accesso semplice ed intuitiva. Ciò che maggiormente distingue questo sito di e-commerce dai prodotti concorrenti è la possibilità di creare pacchetti personalizzati, assemblando i singoli elementi inseriti dai dipendenti nel database: hotel, collegamenti tra le città ed escursioni.
Gli utenti non registrati al sistema (guest) possono esclusivamente visualizzare i pacchetti predefiniti, mentre gli utenti iscritti possono sfruttare tutte le funzionalità offerte dal sistema, tra cui la condivisione dei propri pacchetti con altre persone (amici) tramite email.
Gli utenti possono modificare i pacchetti salvati a proprio piacimento fino al momento dell'acquisto: modificando i prodotti selezionati (hotel, collegamenti, escursioni), modificando le date e il numero di partecipanti, aggiungendo destinazioni per creare un tour.
La semplicità dell'uso del software è data dagli iter di creazione e modifica guidati dei pacchetti, dai suggerimenti automatici del sistema e dal controllo degli input inseriti dall'utente.

1.3 Rettifiche al RASD

A livello concettuale non sono state apportate modifiche a quanto esposto nel RASD, tuttavia sono state aggiunte delle precisazioni su alcune assunzioni fatte nel corso della definizione del sistema, non presenti nel diagramma delle classi.

- All'entità Collegamento sono stati aggiunti gli attributi Origine e Destinazione, per specificare il punto di partenza o arrivo all'interno di una città (per es. Milano avrà Malpensa, Linate e Orio al Serio come aeroporti, nella prima versione del software)
- Nello use case UC22 la condizione di uscita era "visualizzazione della pagina di pagamento", mentre dopo l'acquisto l'utente ritorna alla pagina I miei pacchetti, in quanto nella prima versione del software non verrà implementata la fase di pagamento
- All'entità Pacchetto Predefinito è stato aggiunto l'attributo idPacchettoPredefinito per mantenere un collegamento con il pacchetto predefinito creato dal dipendente che l'utente ha salvato nella sua pagina personale

- Il sistema filtrerà le escursioni proposte all'utente all'interno di una destinazione in base alla regione, cioè mostrerà solo le escursioni organizzate nella stessa regione della città di destinazione.

1.4 Definizioni, acronimi e abbreviazioni

Definizioni:

- Javascript: è un linguaggio di scripting orientato agli oggetti comunemente usato nella creazione di siti web. Un tipico uso di Javascript in ambito web è la scrittura di piccole funzioni integrate nelle pagine per compiere determinate azioni non possibili con il solo HTML statico: controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, ecc.
- Enterprise Java Beans: sono i componenti software che implementano, lato server, la logica di business all'interno della piattaforma Java EE espletando servizi a favore della parte di front end, ovvero per la logica di presentazione di un'applicazione web.
- Entity Beans: il loro scopo è inglobare gli oggetti lato server che memorizzano i dati. Gli entity bean forniscono la caratteristica della persistenza dei dati.
- Integrità referenziale: è un vincolo di integrità di tipo interrelazionale ovvero una proprietà dei dati che, se soddisfatta, richiede che ogni valore di un attributo (colonna) di una relazione (tabella) esista come valore di un altro attributo in un'altra (o nella stessa) relazione. Nei database relazionali, affinché sia rispettata l'integrità referenziale, ogni campo in una tabella che sia stato dichiarato come foreign key può contenere solo valori della chiave primaria o chiave candidata di una tabella madre relazionata.

Acronimi e abbreviazioni:

- DD: Design Document
- SDS: Software Design Descriptions
- HTML: HyperText Markup Language
- HTTP: HyperText Transfer Protocol
- JDBC: Java DataBase Connectivity
- JSP: JavaServer Pages
- JSF: JavaServer Faces
- EJB: Enterprise JavaBean
- JEE: Java Enterprise Edition
- DBMS: DataBase Management System
- ER-model: Entity - Relationship model
- BCE: Boundary Control Entity
- UX: User eXperience

1.5 Fonti

- Progetto AA 2013-2014 (TravelDream)
  https://beep.metid.polimi.it/it (richiesto login)
- IEEE Standard for Information Technology - Systems Design - Software Design Descriptions
  http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=741934
- Software design description
  http://en.wikipedia.org/wiki/Software_design_description

1.6 Struttura del documento

1. ARCHITETTURA DEL SISTEMA: in questa sezione vengono esposte e motivate le scelte riguardanti l'architettura del sistema ed elencate le funzionalità che lo caratterizzano, raggruppate per categorie e distinte per attore.
2. PROGETTAZIONE BASE DI DATI: suddivisa in progettazione concettuale e progettazione logica, nella prima viene costruito il diagramma ER di cui vengono illustrate discorsivamente le entità e le relazioni scelte, nella seconda avviene la traduzione del diagramma ER nello schema logico, che servirà per la creazione vera e propria della base di dati.
3. DESCRIZIONE DETTAGLIATA DESIGN: riprendendo i concetti esposti nella prima parte del documento, vengono specificate le interazioni tra i diversi moduli funzionali e la loro suddivisione nei livelli dell'architettura, per fare ciò vengono impiegati diagrammi BCE e diagrammi di sequenza. Nel paragrafo conclusivo viene mostrata come avviene la navigazione all'interno dell'applicazione, utilizzando diagrammi UX.

2 Architettura del sistema

2.1 Architettura e tecnologia del sistema

Per lo sviluppo del software viene utilizzata un'architettura multi-tier in modo da separare le funzionalità su più livelli. Questa separazione è utile perché permette la creazione di software flessibili e scalabili. In particolare vengono utilizzati i seguenti strati:

- Client tier: si occupa principalmente della logica di presentazione e rappresenta il livello col quale interagiscono gli utenti. Comunica con il web tier tramite il protocollo HTTP, dal quale riceve le pagine HTML che deve mostrare e si occupa inoltre dell'invio degli input verso il server. Può anche eseguire codice Javascript.
- Web tier: si occupa della generazione delle pagine HTML da inviare ai client a partire dalle pagine dinamiche JSP / JSF e di ricevere gli input degli utenti; per questo motivo deve comunicare con il business tier.
- Business logic tier: implementa la logica dell'applicazione. In base agli input ricevuti dagli utenti esegue le funzionalità richieste e restituisce i risultati. Deve interagire col data tier per recuperare le informazioni necessarie. Nella piattaforma JEE questo livello è rappresentato dagli Enterprise Java Beans.
- Data tier: questo livello si occupa della persistenza dei dati. Viene utilizzato per memorizzare e recuperare le informazioni utilizzate dal business tier. Nella piattaforma JEE questo livello è rappresentato dagli Entity Beans.

Questa suddivisone viene generalmente considerata di tipo three-tier in quanto fisicamente gli strati precedenti vengono distribuiti in altrettanti livelli distinti:

- Client: corrisponde al client tier. Gli utenti possono interagire con questo livello tramite l'utilizzo di un browser.
- Application Server: in questo livello sono presenti il web tier e il business logic tier. Si tratta del livello che gestisce la logica applicativa. Comunica con i Client e il DBMS.
- DBMS: corrisponde al data tier e si occupa della persistenza dei dati.

Questi livelli possono essere divisi in tre (o più) macchine distinte ma nel nostro caso, viste le finalità didattiche del progetto, gireranno su un'unica macchina che svolgerà tutti i compiti. Durante lo sviluppo del software verrà seguita una struttura modulare che permette una divisione flessibile dei vari livelli.

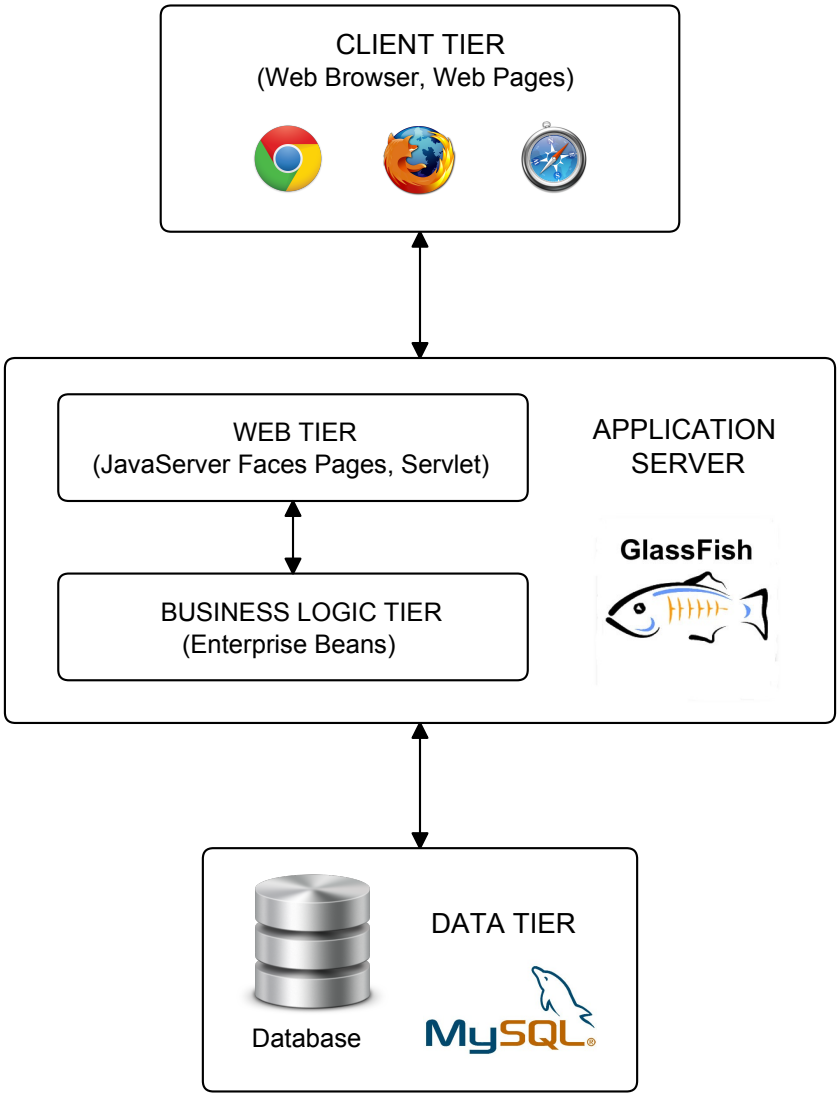

Figura 1: Architettura del sistema

Viene ora mostrato come interagiscono i vari componenti e attori con i livelli elencati in precedenza:

