POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

DESIGN DOCUMENT

TRAVEL DREAM

Progetto di ingegneria del software II

Sara Marchesini, Sebastiano Mariani, Riccardo Mastellone

Contents

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è definire un'architettura generale per il sistema da implementare, che dovrà poi essere raffinata in fase di implementazione.

Questa struttura sarà basata sulle specifiche descritte nel documento di analisi dei requisiti.

Si rivolge principalmente al team di svilupatori incaricati di implementare e mantere il software.

1.2 Referenze

- Documento di analisi dei requisiti TravelDream
- Specifiche di progetto "Progetto AA 2013-2014(TravelDream)"

1.3 Definizioni e abbreviazioni

1.3.1 Definizioni

Parola	Definizione
Cliente	Utente registrato al sito
Dipendente	Impiegato TravelDream con credenziali per l'accesso da dipendente al sito
Amministratore	Capo della TravelDream con credenziali uniche per la gestione dei dipendenti
Prodotto base	Volo/hotel/attività secondarie
Pacchetto	Vacanza organizzata comprendente volo, hotel e attività secondarie

1.3.2 Abbreviazioni

Abbreviazione	Definizione
DBMS	Database Management System
JEE	Java Enterprise Edition
API	Application Programming Interface
E-R	Entità-Relazione
EJB	Enterprise Java Bean
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JDBC	Java Database Connectivity
UML	Unified Modeling Language
UX	User eXperience
JSF	Java Servlet Faces
MVC	Model View Controller
JPA	Java Persistence API
SMTP	Simple Mail Transfer Protocol
XHTML	eXtensible Hypertext Markup Language
POJO	Plain Old Java Object
BCE	Boundari Control Entity

1.4 Panoramica

Il documento è così suddiviso

1. Introduzione: breve introduzione contenente lo scopo del documento e la definizione di alcuni termini e abbreviazioni.
2. Descrizione generale: descrizione generale delle tecnologie scelte e delle varie suddivisioni in layer e tier ritenute opportune.
3. Progetto dei dati: descrizione tramite modelli formali(modello E/R, odello logico) della base di dati
4. Progetto del business tier: descrizione della logica applicativa e delle integrazioni tra i componenti mediante diagrammi BCE
5. Progetto del client: descrizione della navigazione tra le varie pagine mediante diagramma UX

2 Descrizione generale

Il sistema TravelDream è stato ideato per aiutare da un lato i dipendenti dell'azienda nella gestione del catalogo dei pacchetti vacanza e nella vendita degli stessi, e dall'altro i clienti nella consultazione del catalogo e nella creazione e acquisto di pacchetti.

L'interfacciamento al sistema avviene esclusivamente via web per rendere il servizio disponibile al maggior numero di persone, e fruibile in maniera estremamente semplice.

Per raggiungere questo scopo il software mette a disposizione dei dipendenti funzioni intuitive per la gestione dei pacchetti come ricerca, cancellazione e aggiunta, mostrando a schermo tutti i campi necessari all'esecuzione della funzione. inoltre fornisce ai clienti strumenti di ricerca arricchiti di filtri per destinazione, data di partenza e tipo di vacanza, per sfogliare il catalogo in maniera molto semplice, e strumenti per la creazione di pacchetti personalizzati, con possibilità di invito di persone esterne.

Gli utenti che vogliono usufruire di tale dservizio devono essere obbligatoriamente registrati.

2.1 Scelte Tecnologiche

Di seguito sono riportate le scelte tecnologiche, effettuate tenendo conto dei vincoli imposti dall'utilizzo della piattaforma JEE, da rispettare nello sviluppo del software.

Tali scelte condizionano fortemente la progettazione del sistema riportata in questo documento.

- Il sistema è basato sulla piattaforma JEE 7. In particolare verranno utilizzate alcune specifiche:
  - EJB 3.2 per lo sviluppo della logica applicativa
  - JSF 2.2 per lo sviluppo dell'applicazione web
  - JavaMail API per interfacciare il sistema con il servizi di posta elettronica
- Come application server viene utilizzato Glassfish 4.0
- come DBMS viene utilizzato MySQL 5.6.14, tale scelta non vincola la struttura del codice dal momento che l'interfacciamento al database avviene tramite JPA, infatti cambiando il DBMS con un altro compatibile con l'application server utilizzato, il codice non varia

Come ambiente di sviluppo è consigliato l'utilizzo di Eclipse (versione Kepler) con i plugin relativi alle componenti sopra specificate.

Per facilitare la collaborzione tra i vari componenti del team di sviluppo è stata attivata una repository git presso google code a questo indirizzo https://code.google.com/p/traveldream-mariani-marchesini-mastellone/

2.2 Descrizione dell'architettura

Si è scelto di utilizzare una architettura multi-tier suddivisa in 4 livelli logici (tier):

- **Client tier:**
  - livello da cui gli utenti accedono all'applicazione tramite un comune browser web. Comunica con il web tier attraverso il protocollo Http inviando al server i dati inseriti dall'utente e si occupa della presentazione delle pagine una volta ricevuto il file XHtml dal server. Inoltre può eseguire codice Javascript che realizza semplici controlli sulla sintassi dei dati immessi (questi controlli andranno effettuati anche lato server per questioni di sicurezza, servono principalmente per ridurre il traffico tra client e server causato da errori di distrazione (es: mancanza della @ nel campo mail)), oppure che realizza effetti grafici all'interno della pagina e nella transizione tra le varie schermate attraverso l'ausilio delle librerie jQuery.
- **Web tier:**

- Questo livello riceve le richieste del client e in base ai dati ricevuti e al tipo di richiesta(GET, POST ecc..),grazie alla tecnologia JSF, interagisce con il business tier, genera la pagina approriata e la spedisce al client richidente.

- **Business tier:**
  - Questo livello incapsula la logica applicativa dell'intero sistema e comunica direttamente con il database. I dati dell'applicazione, ovvero il modello all'interno del pattern MVC, sono rappresentati da POJO(entity beans) garantiti persistenti grazie alla tecnologia JPA. Il controller invece è realizzato mediante componenti EJB che implementano la logica applicativa. Inoltre interagisce con un server di post elettronica, utilizzando il protocollo SMTP attraverso le API javaMail, che permette a clienti di inviare inviti e liste desderi.

- **Data tier:**
  - È costituito da un DBMS che realizza la persistenza dei dati e li mette a disposizione dell'application server. Quest'ultimo comunica con il DBMS grazie al driver JDBC.

In questo tipo di arcitettura ogni tier comunica solamente con quello immediatamente adiacente, rendendo di fatto l'applicazione modulare.

