

Politecnico di Milano
Scuola di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica



Progetto di Ingegneria del Software 2

TravelDream

(Design Document)

Responsabile:
Prof. Raffaela Mirandola

Progetto di:

Palvarini Matteo Matricola n. 816952

Venturini Gianluca Matricola n. 814236

Zerbinati Francesco Matricola n. 817415

ANNO ACCADEMICO 2013/2014

Contents

1	Introduzione	3
1.1	Panoramica	3
1.2	Acronimi e glossario	4
1.3	Scelte tecnologiche	4
1.4	Descrizione dell'architettura	5
2	Progetto dei dati	5
2.1	Progettazione concettuale	5
2.2	Progettazione logica	8
3	Visione d'insieme	11
4	Progetto del business tier	11
4.1	Modello (Entity Beans, JPA)	11
4.2	Logica applicativa (EJB)	12
5	Progetto del client	13
5.1	Progetto della navigazione	15
5.2	Page sketch	20
6	Rettifiche al RASD	33
7	Ore di lavoro complessive	33

1 Introduzione

L’obiettivo di questo documento è fornire una descrizione della struttura concettuale del sistema che implementeremo, basata sulle specifiche descritte nel documento di analisi dei requisiti. Si rivolge principalmente al gruppo di lavoro che sarà incaricato dell’implementazione e della manutenzione del software. In questo capitolo verrà presentata una panoramica del sistema e della sua architettura. Nei capitoli successivi, analizzeremo in dettaglio la progettazione dei vari elementi che compongono l’applicazione. Dapprima verrà presentata la struttura dei dati, per poi passare agli aspetti relativi alla progettazione delle funzionalità dell’applicazione, e infine al progetto dell’interfaccia utente e della logica connessa. Dove necessario, saranno discussi aspetti implementativi rilevanti per la fase di progettazione.

1.1 Panoramica

Il sistema che verrà realizzato è un applicativo software di e-commerce denominato “TravelDream”. Obiettivo del prodotto è quello di supportare gli utenti che vogliono prenotare un viaggio durante la procedura di scelta e acquisto del pacchetto desiderato. Inoltre il sistema darà un ampio supporto all’azienda omonima durante le procedure di inserimento/modifica/cancellazione dei vari pacchetti e di tutte le componenti che li costituiscono.

Per l’utilizzo della piattaforma si è deciso di progettare un’interfaccia web in primo luogo per rendere il servizio disponibile ad un maggior numero di persone e, in secondo luogo, per permettere un facile aggiornamento del sistema e buona mantenibilità senza dover richiedere particolari installazioni o aggiornamenti da parte dell’utente, in quanto l’applicativo risiederà sul server di TravelDream.

Gli utenti registrati del sistema TravelDream possono visualizzare i pacchetti predefiniti proposti dall’azienda in due modi: utilizzando il catalogo messo a disposizione dal sito oppure utilizzando un piccolo motore di ricerca che permette di selezionare solo i pacchetti viaggio verso le destinazioni di interesse.

In alternativa gli utenti possono creare un loro pacchetto personalizzato, partendo dal selezionare i vari prodotti base che l’azienda offre ed aggregandoli nel modo che viene ritenuto più opportuno.

Ogni utente registrato nel sistema può possedere una lista di pacchetti regalo, visualizzabile dagli altri utenti tramite ricerca per email.

La procedura di acquisto di un pacchetto inizia subito dopo che un utente lo ha selezionato, a questo punto si può decidere di personalizzarlo con delle modifiche riguardanti solo date e orari, ovviamente il sistema controlla la coerenza tra le date dei voli, le date di check-in, check-out degli hotel e quelle delle escursioni. Dopo la personalizzazione è possibile acquistare il pacchetto, inserirlo nella propria lista regali oppure invitare degli amici ad unirsi al viaggio, in tal caso l’acquisto del pacchetto resterà in sospeso, dando la possibilità agli utenti invitati di confermare la propria partecipazione e all’utente che ha invitato di portare a termine il pagamento in qualsiasi momento.

1.2 Acronimi e glossario

Acronimo	Definizione
JEE	Java Enterprise Edition
EJB	Enterprise Java Beans
JPA	Java Persistence API
JSF	Java Server Faces
XHTML	eXtensible HyperText Markup Language
ER	Modello Entità Relazione
DBMS	Database Management System
DTO	Java Data Transfer Object
EAR	Java Enterprise ARchive
JDBC	Java Database Connectivity

1.3 Scelte tecnologiche

Alcune scelte relative alle tecnologie da utilizzare per l'implementazione del sistema condizionano aspetti relativi alla progettazione. Si evidenziano quindi in questa sezione le tecnologie e gli strumenti che utilizzeremo.