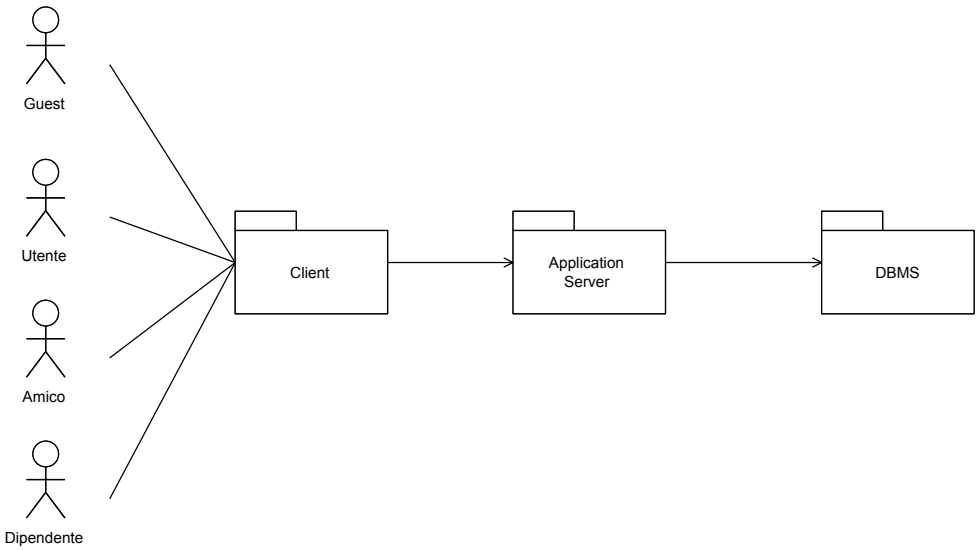

Figura 2: Moduli del sistema

2.2 Moduli funzionali

In questa sezione vengono elencate le funzionalità che verrano implementate, divise per categorie. Tra parentesi viene specificato l'attore che può eseguire l'azione.

- Gestione profilo
  - registrazione (guest)
  - login (utente / dipendente)
  - logout (utente / dipendente)
  - rigenerazione password (utente)
  - visualizzazione area personale (utente)
  - visualizzazione pagina dipendente (dipendente)
  - modifica dati personali (utente)
- Gestione pacchetto
  - elenco pacchetti (utente / dipendente / guest [1])
  - dettagli pacchetto (utente / dipendente / amico [2] / guest [1])
  - creazione pacchetto predefinito (dipendente)
  - creazione pacchetto personalizzato (utente)
  - modifica dati pacchetto (utente / dipendente)
  - salvataggio pacchetto (utente / dipendente)
  - eliminazione pacchetto (utente / dipendente)
  - acquisto pacchetto (utente)
  - condivisione pacchetto (utente)
- Gestione destinazione
  - aggiunta destinazione (utente)
  - visualizzazione città disponibili (utente)
  - modifica dati destinazione (utente)
  - eliminazione destinazione (utente)
- Gestione collegamento
  - creazione collegamento (dipendente)
  - aggiunta collegamento (dipendente / utente)
  - modifica dati collegamento (dipendente)
  - visualizzazione elenco collegamenti (dipendente)
  - visualizzazione collegamenti disponibili (utente)
  - dettagli collegamento (utente / dipendente)
  - eliminazione collegamento (dipendente)

[1] Un ospite può visualizzare solo l'elenco dei pacchetti predefiniti
[2] Un amico può visualizzare solo i dettagli del pacchetto che è stato condiviso con lui

- Gestione hotel
  - creazione hotel (dipendente)
  - aggiunta hotel (utente / dipendente)
  - modifica dati hotel (dipendente)
  - visualizzazione dettagli hotel (utente / dipendente)
  - visualizzazione elenco hotel (dipendente)
  - visualizzazione hotel disponibili (utente)
  - eliminazione hotel (dipendente)
- Gestione escursione
  - creazione escursione (dipendente)
  - aggiunta escursione (utente / dipendente)
  - visualizzazione dettagli escursione (utente / dipendente)
  - modifica dati escursione (dipendente)
  - modifica numero partecipanti (utente)
  - eliminazione escursione (utente / dipendente)

3 Progettazione della base di dati

3.1 Progettazione concettuale

Per progettazione concettuale si intende il processo di definizione delle entità e individuazione delle relazioni tra queste per arrivare a costruire un diagramma ER che rappresenterà la struttura portante del database che si andrà in seguito a costruire.

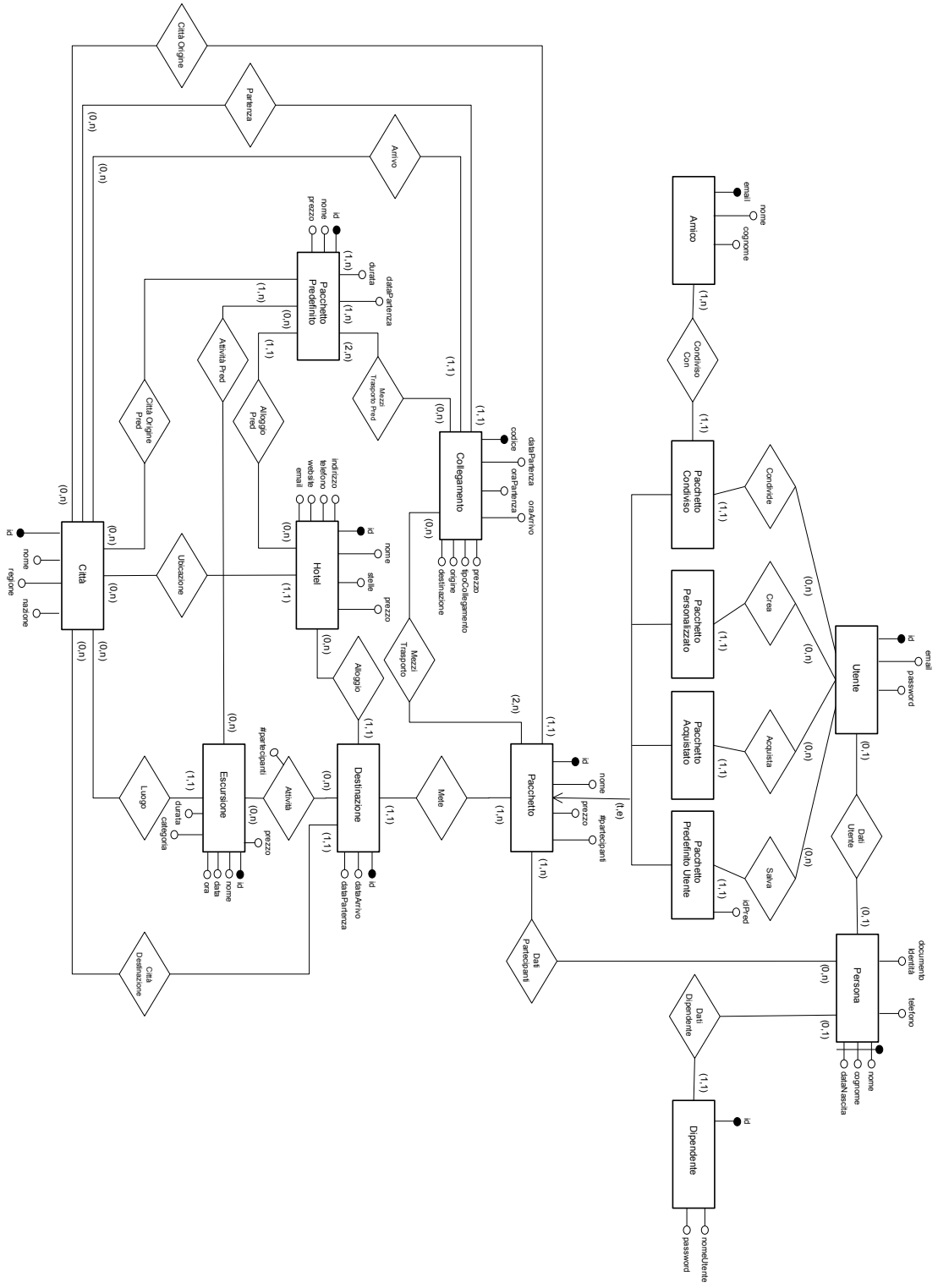

Figura 3: Diagramma ER

3.1.1 Analisi delle entità

- UTENTE: questa entità contiene tutti gli utenti iscritti a TravelDream e dotati quindi di una propria Area Personale dalla quale effettuare le operazioni di gestione dei pacchetti vacanza. È dotata degli attributi:
  - id: un campo univoco automaticamente generato dal sistema
  - email
  - password
- DIPENDENTE: rappresenta l'insieme dei dipendenti dell'agenzia TravelDream che sono in possesso delle credenziali per accedere al sistema ed apportarvi modifiche. Ogni dipendente ha:
  - id: un campo univoco automaticamente generato dal sistema
  - nomeUtente
  - password
- PERSONA: è l'entità contenente i dati personali degli utenti, dei dipendenti di TravelDream e degli eventuali partecipanti che l'utente può aggiungere nei propri pacchetti; sono i dettagli indispensabili per permettere al dipendente di prenotare un biglietto aereo. È composto dagli attributi:
  - nome
  - cognome
  - dataNascita
  - documentoIdentità: carta di identità o passaporto
  - telefono
- AMICO: questa entità rappresenta lo storico degli amici con cui gli utenti hanno condiviso i propri pacchetti. L'utente per condividere un pacchetto con un'altra persona deve inserire:
  - email
  - nome
  - cognome
- PACCHETTO: questa entità comprende tutti i pacchetti degli utenti. È stata impostata come padre di una gerarchia di entità in quanto costituisce la generalizzazione delle varie tipologie di pacchetti gestite dagli utenti. È costituita dai campi:
  - id: un campo univoco automaticamente generato dal sistema
  - nome: il nome assegnato al pacchetto dall'utente o dal sistema
  - prezzo
  - numeroPartecipanti: il numero minimo di partecipanti è 1 (l'utente stesso)
- PACCHETTO PERSONALIZZATO: è una specializzazione dell'entità pacchetto e tiene traccia dei pacchetti personalizzati, cioè quelli creati autonomamente dall'utente