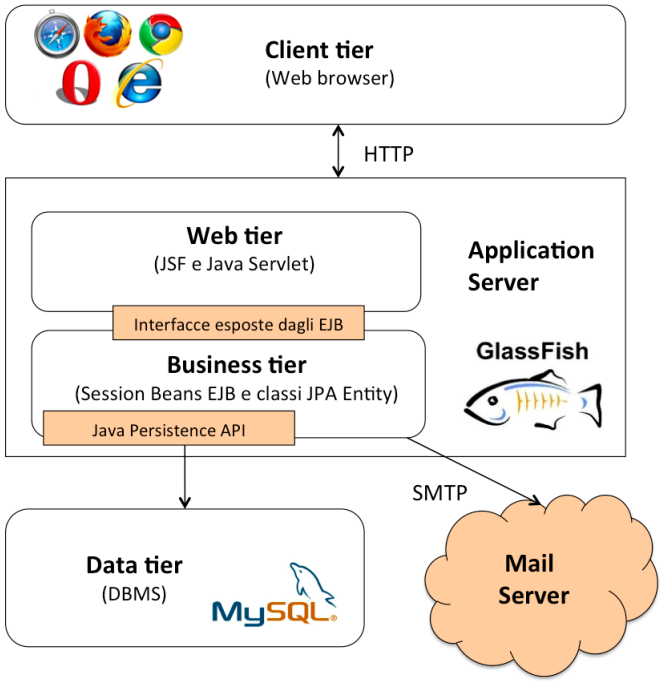

Figure 1: Architettura: tier logici

A livello fisico i 4 tier logici sono suddivisi in 2 layer:

- **Client:**
  - È il terminale da cui l'utente accede all'applicazione, e incapsula il solo client tier.

- **Server:**

– È il server fisico vero e proprio su è deployato l'application server e in questa prima implementazione anche il DBMS. Questo layer incapsula il web tier, il business tier e il data tier.

Il server mail è esterno al sistema e si appoggia a servizi esterni.

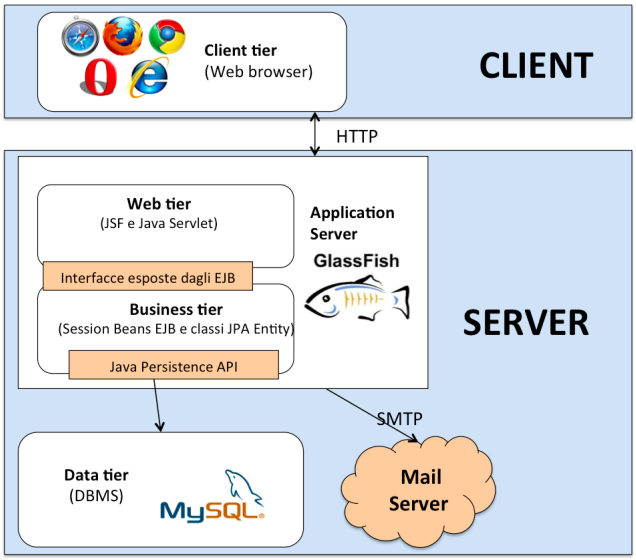

Figure 2: Achitettura: layer fisici

3 Progetto dei dati

La struttura dei dati è basata sul modello relazionale e da implementare su un server MySQL

3.1 Progettazione concettuale

Sono state individuate le entità e le relazioni di interesse per l'applicazione sulla base del diagramma delle classi presente nel documento di analisi dei requisiti.

Di seguito è riportato il diagramma E-R generato da questa analisi.

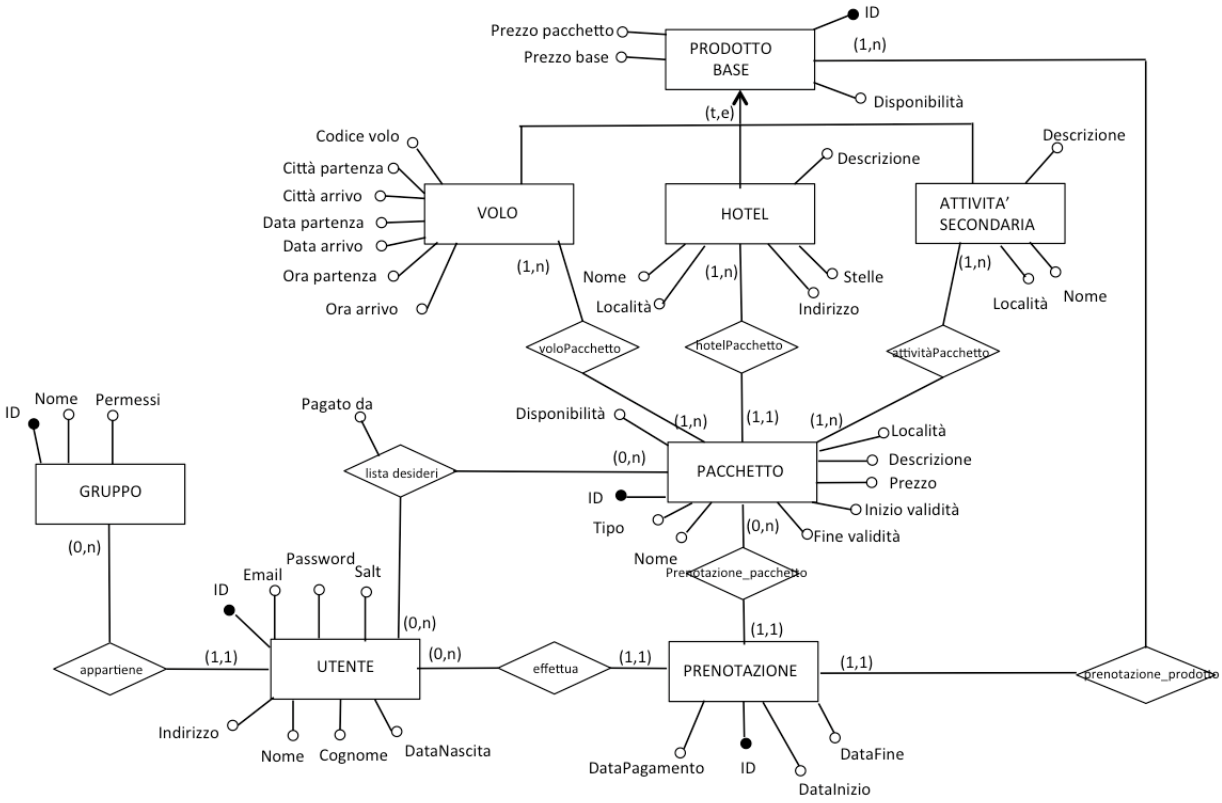

Figure 3: Diagramma E-R

Le entità e le relazioni sono:

- **UTENTE:** rappresenta un utente registrato al sistema, è identificato da un ID univoco autoincrementale, e ha come attributi i dati anagrafici(nome, cognome, indirizzo, data di nascita) e le credenziali di accesso(nome utente coincidente con la mail di registrazione e password); inoltre appare l'attributo salt per rinforzare la sicurezza delle password.
  - APPARTIENE: collega ogni utente a uno e un solo gruppo, permettendo così la distinzione tra clienti e dipendenti e amministratore.
  - EFFETTUA: presente solo negli utenti appartenenti al gruppo cliente; collega il cliente ai pacchetti o ai prodotti base prenotati

  - LISTA DESIDERI: presente solo negli utenti apparteneti al gruppo cliente; specifica per ogni cliente quali pacchetti sono stati aggiunti alla propria lista desideri e attraverso l'attributo "pagato da" quali sono stati pagati e da chi

- **GRUPPO:** entità identificata da un ID univoco autoincremetale, e presenta l'attributo nome di tipo enuerazione che indica il gruppo di appartenenza e quindi di privilegio di un utente. Ammette i seguenti valori: amministratore, con possibilità di scrittura sulla tabella UTENTI per aggiungere o rimuovere dipendenti; dipendete, con possibilità di scrittura sulla tabella PACCHETTO per aggiungere, rimuovere o editare pacchetti; e cliente con permessi di sola lettura sulla tabella PACCHETTO