- Il sistema è basato sulla piattaforma Java Enterprise Edition 7, in particolare della piattaforma verranno utilizzate:
 - Enterprise Java Beans (EJB) per lo sviluppo della logica applicativa
 - Java Persistence API (JPA) per la gestione della persistenza dei dati del DBMS
 - Java Server Faces 2.0 (JSF 2.0) per lo sviluppo dell'applicazione web
- Viene inoltre adottato GlassFish 4 come application server, utilizzato come container per i componenti che verranno sviluppati.
- Verrà utilizzata, come libreria grafica di supporto, PrimeFaces.
- Il DMBS utilizzato è MySQL Community Edition 5.2, questa scelta non è comunque critica in quanto la comunicazion con il database server è astratta dal JPA.
- Per quanto riguarda l'ambiente di sviluppo utilizzeremo Eclipse (versione Kepler) insieme ai plugin relativi a JEE e GlassFish.
- Per semplificare il lavoro tra i vari componenti del gruppo è stato attivato un repository git presso Google Code (<https://code.google.com/p/project-travel-dream>)

1.4 Descrizione dell'architettura

Lo stile architettonico che abbiamo adottato, multi-tier, è tipico delle applicazioni di tipo enterprise, i cui i vari livelli logici (o strati, tier) sono distribuiti su diverse macchine che comunicano tra di loro (nel nostro caso questo si verifica solo per il Client Tier). Questo modello rende il sistema estremamente modulare.

L'applicativo che si andrà a realizzare sarà suddiviso in quattro livelli: il client tier, il web tier, il business tier e il data tier.

- **Client Tier:** E' il livello tramite il quale gli utenti possono accedere alla funzionalità messe a disposizione dall'applicativo. Il tier comunica tramite il protocollo HTTP con il web tier, inviando al server i dati ricevuti dall'utente e visualizzando le pagine ricevute in risposta dal server. Per la realizzazione di questo tier utilizzeranno la tecnologia JSF e la libreria PrimeFaces.
- **Web Tier:** Questo strato riceve le richieste HTTP dal client tier e risponde inviando le pagine richieste dall'utente. Le pagine vengono generate dinamicamente in base ai dati ricevuti. Il web tier verrà implementato con JEE e funzionerà all'interno dell'application server GlassFish.
- **Business Tier:** Il livello fornisce la business logic dell'applicativo e comunica direttamente con il database. In questo livello sono utilizzati gli EJB e JPA per accedere al database. E' previsto che questo strato interagisca anche con un server di posta elettronica SMTP, per inviare i messaggi di invito ad utenti al sistema. Questo server è esterno al sistema e può quindi essere affidato ad un servizio esterno.
- **Data Tier:** E' costituito dal DBMS MySQL, che garantisce la persistenza dei dati. L'application server comunica con il database utilizzando la tecnologia standard di Java, JDBC.

2 Progetto dei dati

In questo capitolo forniremo il dettaglio delle strutture dati a supporto del sistema, implementate da un DBMS, nello specifico MySql Server 5.2. Si andrà prima ad analizzare la progettazione concettuale, ovvero il modello ER elaborato per l'organizzazione del database, in seguito si forniranno la traduzione logica di quanto visto nel modello e infine si presenterà un grafo completo, realizzato tramite MySQL Workbench, relativo al progetto logico riportato.

2.1 Progettazione concettuale

Si sono individuate le entità e le relazioni dell'applicativo secondo quanto già discusso nel documento RASD. Il risultato di questo lavoro è rappresentato dal diagramma ER di Figura 2.