- PACCHETTO PREDEFINITO UTENTE: è una specializzazione dell'entità pacchetto e include i soli pacchetti predefiniti che l'utente ha salvato nella propria Area personale. È dotata dell'attributo idPred che deriva dall'id dell'entità Pacchetto predefinito, per mantenere traccia del pacchetto predefinito originale che è stato salvato dall'utente e per permettere a quest'ultimo di effettuare modifiche del pacchetto salvato nell'area personale (dopo il salvataggio sono consentite le operazioni di modifica durata, città di partenza e data di partenza).
- PACCHETTO CONDIVISO: è una specializzazione dell'entità pacchetto e contiene i pacchetti che l'utente ha condiviso con un'altra persona (amico), si tratta di copie di pacchetti predefiniti o personalizzati salvati dall'utente
- PACCHETTO ACQUISTATO: è una specializzazione dell'entità pacchetto, vengono spostati in questa entità i pacchetti predefiniti e personalizzati che l'utente ha già acquistato
- PACCHETTO PREDEFINITO: questa entità include tutti i pacchetti predefiniti creati dai dipendenti di TravelDream, possono essere visualizzati dai guest e salvati degli utenti iscritti al sistema. È determinata dagli attributi:
  - id: un campo univoco automaticamente generato dal sistema
  - nome: il nome assegnato al pacchetto dal dipendente
  - prezzo
  - durata: il numero di giorni che dura la vacanza; questo attributo è caratterizzato da una cardinalità 1-a-molti, poiché l'utente può scegliere da una lista proposta dal sistema il numero di giorni desiderato
  - dataPartenza: questo attributo è caratterizzato da una cardinalità 1-a-molti, poiché l'utente può scegliere da una lista proposta dal sistema la data di partenza desiderata
- DESTINAZIONE: costituisce la parte basilare di un pacchetto, è creata dall'utente ed è l'entità che definisce le date di permanenza in una determinata città. È definita da:
  - id: un campo univoco automaticamente generato dal sistema
  - dataArrivo: la data di arrivo nella città
  - dataPartenza: la data di partenza dalla città (deve essere maggiore di dataArrivo)
- HOTEL: l'entità che accoglie tutti i dettagli riguardanti un determinato hotel inserito nel sistema dal dipendente, nello specifico:
  - id: un campo univoco automaticamente generato dal sistema
  - nome: il nome dell'hotel
  - stelle: memorizzato nel sistema come un numero da 1 a 5
  - prezzo: si intende il prezzo per notte
  - indirizzo
  - telefono
  - website
  - email

- COLLEGAMENTO: comprende i dettagli di tutti i possibili collegamenti tra due città del sistema, in particolare:
  - codice: un campo univoco automaticamente generato dal sistema
  - dataPartenza
  - oraPartenza
  - oraArrivo
  - prezzo
  - tipoCollegamento: nella prima versione del software sarà solamente "aereo", in futuro saranno presenti anche i campi "treno" e "auto"
  - origine: si intende il punto di partenza specifico nella città; nella prima versione del software l'utente potrà scegliere tra i vari aeroporti della città e limitrofi, successivamente anche tra le stazioni e gli uffici di noleggio auto
  - destinazione: si intende il punto di arrivo specifico nella città; nella prima versione del software l'utente potrà scegliere tra i vari aeroporti della città e limitrofi, successivamente anche tra le stazioni e gli uffici di noleggio auto
- ESCURSIONE: è l'entità che racchiude i dettagli di tutte le escursioni inserite nel sistema dai dipendenti, in dettaglio:
  - id: un campo univoco automaticamente generato dal sistema
  - nome: il nome assegnato all'escursione dal dipendente
  - data: la data in cui avrà luogo l'escursione (deve essere compresa tra dataArrivo e dataPartenza della rispettiva destinazione)
  - ora: l'orario di inizio dell'escursione
  - durata: la durata in ore dell'escursione (si presume minore di 24 ore)
  - categoria: la tipologia di escursione (selezionabile tra: Sport, Cultura, Relax, Mare, Montagna, Altro)
- CITTÀ: comprende tutte le città presenti nel sistema ed è caratterizzata dagli attributi:
  - id: un campo univoco automaticamente generato dal sistema
  - nome
  - regione
  - nazione

3.1.2 Analisi delle relazioni

- DATI UTENTE: lega l'entità Utente con l'entità Persona, permette ad ogni utente di memorizzare nel sistema i propri dati personali. È definita da una cardinalità (0,1) sia dal lato Utente poiché l'utente può essere iscritto al sito ma non aver ancora inserito i propri dettagli personali (significa che l'utente non ha ancora acquistato nessun pacchetto), sia dal lato Persona perché una persona può non essere un utente, ma essere un dipendente o un partecipante non iscritto al sistema.
- DATI DIPENDENTE: unisce Dipendente con l'entità Persona, legando ad ogni nomeUtente del sistema i rispettivi dati personali del dipendente. La cardinalità dal lato del Dipendente è (1,1) poiché ogni dipende dell'agenzia deve appartenere all'anagrafe nel database.

- DATI PARTECIPANTI: è la relazione tra un Pacchetto e l'entità Persona, obbliga l'utente ad inserire i propri dati personali e quelli degli eventuali partecipanti al momento dell'acquisto del pacchetto, infatti dal lato Pacchetto la cardinalità è (1,n) (c'è sempre almeno un partecipante al viaggio: l'utente stesso). Dal lato Persona, invece, la cardinalità è (0,n) poiché nel database ci sono sia persone che non compariranno mai in un pacchetto (per es. i dipendenti) sia persone che compariranno in più pacchetti (per es. un utente sarà un partecipante di tutti i pacchetti acquistati).
- CREA: abbina i pacchetti personalizzati all'utente che li ha creati. Un pacchetto è creato da un unico utente, cardinalità (1,1), mentre l'utente può creare più pacchetti personalizzati: (0,n).
- SALVA: collega un pacchetto predefinito all'utente che l'ha salvato nella propria Area Personale, dopo aver inserito alcuni dettagli di viaggio. L'utente può salvare più pacchetti predefiniti: (0,n), mentre un pacchetto predefinito diventa un pacchetto predefinito utente quando viene salvato e quindi diventa specifico per il singolo utente: (1,1).
- CONDIVIDE: definisce i pacchetti (sia predefiniti che personalizzati) che l'utente ha condiviso con un amico. Ogni pacchetto è stato condiviso dall'utente che l'ha creato (1,1), ma l'utente può condividere più pacchetti (0,n).
- CONDIVISO CON: determina l'amico con cui l'utente ha condiviso il pacchetto; viene creata una copia di un pacchetto salvato dall'utente per ogni amico con cui decide di condividere tale pacchetto, infatti la cardinalità dal lato Pacchetto Condiviso è (1,1). Con lo stesso amico, però, possono essere condivisi più pacchetti (minimo 1, altrimenti l'amico con apparterrebbe al database): (1,n).
- ACQUISTA: collega l'utente con i pacchetti che ha già acquistato, che possono essere numerosi (0,n), mentre ogni pacchetto è stato acquistato da un singolo utente (1,1).
- CITTÀ ORIGINE: determina la città di partenza di un pacchetto. Esiste un'unica città di partenza (1,1) per ogni pacchetto, mentre la stessa città può essere la città di partenza di più pacchetti (0,n) (è molto probabile che l'utente parta spesso dalla città in cui risiede).
- MEZZI TRASPORTO: definisce i mezzi di collegamento scelti dall'utente all'interno di un pacchetto per spostarsi da una destinazione all'altra. Un collegamento inserito nel database dal dipendente potrebbe non essere mai scelto da un utente per un proprio pacchetto (la cardinalità parte da 0) o essere scelto numerose volte (la cardinalità arriva fino a n). In un pacchetto salvato dall'utente, però, ci sono sempre minimo 2 collegamenti: il collegamento di andata tra la città di partenza e la destinazione e il collegamento di ritorno tra le stesse.
- METE: determina le destinazioni contenute in un pacchetto personalizzato. Ogni destinazione è singolare e creata ad hoc per uno specifico pacchetto, per questo la cardinalità è (1,1): una stessa destinazione non può appartenere a più di un pacchetto, ma nemmeno a nessuno altrimenti non sarebbe stata creata. Un pacchetto deve contenere almeno una destinazione poiché è il primo step necessario per creare un pacchetto personalizzato, però in un pacchetto possono esserci più di una destinazione (se si tratta di un tour): cardinalità (1,n).
- CITTÀ DESTINAZIONE: è la relazione che lega l'entità Destinazione con la città di destinazione contenuta nell'entità Città. Una città del database può appartenere a più destinazioni contemporaneamente o a nessuna (0,n), mentre in una destinazione può essere contenuta una sola città (per la definizione stessa di "destinazione"): (1,1).
- ALLOGGIO: abbina la destinazione con l'hotel selezionato nella stessa città. Una destinazione contiene un unico hotel (1,1), mentre un hotel inserito dal dipendente nel database può appartenere a zero o più destinazioni create dagli utenti (0,n).

- ATTIVITÀ: determina la lista delle escursioni che possono essere presenti in una destinazione. Entrambe le cardinalità sono (0,n) poiché l'inserimento di una escursione in una destinazione è facoltativo per l'utente, quindi può decidere di non selezionarne, mentre una escursione inserita nel database dal dipendente può essere scelta da più utenti o da nessuno.
- CITTÀ ORIGINE PRED: collega un pacchetto predefinito creato dal dipendente alle città tra cui l'utente può decidere di partire, infatti la cardinalità dal lato del Pacchetto predefinito è (1,n), poiché è necessaria almeno una città di partenza, ma ce ne sono (generalmente) più di una disponibili tra cui l'utente può scegliere. Dall'altro lato una città del database può essere la città di partenza di zero o più pacchetti predefiniti.
- MEZZI TRASPOSTO PRED: definisce i mezzi di collegamento proposti dal sistema all'interno di un pacchetto predefinito tra le città di partenza e la destinazione del pacchetto. Un collegamento inserito nel database dal dipendente potrebbe non essere mai usato all'interno di un pacchetto predefinito (la cardinalità parte da 0) o essere scelto numerose volte (la cardinalità arriva fino a n). In un pacchetto predefinito creato dal dipendente ci sono sempre minimo 2 collegamenti (andata e ritorno nel caso in cui sia proposta una sola città di partenza) fino a tutte le n possibili combinazioni di collegamenti tra le città di partenza proposte e la città di destinazione.
- ALLOGGIO PRED: come per la relazione Alloggio.
- ATTIVITÀ PRED: come per la relazione Attività.
- UBICAZIONE: lega l'hotel con la città in cui è situato. Un hotel può essere ubicato in un'unica città (1,1), ma in una città possono esserci numerosi hotel (0,n) (la cardinalità parte da 0 poiché in alcune città del database potrebbero non esserci hotel, per es. nelle città in cui vengono svolte le escursioni).
- LUOGO: abbina un'escursione alla città in cui è svolta. Un'escursione si svolge in un'unica città (1,1), mentre in una città possono esserci più escursioni (0,n).
- PARTENZA: lega il collegamento con la città da cui parte il mezzo di trasporto selezionato. La partenza può avvenire da una e una sola città (1,1), mentre da una città possono essere previsti diversi tipi di collegamenti e più collegamenti dello stesso tipo (0,n).
- ARRIVO: lega il collegamento con la città in cui arriva il mezzo di trasporto inserito nel database. Una città può essere raggiunta tramite diversi mezzi di trasporto (0,n), mentre un singolo collegamento arriva in un'unica città (1,1).