- **PACCHETTO:** entità identificata da un ID univoco autoincrementale, e caratterizzata da un tipo, nome, prezzo, località, data inizio e fine validità e descrizione. Inoltre presenta anche un campo disponibilità di tipo booleano che verrà settato a False appena uno dei componenti base che lo compongono non sarà più disponibile. Presenta le seguenti relazioni:

  - COMPRENDE_VOLI: collega al pacchetto i voli di andata e di ritorno compresi proposti nell'arco di tempo di validità del pacchetto
  - COMPRENDE_HOTEL: collega al pacchetto uno e un solo hotel convenzionato nella vacanza
  - COMPRENDE_ATTIVITA': collega al pacchetto una o più attività convenzionata nella vacanza

- **PRODOTTO BASE:** entità che specifica le caratteristiche comuni a tutti i prodotti base(volo, hotel, attività secondarie). è identificato da un ID univoco autoincrementale, e caratterizzata da un prezzo base(prezzo nel caso di personalizzazione e creazione del pacchetto), prezzo pacchetto(prezzo scontato che si avrà solo all'interno dell'acquisto dell'intero pacchetto senza modifiche), disponibilità(numero intero decrementato ad ogni acquisto del componente). È generalizzazione totale ed esclusiva di VOLO, HOTEL, ATTIVITA' SECONDARIA.

- **VOLO:** entità caratterizzata da codice volo(codice IATA + id) , città partenza, città arrivo, data partenza, data arrivo, ora partenza, ora arrivo. Un volo è associato ad uno ed un solo pacchetto tramite la relazione COMPRENDE_VOLI

- **HOTEL:** entità caratterizzata dalla località dove situato l'hotel, l'indirizzo, il nome, il numero di stelle ed una breve descrizione. è associato tramite la relazione COMPRENDE_HOTEL a uno o più pacchetti, in quanto lo stesso hotel può essere parte di pacchetti diversi.

- **ATTIVITA' SECONDARIA:** entità caratterizzata dalla località dove si svolge, dal nome, e da una breve descrizione. È associata tramite la relazione COMPRENDE_ATTIVITA' ad un solo pacchetto.

- **PRENOTAZIONE:** entità identificata da un ID univoco autoincrementale, e caraterizzata dalle seguenti relazioni:

  - EFFETTUA: relazione che specifica l'ID del cliente che ha effettuato la prenotazione
  - PRENOTAZIONE_PACCHETTO: relazione che specifica, nel caso di prenotazioni di un pacchetto, l'ID di quest'ultimo.
  - PRENOTAZIONE_PRODOTTO: relazione che specifica, nel caso di prenotazioni di uno o più prodotti base, l'ID di questi.

3.2 Progettazione logica

Lo schema concettuale proposto al punto precedente viene tradotto nel conseguente schema logico, che corrisponde alla reale struttura del database relazionale.

Lo schema del database risultante è il seguente:

- UTENTE (<u>id</u>, nome, cognome, dataNascita, indirizzo, email, password, salt, id_gruppo)

- GRUPPO (id, nome, permessi)
- PACCHETTO (id, tipo, nome, località, descrizione, prezzo, inizioValidità, fineValidità, id_hotel)
- VOLO(id, disponibilità, prezzoPacchetto, prezzoBase, codiceVolo, cittaPartenza, cittaArrivo, dataPartenza, dataArrivo, oraPartenza, oraArrivo, id_pacchetto)
- HOTEL(id, disponibilità, prezzoPacchetto, prezzoBase, nome, località, indirizzo, stelle, descrizione)
- ATTIVITASECONDARIA(id, disponibilità, prezzoPacchetto, prezzoBase, nome, località, descrizione, id_pacchetto)
- PRENOTAZIONE (id, id_utente, id_pacchetto*, id_volo*, id_hotel*, id_attivitaSecondaria*, dataPagamento,dataInizio, dataFine)
- LISTADESIDERI(id, id_pacchetto, id_utente, pagatoDa*)
- VOLOPACCHETTO(id, id_volo, id_pacchetto)
- ATTIVITAPACCHETTO(id, id_attività, id_pacchetto)

Lo schema di traduzione adottato è quello standard.

Le associazioni uno-a-molti sono state tradotte inserendo, all'interno della tabella che partecipa all'associazione dal lato "molti", un'attributo contenente la chiave primaria della tabella cui è collegata, dichiarando gli opportuni vincoli di chiave esterna (ad esempio: id_gruppo in UTENTE, id_hotel in PACCHETTO, id_pacchetto in VOLO ecc).

Le relazioni molti-a-molti sono state tradotte invece con una tabella avente come attributi le chiavi delle entità coinvolte (esempio: VOLOPACCHETTO ha come campi id_volo e id_pacchetto), più gli eventuali attributi della relazione(esempio: LISTADESIDERI oltre ad avere come attributi le chiavi esterne id_pacchetto e id_utente, ha anche pagatoDa).

Per quanto riguarda le generalizzazioni, l'unico caso presente nel diagramma E-R è stato tradotto inserendo nello schema delle entità figlie i campi dell' entità padre; data la diversità e la varietà dei campi delle entità figlie, è sembrato opportuno trattarle come tabelle diverse.

Infine, gli attributi caratterizzati dal simbolo * possono assumere valore nullo.

Il grafo completo del progetto logico è riportato nella figura sottostante. Nel diagramma sono rappresentati anche i vincoli di chiave esterna tra le tabelle (con una notazione simile a quella utilizzata per le associazioni tra classi in UML) e i tipi di dato utilizzati per ciascun campo.