3.2 Progettazione logica

3.2.1 Schema logico

La progettazione logica consiste nella traduzione del diagramma ER in uno schema logico che definisca le tabelle, le chiavi primarie e le chiavi esterne che andranno a costituire gli elementi del database del sistema. Questo processo si sviluppa a partire dalla traduzione vera e propria dello schema ER, svolta seguendo determinate regole prestabilite, per giungere alla costruzione dello schema logico e alla definizione dei vincoli di integrità referenziale che lo caratterizzano. Il paragrafo si conclude con lo schema del database, che offre un modello visivo delle tabelle definite nello schema logico, delle relazioni tra queste, delle cardinalità che le caratterizza e del tipo di dati degli attributi.

La prima fase di ristrutturazione del diagramma logico consiste nell'eliminazione delle gerarchie dello schema ER. Nel nostro caso la gerarchia è rappresentata dall'entità Pacchetto e dalle sue quattro specializzazioni: Pacchetto Personalizzato, Pacchetto Predefinito Utente, Pacchetto Condiviso e Pacchetto Acquistato. Si è deciso di far collassare la gerarchia verso l'alto in quanto i figli non hanno attributi propri (ad eccezione di idPred del Pacchetto Predefinito Utente), quindi effettuando questa scelta non andiamo a perdere informazioni specifiche sui figli. Inoltre si tratta di una gerarchia totale ed esclusiva, quindi l'entità padre contiene tutte le tuple dei figli (totalità) e non esistono tuple ripetute (esclusività), questo conferma la scelta di conservare la generalizzazione nel progetto logico.
Nel diagramma ER non sono presenti né attributi composti né entità deboli, quindi non è stato necessario fare delle scelte di traduzione per questo tipo di elementi.
A questo punto siamo passati alla traduzione delle entità forti e delle relazioni in schemi relazionali.

Nella fase di traduzione si passa alla vera e propria creazione dello schema logico, durante la quale sono state seguite le seguenti regole:

- nella gerarchia viene tradotto solo il padre (generalizzazione) e non le entità figlie (specializzazioni), quindi nello schema logico compare una tabella corrispondente all'entità Pacchetto, mentre non saranno tradotte in tabelle le entità Pacchetto Personalizzato, Pacchetto Predefinito Utente, Pacchetto Condiviso e Pacchetto Acquistato
- l'unico attributo che possedeva uno dei figli (idPred di Pacchetto Predefinito Utente) viene aggiunto alla tabella del padre e può essere un campo Null (se la tupla non corrisponde ad un Pacchetto Predefinito)
- tutte le altre entità vengono tradotte in tabelle
- le relazioni 0-a-molti, 1-a-molti e molti-a-molti vengono tradotte in tabelle le cui chiavi saranno composte dall'unione delle chiavi delle due entità ai capi della relazione
- la relazione Condiviso Con viene tradotta in una tabella (nonostante da un lato sia una relazione 1-a-1) per non dover inserire l'attributo email dell'entità Amico nella tabella Pacchetto, per evitare di ottenere molti campi Null in questa tabella (poiché si presume che la maggior parte dei pacchetti salvati dall'utente non siano stati condivisi con nessuno)
- per le entità caratterizzate da una relazione 1-a-1 viene scelta una chiave esterna dall'entità collegata da inserire nella tabella creata (la chiave esterna solitamente è l'id dell'entità collegata o comunque un campo univoco)
- gli attributi con cardinalità 1-a-molti vengono tradotti in tabella; si tratta degli attributi durata e dataPartenza dell'entità Pacchetto Predefinito, le cui tabelle risultanti saranno composte dall'id del Pacchetto e, rispettivamente, dalle durate e dalle date di partenza.

Dall'applicazione delle suddette regole, ne deriva il seguente schema logico del database:

UTENTE (id, email, password)
PERSONA (nome, cognome, dataNascita, documentoIdentità, telefono, idUtente*, idDipendente*)
DIPENDENTE (id, nomeUtente, password)
AMICO (email, nome, cognome)
CONDIVISOCON (idPacchetto, emailAmico)
PACCHETTO (id, nome, #partecipanti, prezzo, idCittàOrigine, tipoPacchetto, idUtente, idPred*)
MEZZITRASPORTO (idPacchetto, idCollegamento)
DATIPARTECIPANTI (idPacchetto, nomePart,cognomePart, dataNascitaPart)
COLLEGAMENTO (codice, tipoCollegamento, cittàPartenza, cittàArrivo, origine, destinazione, dataPartenza, oraPartenza, oraArrivo, prezzo)
HOTEL (id, nome, stelle, indirizzo, città, telefono, website, email, prezzo)
ESCURSIONE (id, nome, città, data, ora, durata, categoria, prezzo)
DESTINAZIONE (id, città, dataArrivo, dataPartenza, idHotel, idPacchetto)
ATTIVITÀ (idDestinazione, idEscursione, #partecipanti)
PACCHETTOPREDEFINITO (id, nome, prezzo, idHotel)
CITTÀORIGINEPRED (idPacchettoPredefinito, idCittà)
MEZZITRASPORTOPRED (idPacchettoPredefinito, idCollegamento)
ATTIVITÀPRED (idPacchettoPredefinito, idEscursione)
CITTÀ (id, nome, regione, nazione)
DATEPARTENZA (idPacchettoPredefinito, data)
DURATE (idPacchettoPredefinito, durata)

Il simbolo * indica che il campo può assumere valore nullo.

3.2.2 Vincoli di integrità referenziale

PERSONA.idUtente → UTENTE.id
PERSONA.idDipendente → DIPENDENTE.id
CONDIVISOCON.idPacchetto → PACCHETTO.id
CONDIVISOCON.emailAmico → AMICO.email
PACCHETTO.idCittàOrigine → CITTÀ.id
PACCHETTO.idUtente → UTENTE.id
PACCHETTO.idPred → PACCHETTOPREDEFINITO.id
MEZZITRASPORTOPRED.idPacchettoPredefinito → PACCHETTOPREDEFINITO.id
MEZZITRASPORTOPRED.idCollegamento → COLLEGAMENTO.codice
DATIPARTECIPANTI.idPacchetto → PACCHETTO.id
DATIPARTECIPANTI.[nomePart,cognomePart,dataNascitaPart] → PERSONA.[nome, cognome, dataNascita]
COLLEGAMENTO.cittàPartenza → CITTÀ.id
COLLEGAMENTO.cittàArrivo → CITTÀ.id
HOTEL.città → CITTÀ.id
ESCURSIONE.città → CITTÀ.id
DESTINAZIONE.città → CITTÀ.id
DESTINAZIONE.idHotel → HOTEL.id
DESTINAZIONE.idPacchetto → PACCHETTO.id
ATTIVITÀ.idDestinazione → DESTINAZIONE.id
ATTIVITÀ.idEscursione → ESCURSIONE.id
PACCHETTOPREDEFINITO.idHotel → HOTEL.id
CITTÀORIGINEPRED.idPacchettoPredefinito → PACCHETTOPREDEFINITO.id
CITTÀORIGINEPRED.idCittà → CITTÀ.id
MEZZITRASPORTO.idPacchetto → PACCHETTO.id
MEZZITRASPORTO.idCollegamento → COLLEGAMENTO.codice
ATTIVITÀPRED.idPacchettoPredefinito → PACCHETTOPREDEFINITO.id
ATTIVITÀPRED.idEscursione → ESCURSIONE.id
DATEPARTENZA.idPacchettoPredefinito → PACCHETTOPREDEFINITO.id
DURATE.idPacchettoPredefinito → PACCHETTOPREDEFINITO.id

3.3 Schema database

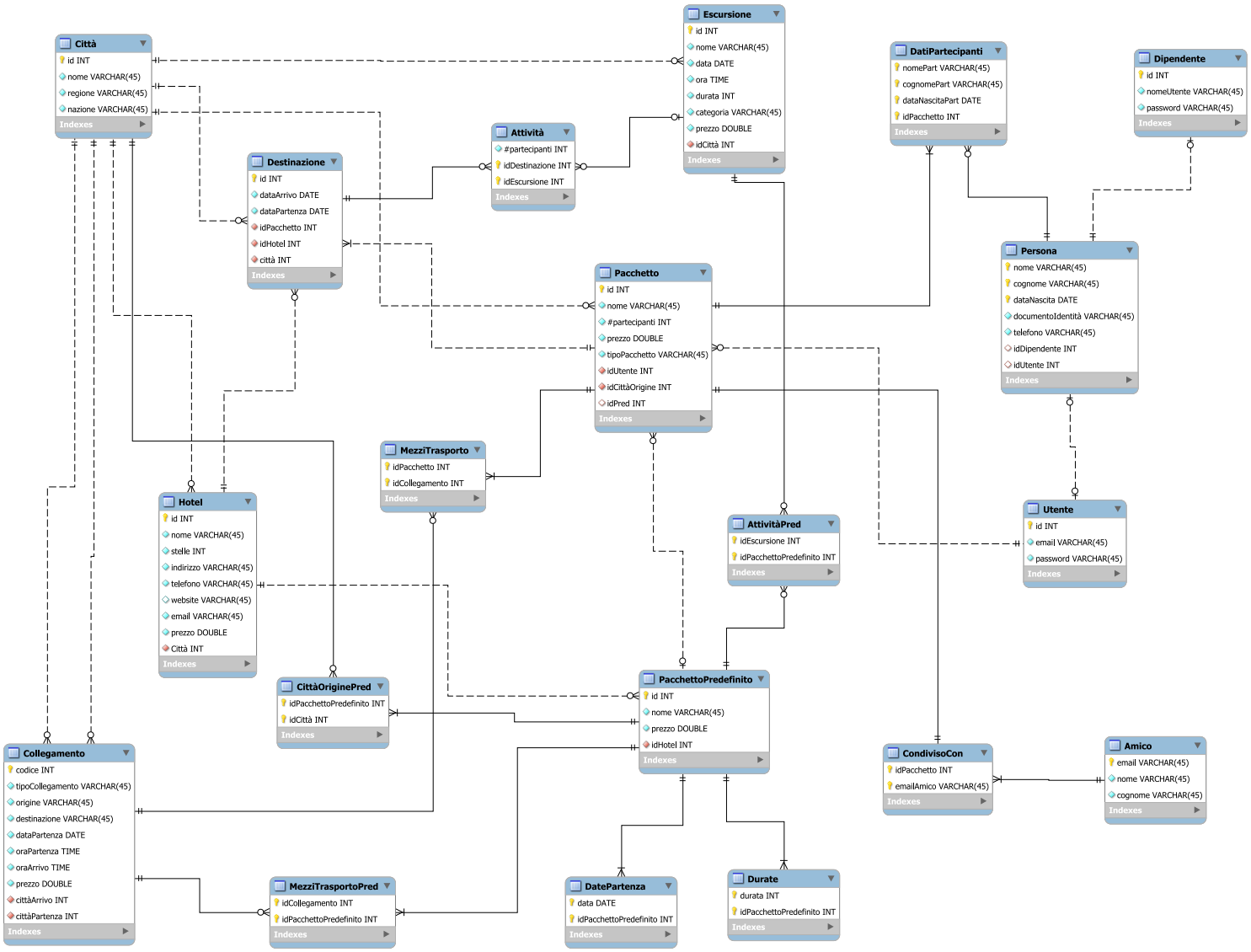

Figura 4: Schema Database

4 Descrizione dettagliata del design

4.1 Functional View

4.1.1 Diagrammi di analisi

In questa sezione viene mostrata l'interazione tra i diversi moduli funzionali e la loro suddivisione nei vari livelli dell'architettura three-tier scelta. Si è deciso di utilizzare dei diagrammi BCE così composti:

- boundary: queste classi sono predisposte all'interazione con l'utente. Si occupano di mostrare le informazioni agli utenti e di accettare input esterni e trasferirli alla logica applicativa del sistema (control). Per questo motivo possono interagire solamente con questi ultimi. Non possiedono attributi ma solo metodi: alcuni coincidono con quelli degli screen dei diagrammi UX che gestiscono l'input dell'utente. Altri metodi servono a fornire degli output all'utente. Questi metodi verranno implementati dalle pagine JSP o JSF
- control: queste classi implementano la logica applicativa del sistema. Forniscono la logica alle classi boundary. Utilizzano a loro volta le classi entity per acquisire i dati necessari. Le classi control possono avere associazioni tra loro se necessario. Nella piattaforma JEE corrispondono ai Session Bean o ai Message Driven Bean
- entity: queste classi vengono utilizzate per l'accesso ai dati e alle risorse. Gli attributi rappresentano i dati disponibili. Possiedono metodi per l'elaborazione delle informazioni, come i setter, getter, metodi di ricerca, ecc. Nei seguenti grafici questi metodi vengono omessi. Nella piattaforma JEE corrispondono agli Entity Bean

Per renderli più chiari, i diagrammi sono stati divisi per funzionalità o attore.

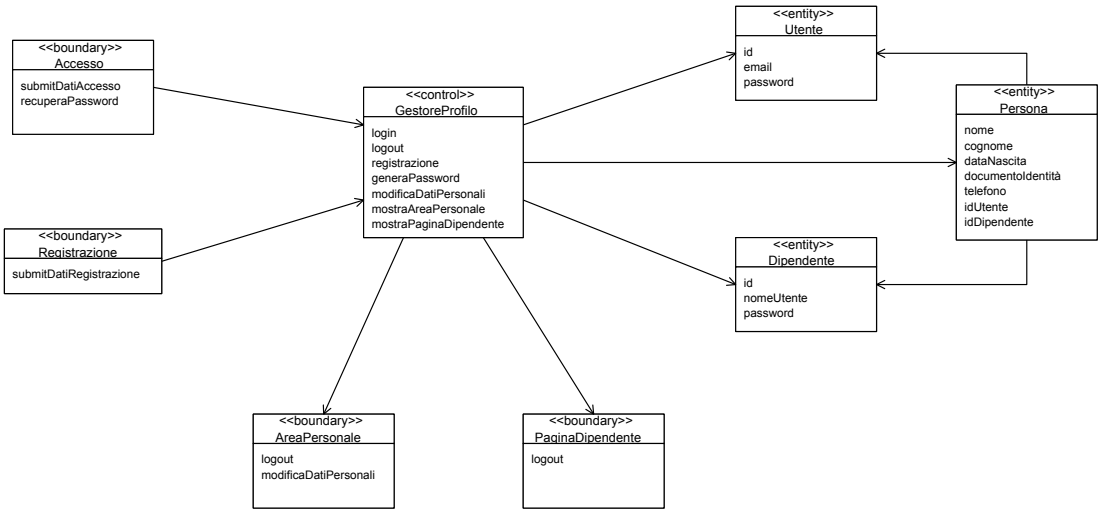

Figura 5: Gestore Profilo

In figura 5 vengono mostrate tutte le funzionalità implementate dal gestore dei profili. Da notare che viene utilizzato un metodo login comune sia per i dipendenti sia per gli utenti. L'unica differenza è rappresentata dalle classi boundary verso le quali si viene reindirizzati dopo il login: PaginaDipendente per i dipendenti e AreaPersonale per gli utenti.

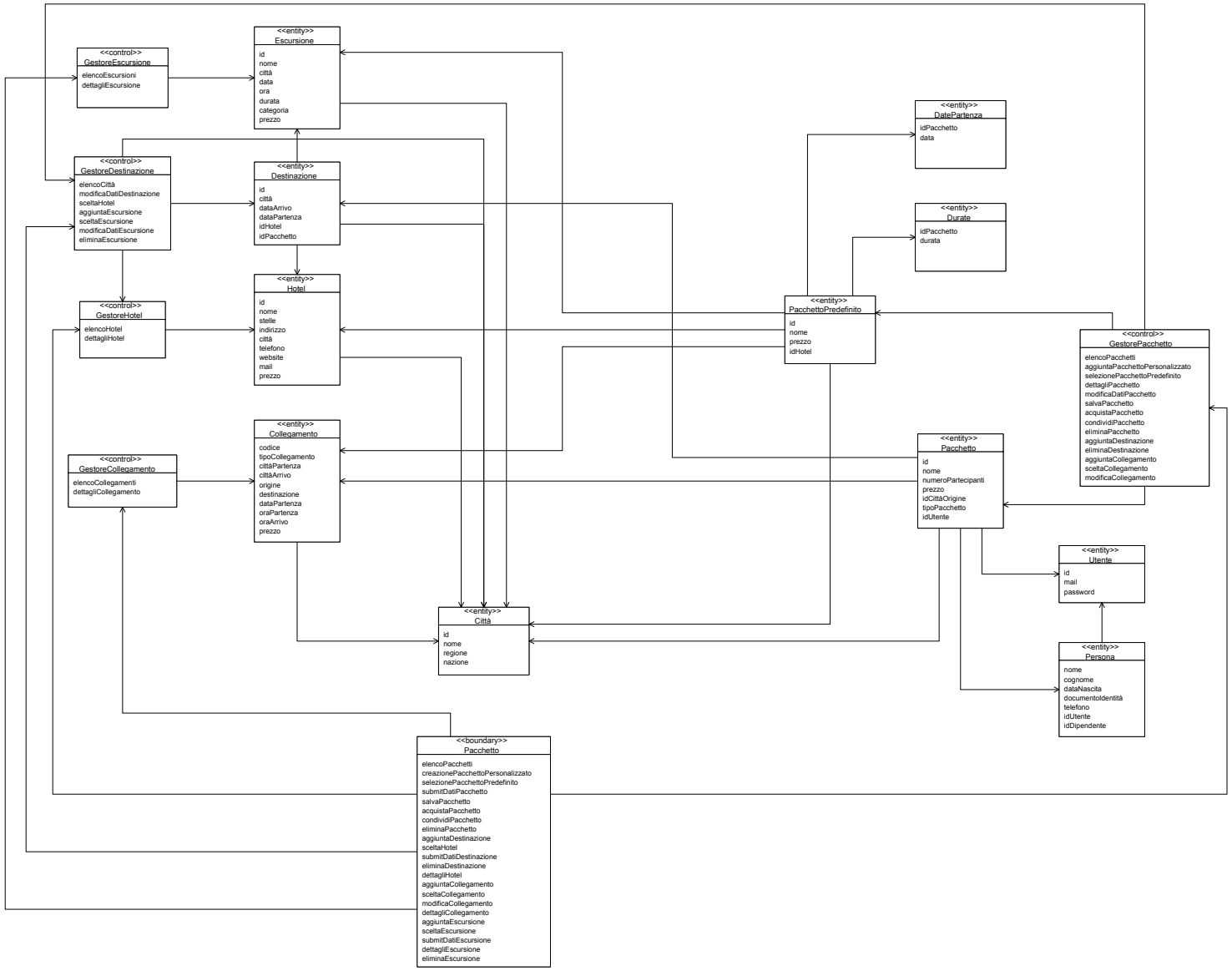

Figura 6: Gestione Pacchetto

In figura 6 vengono mostrati tutti i servizi relativi ai pacchetti lato utente. Tra questi sono presenti le funzioni per la creazione, personalizzazione, acquisto e condivisione dei pacchetti. I vari metodi "modificaDati" stanno a indicare che esistono diverse funzioni per modificare i singoli campi delle rispettive entità. Questo raggruppamento viene fatto per rendere più comprensibile il grafico.