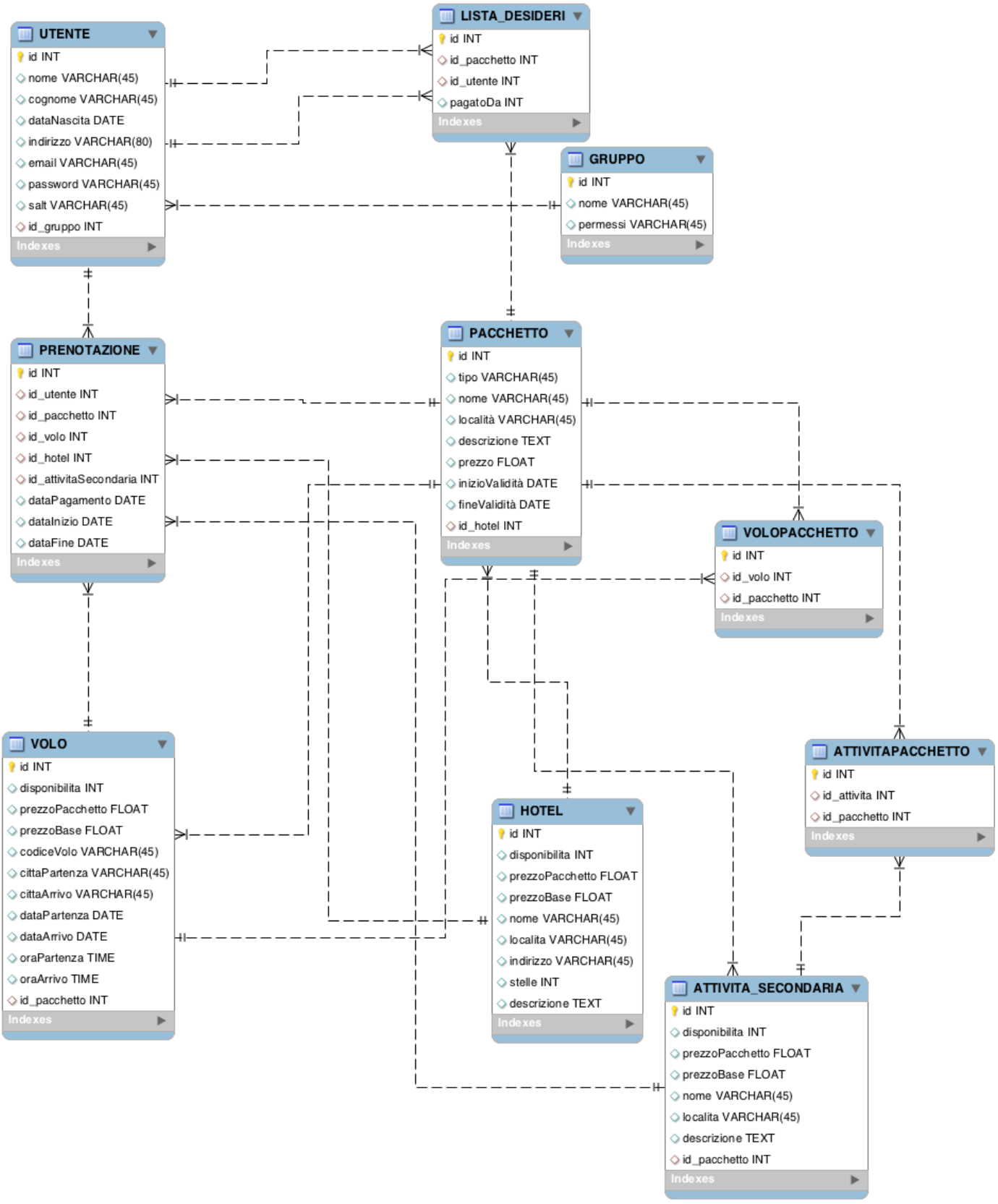

Figure 4: Modello logico del database

4 Progetto del business tier

Una volta individuati i dati di interesse per il dominio applicativo è stata progettata l'applicazione secondo l'architettura introdotta nella prima sezione e sono state individuate le componenti software per implementare la logica applicativa.

Si è scelto di basarsi su un architettura Model-View-Controller, architettura modulare e altamente scalabile che presenta i vantaggi di dividere in modo netto la logica applicativa, i dati e l'interfaccia utente.

4.1 Modello (Entity beans)

la struttura dell'insieme delle classi JPA entity(entity beans) astraggono il modello relazionale dei dati basato sul DBMS MySQL. Questa astrazione semplifica notevolmente la gestione dei dati in un linguaggio orientato agli oggetti quale è JEE.

4.2 Diagrammi Boundary-Control-Entity

Si è scelto di rappresentare l'interazione e la suddivisione tra i vari componenti mediante diagramma BCE composto da 3 elementi fondamentali:

- <<boundary>>: insieme di funzionalità offerte all'utente coerenti con i modelli di navigazione descritti nella sezione 5; rappresentano l'interfaccia utente dell'applicazione.
- <<control>>: controller che implementano la logica applicativa dell'applicazione; fanno da mediatori tra interfaccia utente e dati
- <<entity>>: astrazione del modello relazionale dei dati

Nella lettura dei diagrammi assumere che componenti con lo stesso nome in scenari diversi come lo stesso lo stesso coponente.

Per non appesantire la struttura sono stati omessi i metodi getter e setter.

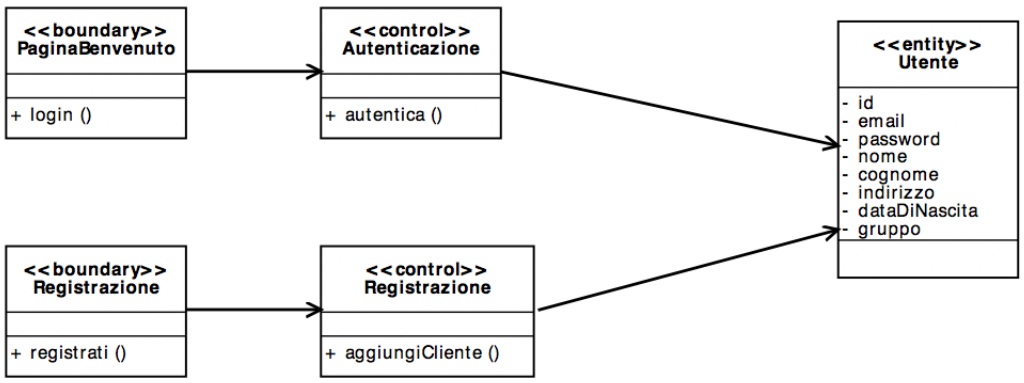

Figure 5: BCE: Utente non registrato o loggato

Diagramma rappresentante i metodi di login per un utente già registrato e i metodi di registrazioni per quelli non registrati.

Sia il control Autenticazione, sia quello Registrazione dovranno implementare tutti gli aspetti legati alla sanitizzazione dell'input e della validazione dell'utente una volta loggato.

Agiscono entrambi sull'entity utente, il primo in lettura mentre il secondo in scrittura, creando l'apposita tupla del nuovo cliente tramite il metodo aggiungiCliente().

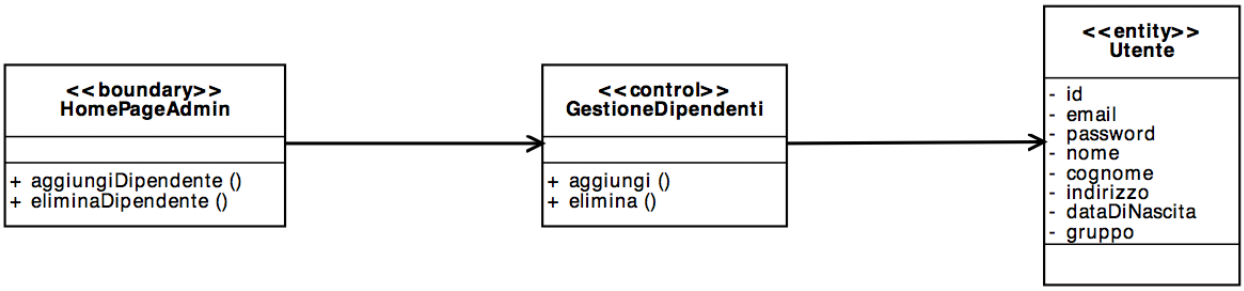

Figure 6: BCE: Amministratore

Diagramma rappresentante i metodi rservati all'amministratore.

L'amministratore tramite i metodi aggiungi() ed elimina() può acedere in scrittura all'entità entity aggiungendo o eliminando nuovi dipendenti.

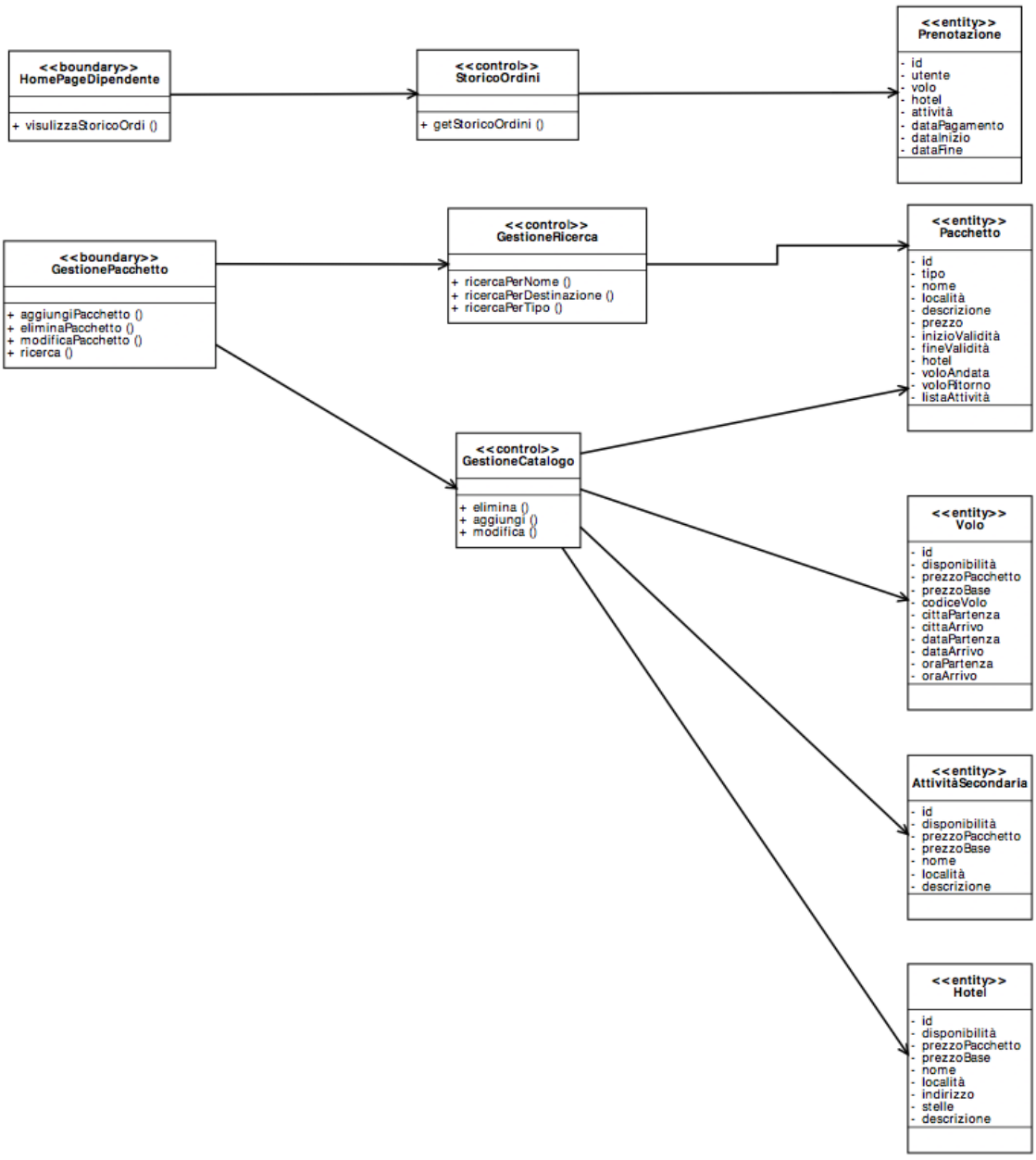

Figure 7: BCE: Dipendente

Diagramma rappresentante i metodi messi a disposizione al dipendente.
A partire dalla HomePageDipendente può eseguire il seguente metodo:

- visualizzaStoricoOrdini(): chiama il metodo getStoricoOrdini presente nel control StoricoOrdini il quale si limita, agendo sull'entità prenotazione, al reperimento di tutte le tuple presenti nella tabella.

mentre dalla pagina GestionePacchetto può eseguire i seguenti metodi:

- aggiungiPacchetto(), eliminaPacchetto() e modificaPacchetto(), i quai invocano i rispettivi metodi aggiungi(), modifica() ed elimina() presenti nel controller GestioneCatalogo. Questo controller accede in

scrittura alle entità pacchetto, volo, hotel e attivitàSecondaria

- ricerca(): a seconda del campo immesso, se nome, tipo o destinazione, chiama uno dei metodi presenti nel controller GestioneRicerca il quale si occuperà di reperire le entity pacchetto appropriate.