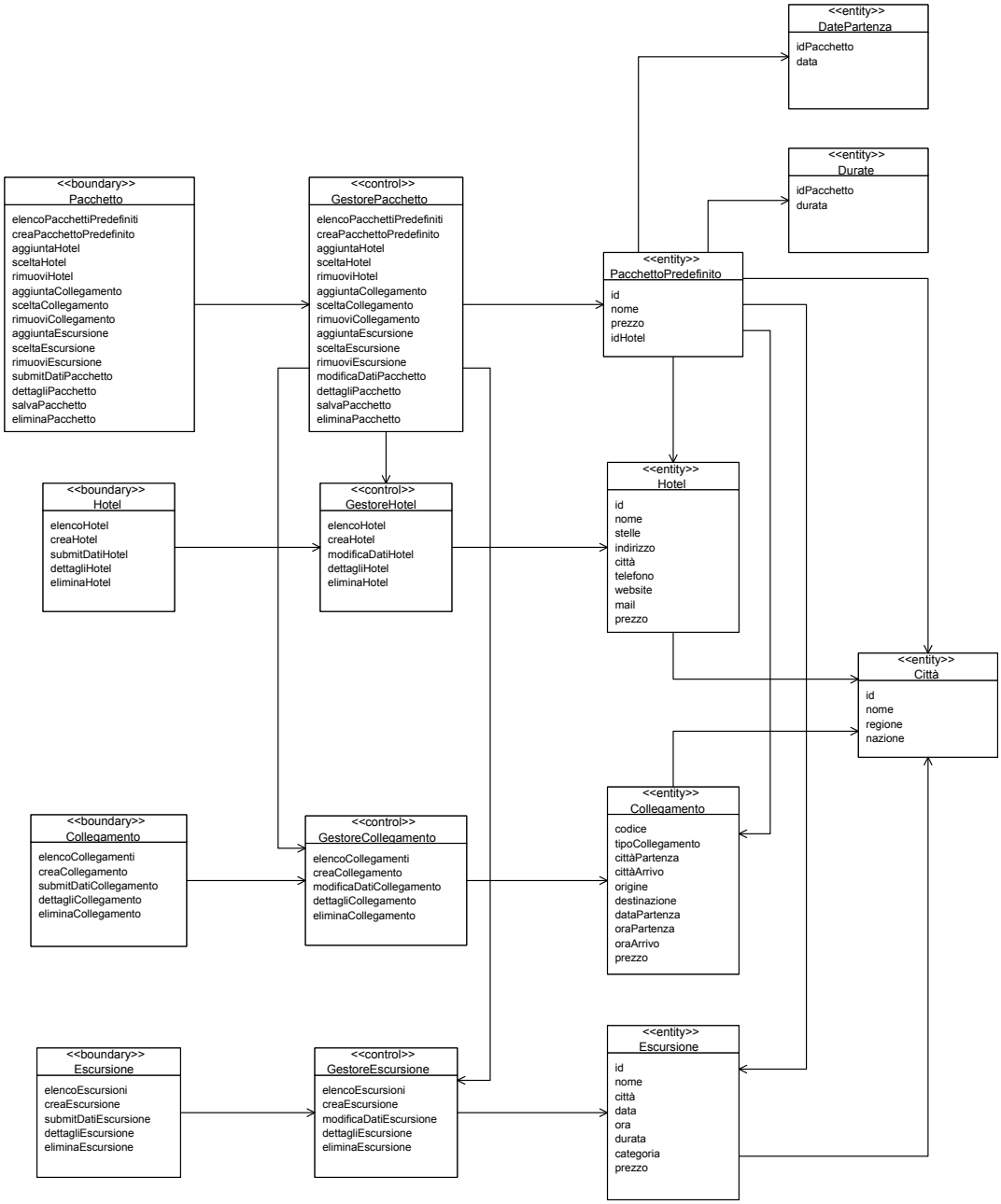

Figura 7: Dipendente

In figura 7 vengono mostrate le funzionalità dei dipendenti. Si può notare che il diagramma viene diviso in quattro categorie: PacchettoPredefinito, Hotel, Collegamento ed Escursione. Per ogni categoria è possibile creare un nuovo elemento, modificare o eliminare un elemento già esistente. Per tutte le categorie, tranne PacchettoPredefinto, è presente un metodo "aggiunta" che permette l'inserimento dell'elemento selezionato in un pacchetto predefinito.

4.1.2 Diagrammi di sequenza

In questa sezione vengono inseriti dei diagrammi di sequenza per mostrare visivamente l'ordine delle chiamate tra i componenti dei diagrammi BCE. Sono stati creati solo i diagrammi riguardanti le funzionalità principali del software.

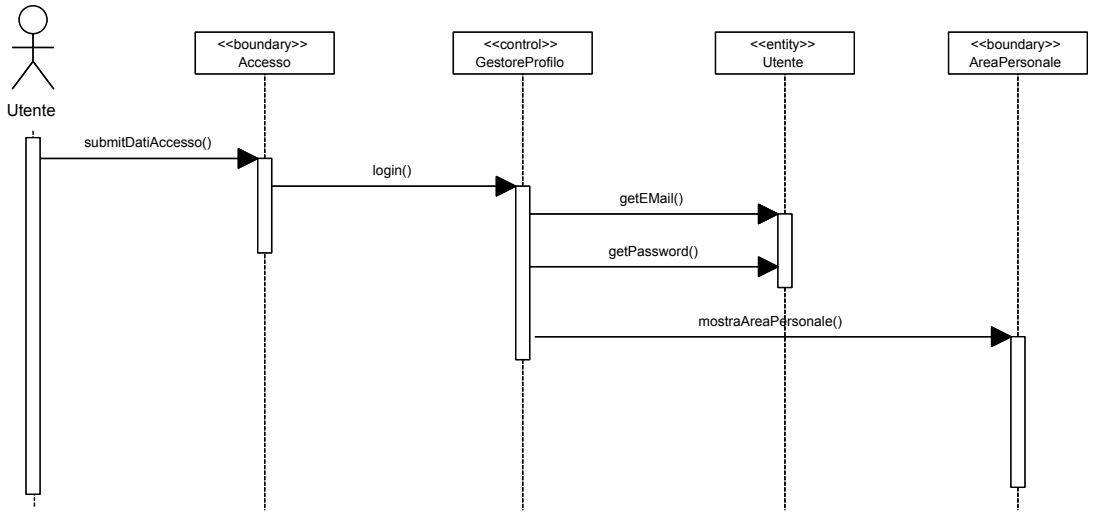

Figura 8: Login Utente

In figura 8 viene mostrata la sequenza di login di un utente. Questo diagramma è valido anche per il login dei dipendenti, cambia solo il metodo "mostraAreaPersonale" che diventa "mostraPagina-Dipendente".

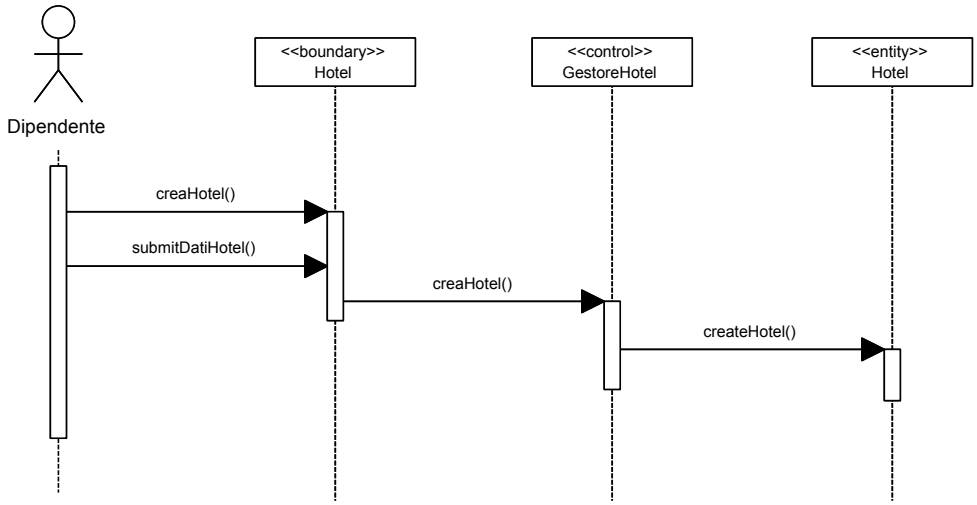

Figura 9: Creazione Nuovo Hotel

In figura 9 viene mostrata la fase di aggiunta di un nuovo hotel nel database. Questa procedura è valida anche per gli altri prodotti offerti da TravelDream.

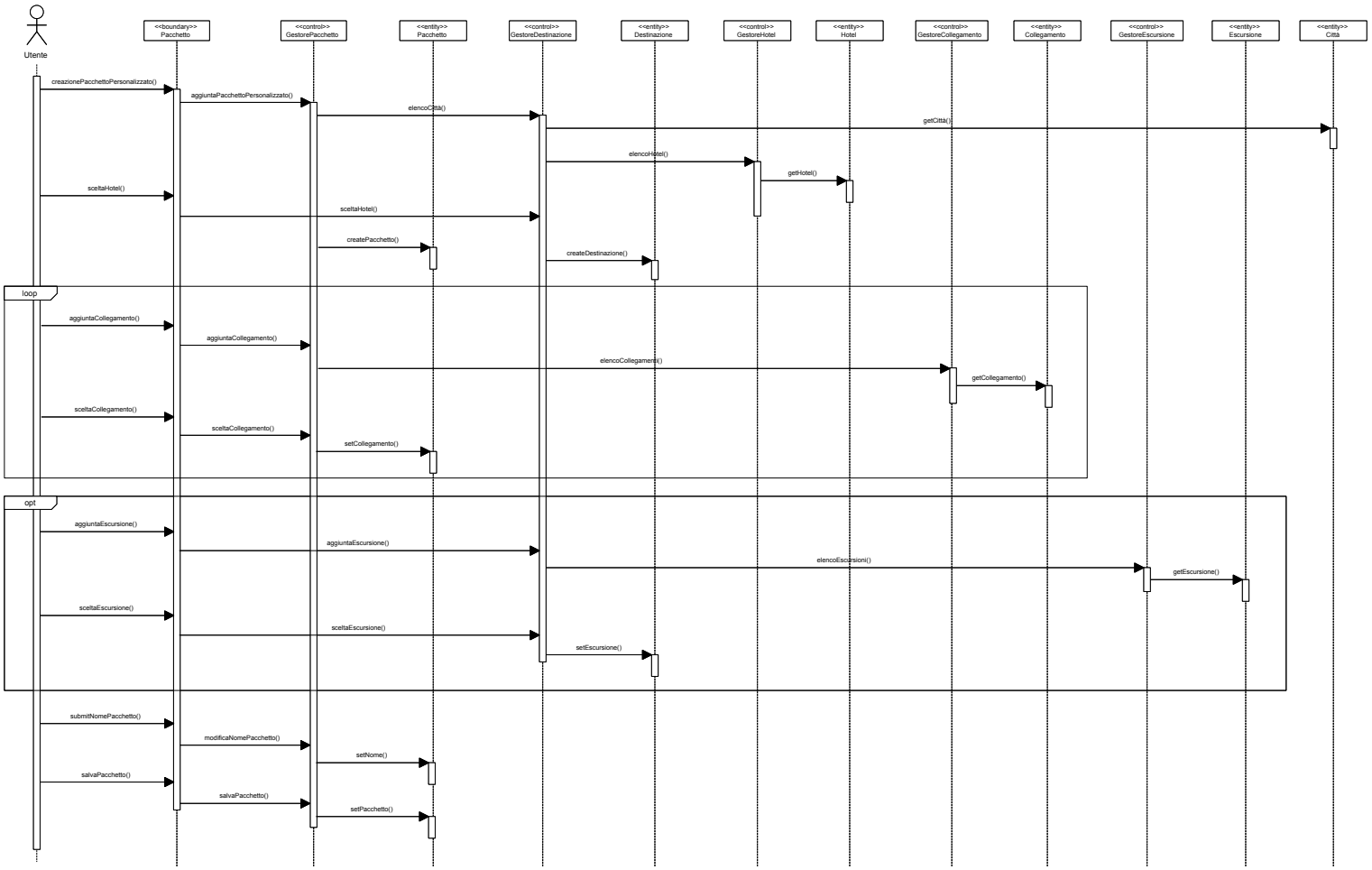

Figura 10: Creazione Pacchetto Personalizzato

In figura 10 viene mostrata la creazione di un nuovo pacchetto personalizzato. L'utente sceglie una città di destinazione dall'elenco delle città presenti nel database, compila i dati richiesti (data di partenza, data di ritorno, numero di partecipanti, città di partenza) e sceglie un hotel. Viene così creata la base del pacchetto. Ora l'utente deve aggiungere i collegamenti di andata e ritorno. Se interessato, l'utente può anche aggiungere delle escursioni. Completati i collegamenti e scelto un nome da dare al pacchetto, l'utente salva il pacchetto.

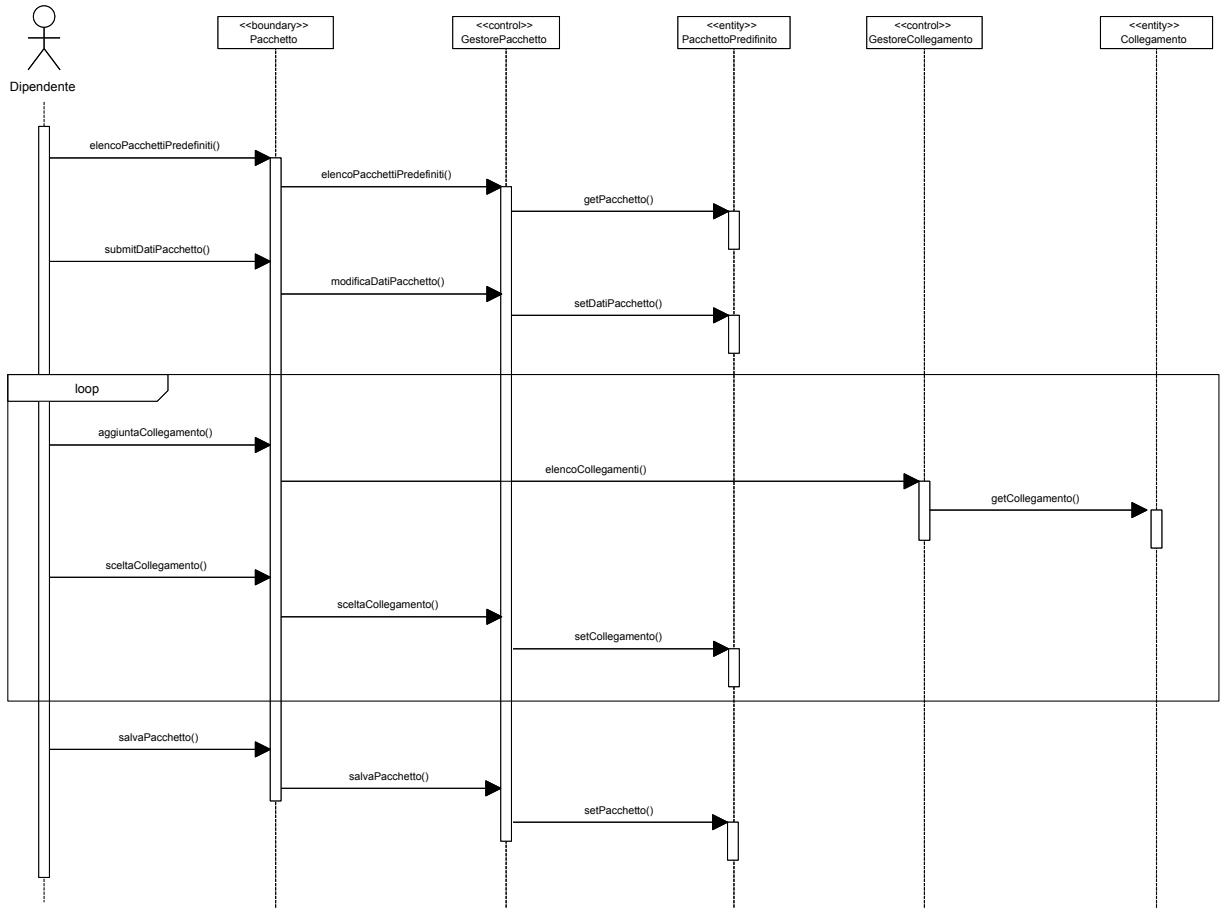

Figura 11: Modifica Pacchetto Predefinito

In figura 11 viene mostrata la fase di modifica di un pacchetto predefinito da parte di un dipendente. Il dipendente visualizza l'elenco dei pacchetti e seleziona quello che vuole modificare. Gli viene fornita la possibilità di modificare tutti i dati riguardanti il pacchetto in un'unica pagina. Nel diagramma viene mostrata anche la fase di aggiunta di un collegamento al pacchetto. Terminate le modifiche il dipendente salva il pacchetto.

4.2 Module View

Il seguente diagramma, fortemente correlato con l'architettura utilizzata, mostra come comunicano tra di loro i vari livelli nei quali è diviso il software.

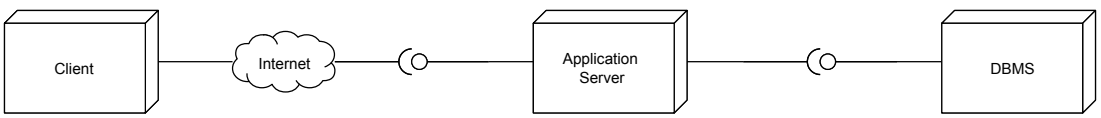

Figura 12: Module View

4.3 Deployment view

Il seguente diagramma mostra l'allocazione fisica dei diversi componenti del software. Vengono messi in evidenza anche i protocolli utilizzati per la comunicazione tra i diversi moduli.

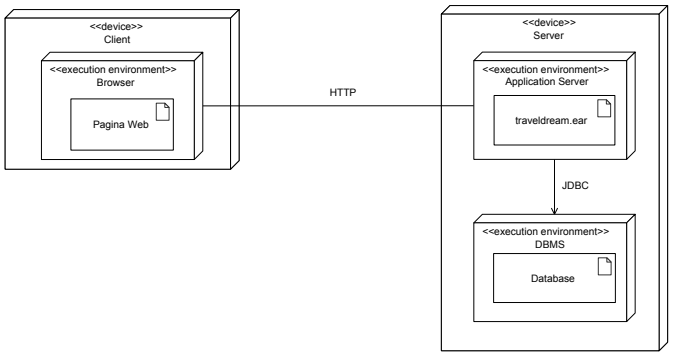

Figura 13: Deployment View

4.4 Runtime view

Il seguente diagramma è molto simile alla deployment view, ma a differenza di quest'ultima mette in evidenza che più client possono interagire contemporaneamente col server. Nel diagramma viene visualizzato un client per attore: nella realtà possono esistere più client di quelli rappresentati.

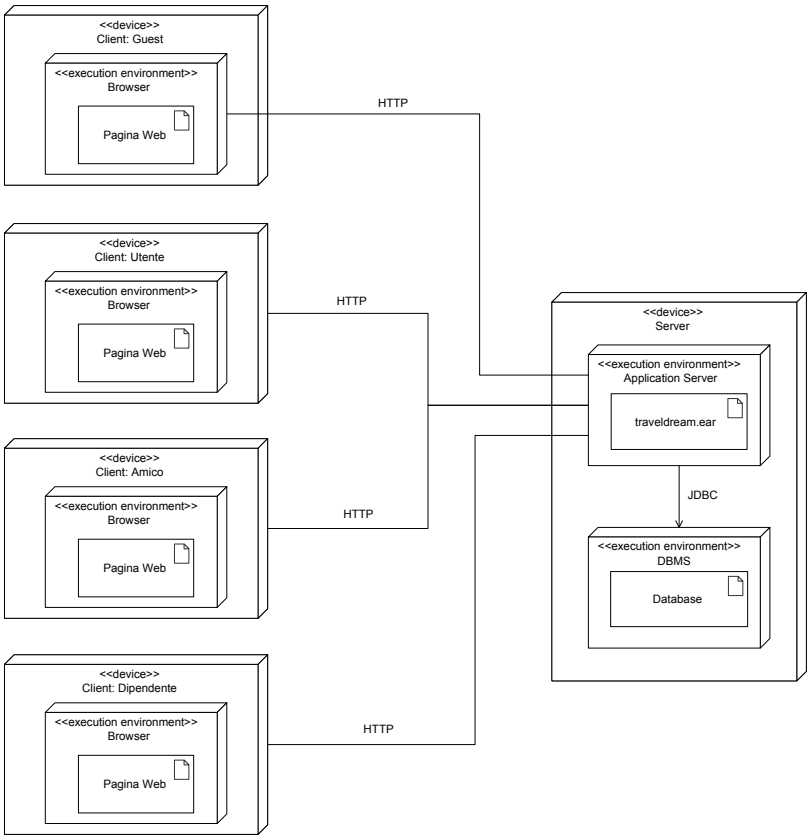

Figura 14: Runtime View

4.5 Modelli di navigazione

Per mostrare come avviene la navigazione all'interno dell'applicazione vengono utilizzati dei diagrammi UX così strutturati:

- screen: rappresenta la schermata che viene presentata all'utente. I metodi corrispondono alle azioni che gli utenti possono compiere su una schermata. Le frecce indicano il il passaggio da una schermata ad un'altra e sopra ogni freccia viene indicato il metodo col quale è possibile effettuare la rispettiva azione. Ogni screen ha il metodo navigateTo() che viene invocato per istanziare la schermata.
- screen compartment: rappresenta una parte della schermata. Non ha vita propria ma è parte di uno screen, al quale è collegato mediante composizione. Nei successivi diagrammi viene utilizzato per le schermate di modifica dei pacchetti, in quanto non verrà realizzata una apposita pagina ma le modifiche verranno effettuate nella stessa schermata.
- input form: rappresenta un modulo di inserimento. L'utente interagisce con il sistema inserendo le informazioni richieste. Non possiede metodi ma solo gli attributi che l'utente può inserire. Viene associato ad una schermata mediante composizione per gestire meglio la navigabilità. In uscita dagli input form sono presenti due frecce: una "OK" nel caso l'operazione di invio dei dati vada a buon fine ed una "errore" che mostra la pagina verso la quale si viene rimandati in caso di errore. In quest'ultima pagina verrà mostrato un messaggio che spiega all'utente per quale motivo l'operazione non si è conclusa correttamente.
- $: questo simbolo viene aggiunto alle pagine verso le quali si può accedere da qualsiasi pagina, per esempio tramite menù.

Nella realizzazione dei diagrammi ci si è concentrati sulla parte riguardante le funzioni principali del sistema. Per questo motivo vengono omesse le pagine statiche. I diagrammi vengono divisi per funzionalità o attore per creare meno confusione e rendere i diagrammi più leggibili.