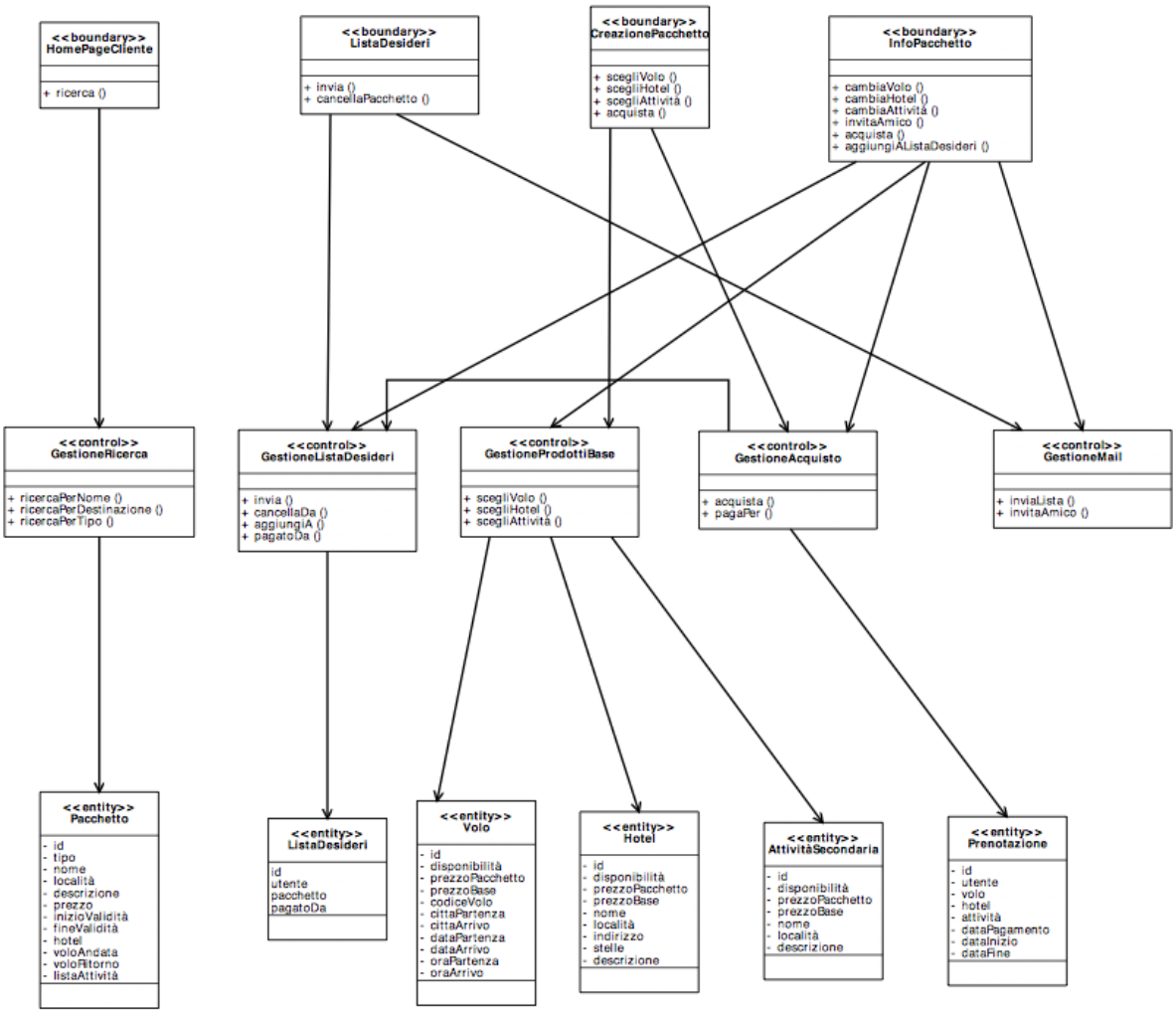

Figure 8: BCE: Cliente

Diagramma rappresentante i metodi messi a dispozione al cliente.
Da HomePageCliente è possibile invocare:

- ricerca(): a seconda del campo immesso, se nome, tipo o destinazione, chiama uno dei metodi presenti nel controller GestioneRicerca il quale si occuperà di reperire le entity pacchetto appropriate.

Il boundary ListaDesideri permette di consultare la propria lista desideri e di compiere le seguenti operazioni:

- invia(): chiama il metodo inviaLista(), presente nel controller GestioneMail, il quale si incaricherà di inviare la mail al giusto destinatario appoggiandosi a server esterni(il controller non si interfaccia con nessuna entity).

- cancellaPacchetto(): permette di cancellare un pacchetto dalla propria lista desideri, invocando il metodo cancellaDa() presente nel controller GestioneListaDesideri, il quale modificherà l'opportuna entity ListaDesideri.

il boundary CreazionePachhetto permette di scegliere e acquistare uno o più prodotti base grazie ai seguenti metodi:

- scegliVolo(), scegliHotel(), scegliAttività(), i quali chiamano gli omonimi metodi presenti nel controller GestorePacchetti. Questi si occuperanno di restituire le entity appropriate.
- acquista(): invoca il metodo acquista() del controller GestioneAcquista, che implementerà il processo di pagamento e si occuperà di creare una nuova entity di tipo prenotazione con l'informazioni del cliente pagante e dei prodotti acquisati.

infine il boundary InfoPacchetto, dove è possibile vedere tutte le info del pacchetto selezionato, eventualmente personalizzarne i contenuti, acquistare, aggiungere alla lista desideri e invitare un amico, ha a disposizione i seguenti metodi:

- cambiaVolo(), cambiaHotel(), cambiaAttività(), i quali chiamano i metodi scegliVolo(), scegliHotel() e scegliAttività() presenti nel controller GestorePacchetti. Questi si occuperanno di restituire le entity appropriate e di settarle come nuove componenti del pacchetto.
- invitaAmico(): chiama l'omonimo metodo presente in GestioneMail, il quale si incaricherà di inviare la mail al giusto destinatario appoggiandosi a server esterni(il controller non si interfaccia con nessuna entity)
- acquista(): invoca il metodo acquista() del controller GestioneAcquista, che implementerà il processo di pagamento e si occuperà di creare una nuova entity di tipo prenotazione con l'informazioni del cliente pagante e dei prodotti acquisati.
- aggiungiAListaDesideri(): invoca il metodo aggiungiA() presente in GestioneListaDesideri, il quale modificherà l'opportuna entity ListaDesideri.

5 Progetto del client

5.1 Progetto della navigazione

Di seguito è trattata la progettazione dell'interfaccia utente e sono mostrati modelli UX per definire i percorsi di navigazione tra le pagine e come queste interagiscono tra loro.

Per non appesantire inutilmente la notazione, i diagrammi che presentiamo di seguito sono separati per tipologia di utente.

Il primo diagramma presentato è relativo alla porzione dell'applicazione accessibile a tutti tramite il bowser: la pagina di benvenuto.

Un utente che accede alla pagina di benvenuto del sito non può fare altro che registrarsi o loggarsi.

In particolare, come rappresentato graficamente nel diagramma, la <<screen>> pagina di benvenuto contiene una <<input form>>, che richiede email e password, per effettuare il login.

Una volta effettuato il login il sistema verifica se le credenziali immesse sono di un cliente registrato, di un dipendente o dell' amministratore, e in base a questo apre schermate appropriate al tipo di utente, rispettivamente AreaCliente, AreaDipendente o AreaAdmin, le cui funzionalità e pagine saranno rappresentate nei diagrammi successivi.

Dalla pagina di benvenuto l'unica altra alternativa è premere il pulsante registrati, in seguito al quale si apre la schermata "registrazione" composta di una form da riempire con i dati. Confermando i dati, in caso di dati validi e quindi di una registrazione avvenuta con successo, si viene reindirizzati allapagina personale AreaCliente.

Banalmente, da qualsiasi area o pagina, una volta loggati, è possibile effettuare il logout e tornare alla pagina di benvenuto.

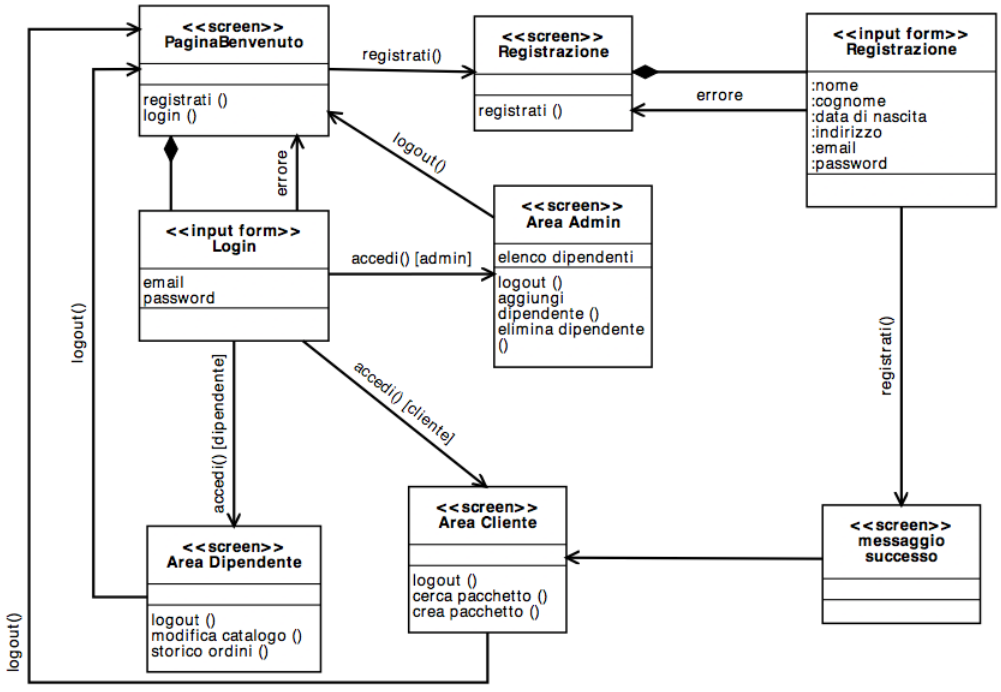

Figure 9: Diagramma UX: funzionalità per utenti non registrati o non loggati

Gli altri diagrammi UX rappresentano le funzionalità dell'appli-cazione accessibili previa autenticazione.