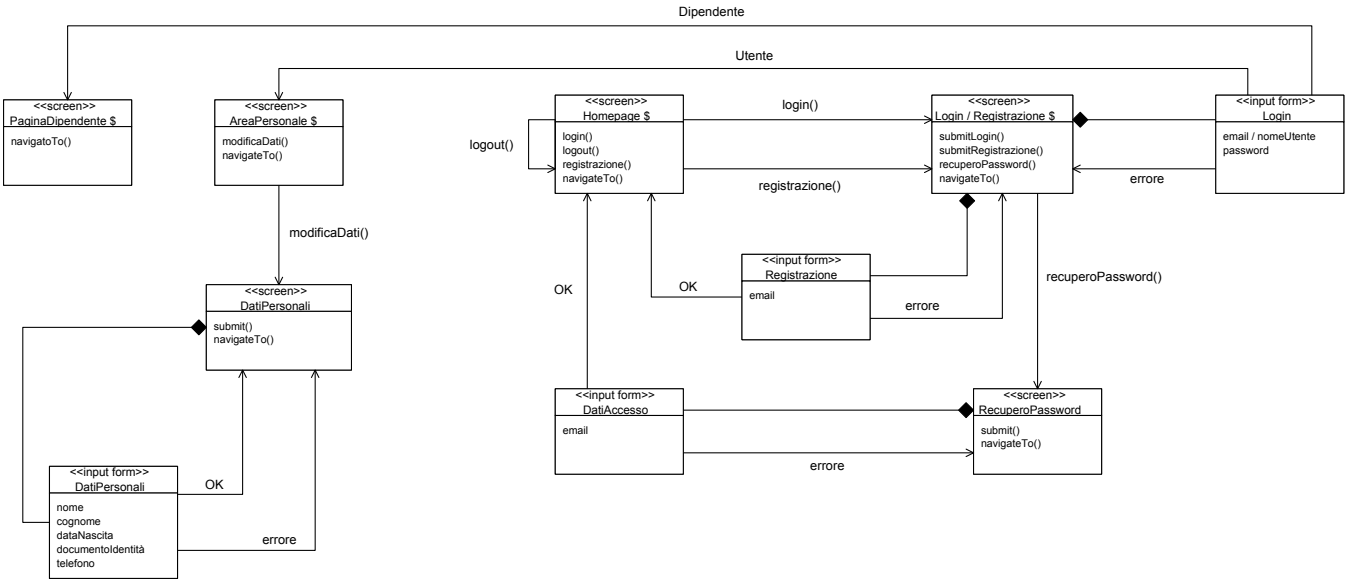

Figura 15: Gestione Profilo

In figura 15 vengono mostrate le schermate che hanno attinenza con la gestione del profilo degli utenti. Si noti che, come nei diagrammi di analisi, viene fornita un'unica pagina di login utilizzabile sia dagli utenti sia dai dipendenti. Se il login ha successo il sistema reindirizza verso la rispettiva pagina personale (come indicato dalle due frecce uscenti dall'input form "Login").

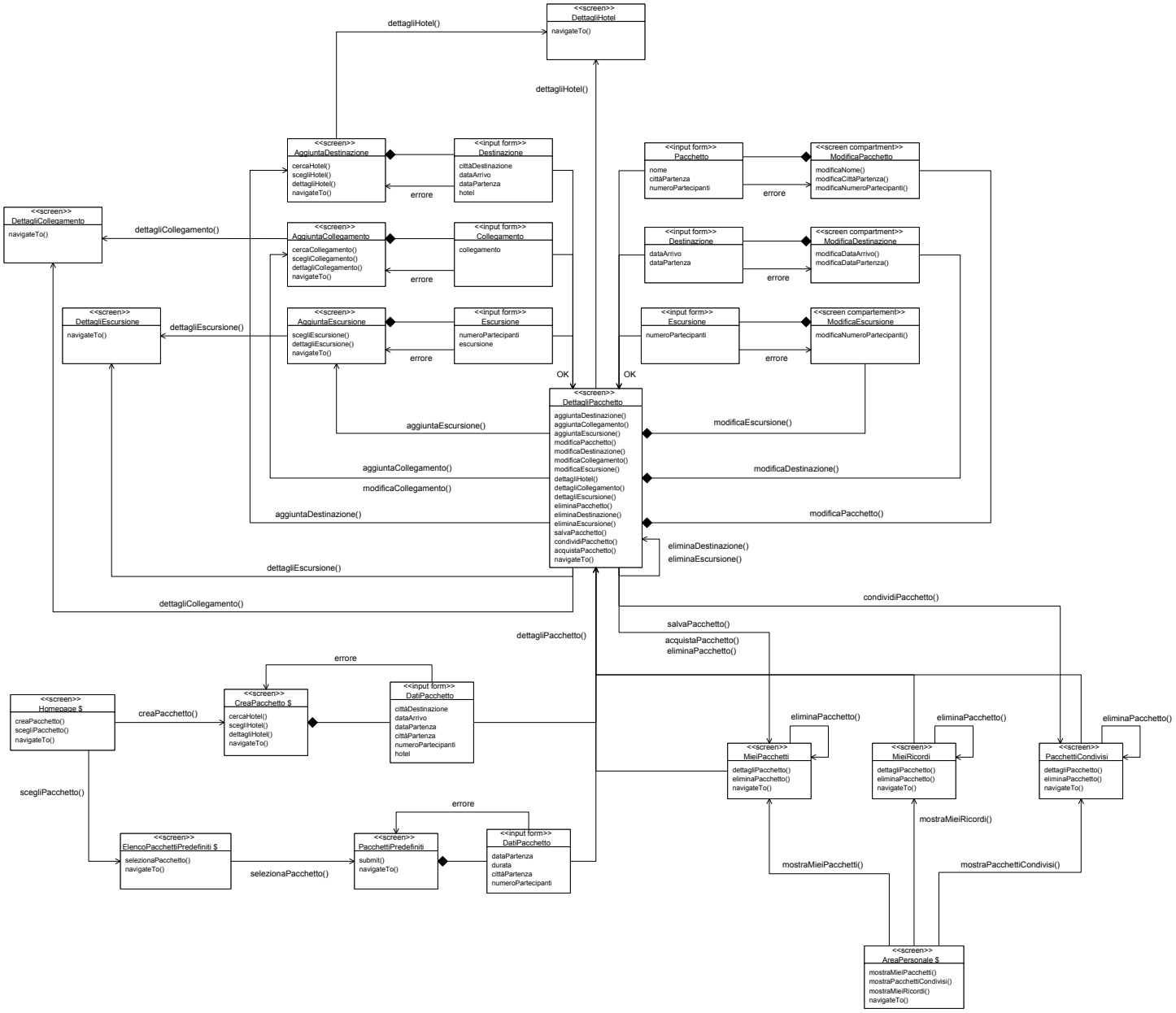

Figura 16: Gestione Pacchetto

In figura 16 vengono mostrate le schermate che riguardano tutte le operazioni che possono essere effettuate sui pacchetti lato utente. Nei compartimenti riguardanti la modifica del pacchetto viene collegato un singolo input form ma vengono inseriti più metodi. Questo per sottolineare che l'utente può modificare un solo campo alla volta. Nel caso dei pacchetti predefiniti alcune funzioni di modifica vengono disabilitate (es.: aggiunta destinazione).

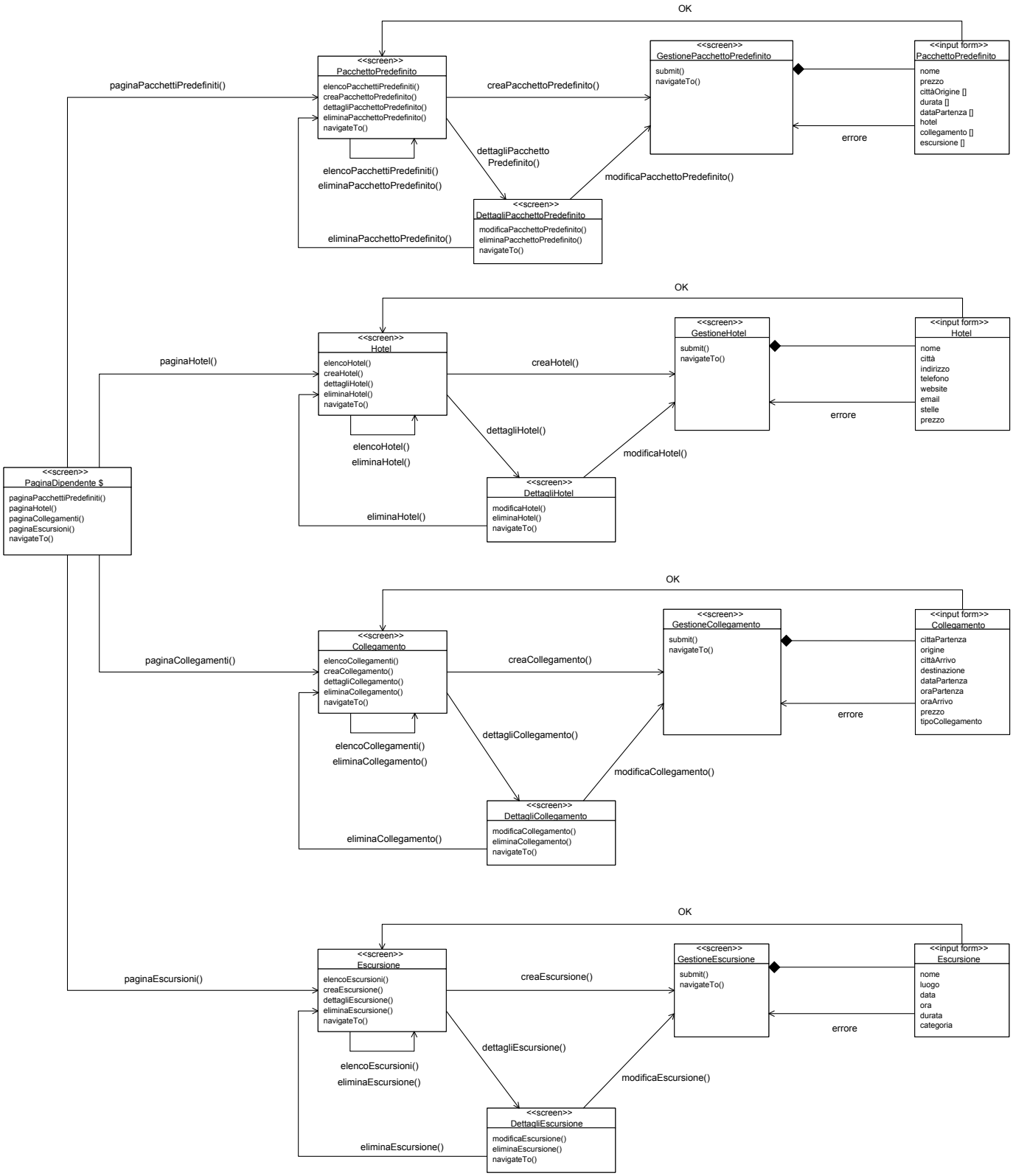

Figura 17: Dipendente

In figura 17 viene mostrato il funzionamento del back end dell'applicazione, cioè tutte le operazioni che i dipendenti possono effettuare. Si noti che alcuni attributi negli input form sono seguiti da due partentesi quadre (' [] ') per indicare che possono essere inseriti più valori.