Il secondo rappresenta in dettaglio l' "AreaAdmin" prima nominata, ossia le funzionalità offerte all'amministratore una volta loggatosi.

La pagina iniziale cui accede l'amministratore è costituita da un elenco di dipendenti (in caso di elenco numeroso i dipendenti sono spartiti in più pagine che l' amministratore può scorrere con le opzioni prec e succ).

Le opzioni offerte all' amministratore sono aggiungiDipendente, che apre una pagina con una form da compilare con i dati e le credenziali del nuovo impiegato e in seguito alla conferma ritorna la lista dei dipendenti aggiornata, e eliminaDipendente, che richiede conferma per la cancellazione del cliente.

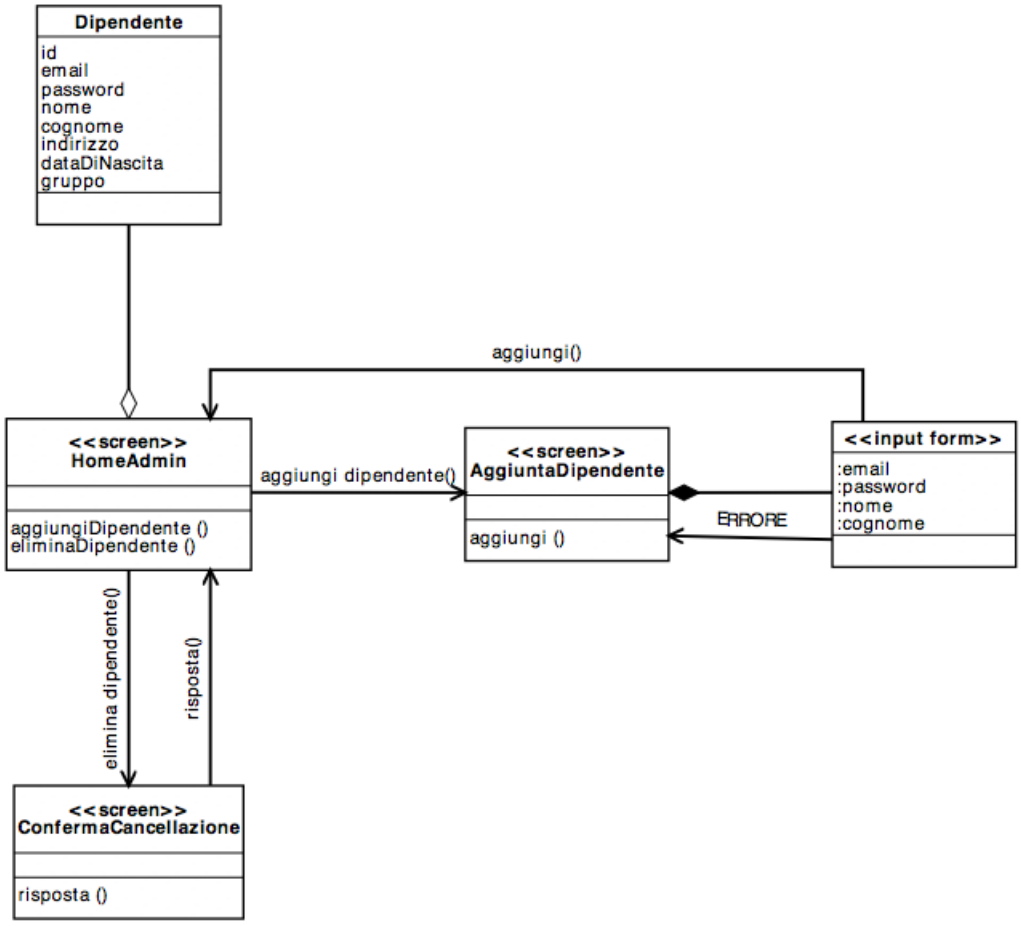

Figure 10: Diagramma UX: AreaAdmin

Il terzo diagramma rappresenta in dettaglio l' "AreaDipendente" nominata nel primo diagramma, ossia le funzionalità offerte ai dipendenti una volta loggatisi.

La pagina iniziale cui accede il dipendente, e a cui può sempre tornare (proprietà rappresentata dal simbolo $), presenta due opzioni: storicoOrdini e modificaCatalogo.

Con la prima si accede all' elenco delle prenotazioni, e si possono scorrere le varie pagine.

Con la seconda invece si accede all' elenco dei pacchetti; le varie pagine si possono scorrere oppure si può effettuare una ricerca per nome inserendo il parametro nel motore di ricerca della pagina.

In questa pagina è fornita l' opzione aggiungiPacchetto in cima all' elenco; premendo il tasto si accede alla schermata con l'input form "DatiPacchettodaRiempire" contenente i campi vuoti da compilare con i dati del pacchetto da aggiungere. Confermando e in mancanza di errori si torna alla pagina con l'elenco dei pacchetti aggiornati.

Inoltre sono fornite le operazioni eliminaPacchetto e modificaPacchetto per ogni pacchetto della lista; la prima richiede una conferma e la seconda fornisce la schermata con l'input form "DatiPacchettoDaModificare" contenente i campi riempiti con i dati attuali da modificare. Entrambe comunque dopo le esecuzioni ritornano la pagina con l'elenco aggiornato.

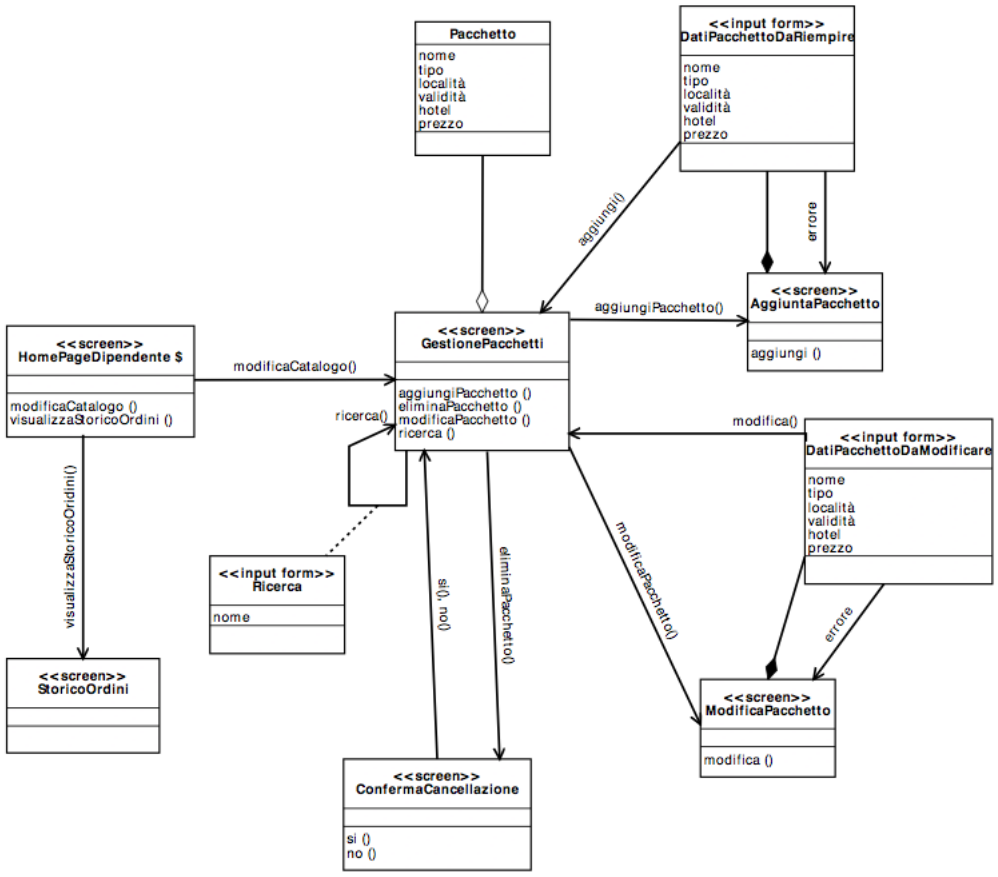

Figure 11: Diagramma UX: AreaDipendente

Il quarto diagramma rappresenta in dettaglio l' "AreaDipendente" nominata nel primo diagramma.

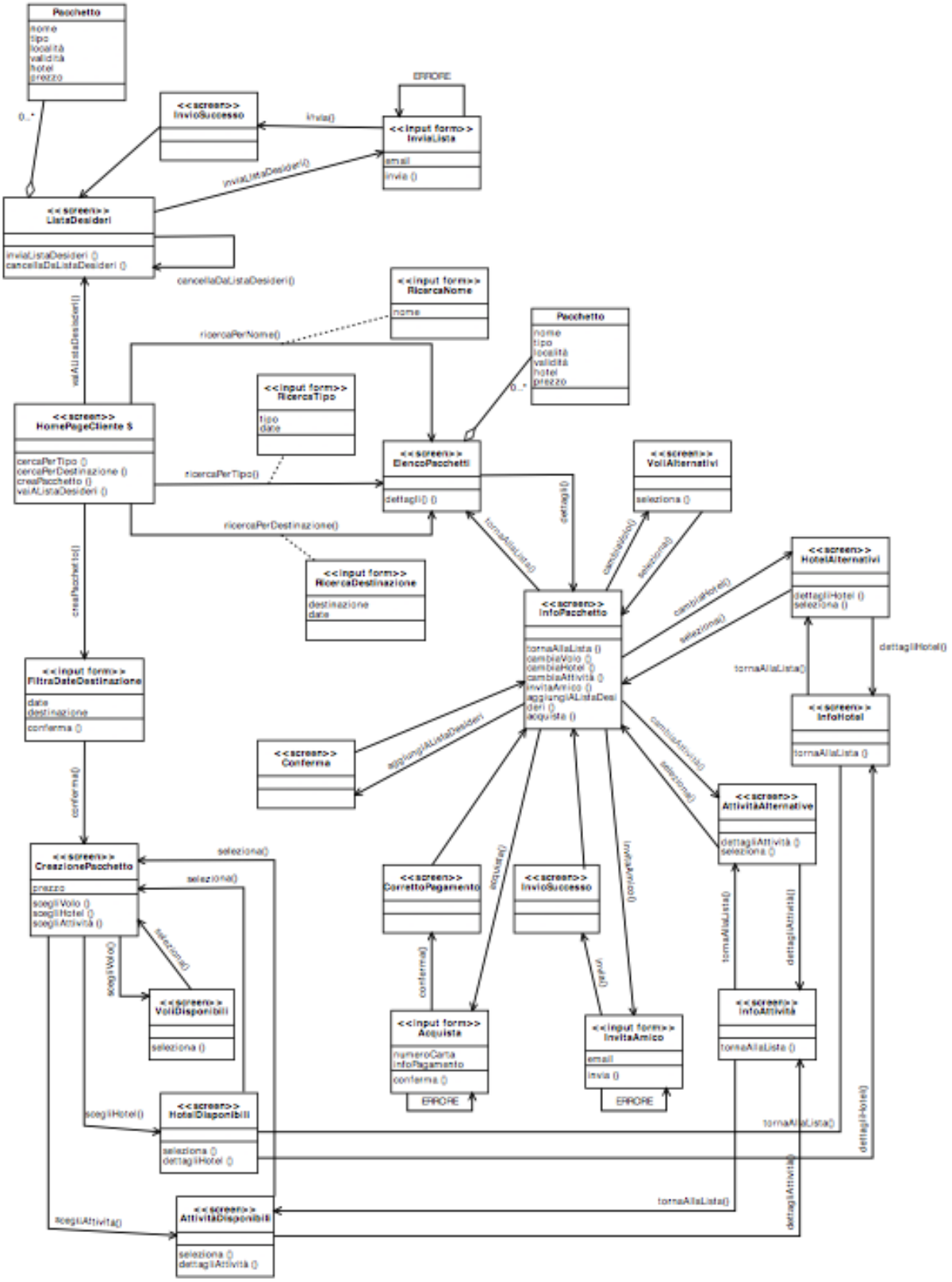

Figure 12: Diagramma UX: AreaCliente

I clienti accedono all'homepage del sito, pagina in cui possono sempre tornare, e che fornisce le operazioni disponibili ai clienti.

Possono, tramite l'opzione "vai alla lista desideri", accedere alla loro lista desideri, cancellare un pacchetto da questa (operazione che non richiede conferma e restituisce la lista aggiornata), e inviarla riempiendo la form che compare con l'email del destinatario.

Possono scegliere l'opzione creaPacchetto,con la quale viene aperta una form "FiltraDataDestinazione" e una volta inseriti e confermati i campi data e destinazione si accede alla pagina per la creazione del pacchetto con le opzioni scegliVolo, scegliHotel, scegliAttività, ognuna delle quali restituisce l'elenco dei prodotti base disponibili, e di cui si possono vedere i dettagli (pagina InfoProdotto) tramite la funzione dettagliProdotto.

Questa pagina ha come attributo il prezzo che viene aumentato ogni volta si aggiunge un prodotto base.

Infine il cliente può effettuare una ricerca, per tipo, inserendo tipo e date nella input form "RicercaTipo", oppure per destinazione inserendo destinazione e date nella input form "RicercaDestinazione", entrambe le form presente nella pagina come motore di ricerca.

Queste ricerche ritornano i pacchetti coerenti con questa.

Scegliendo "dettagli" di fianco ad ogni pacchetto è possibile accedere alla pagina "InfoPacchetto" sulla quale sono disponibili tutte le operazioni: aggiungi a lista desideri, acquista, invita un amico, personalizza (cambia volo, cambia hotel, cambia attività).