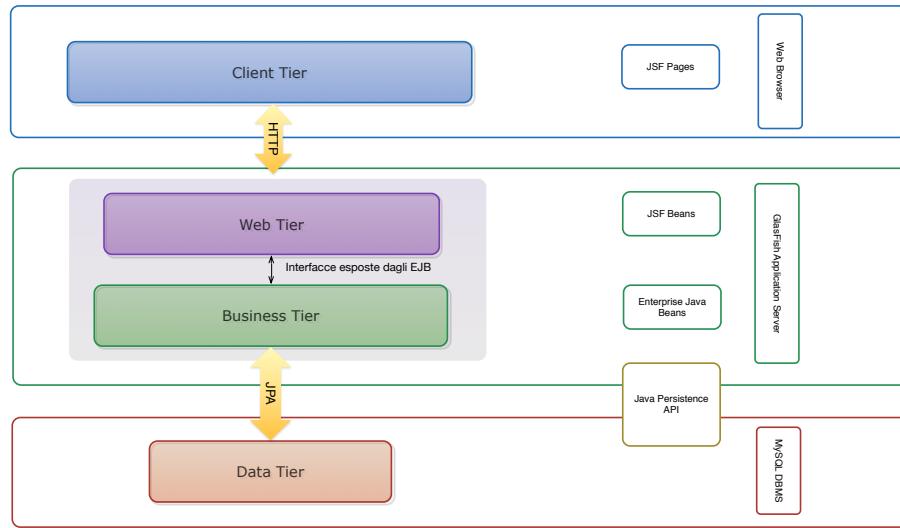


Figure 1: Vista dell'architettura del progetto

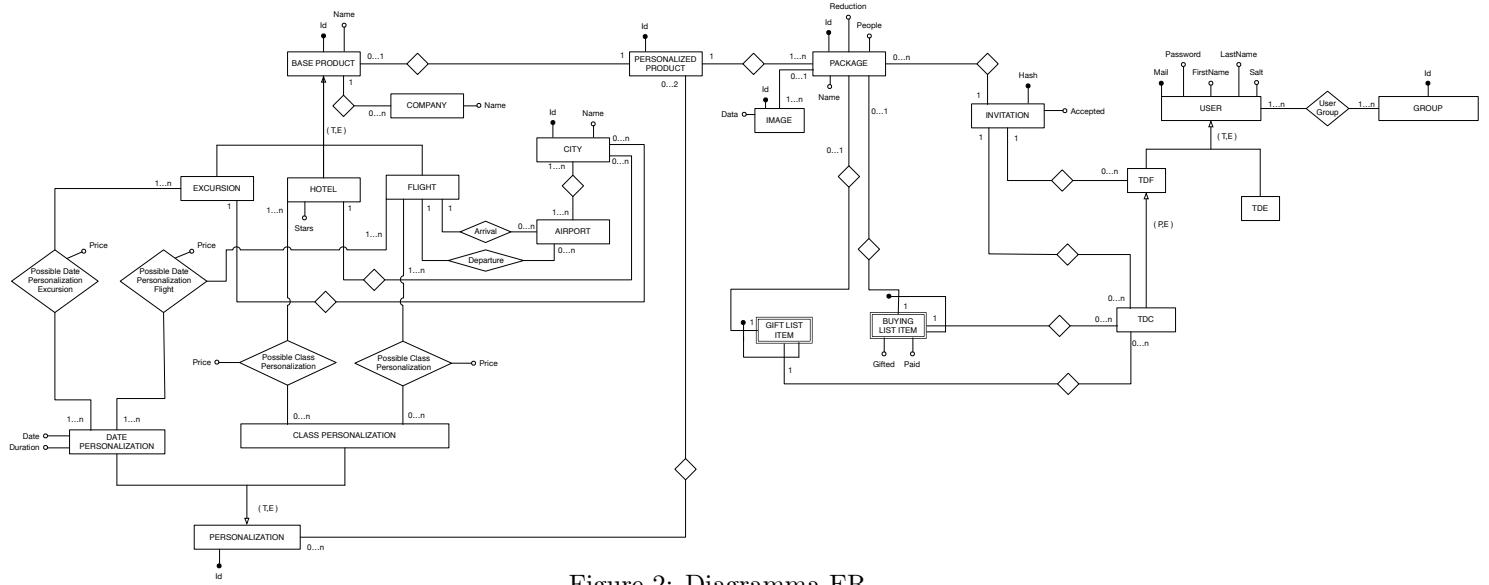


Figure 2: Diagramma ER

Le entità e le associazioni che abbiamo individuato tramite lo schema ER sono di seguito descritte:

- **USER:** Rappresenta un utente registrato al sistema, è identificato da un attributo Email, che è univoco e ha come attributi la Password per l'accesso e alcune informazioni anagrafiche (FirstName e LastName).
- **TDE, TDF, TDC:** Rappresentano gli attori descritti nel RASD, ci si riferisca dunque a tale documento. Ogni USER può appartenere ad uno solo di questi tre gruppi, l'appartenenza è modellata con una relazione tra le entità GROUP e USER.
- **BUYING LIST ITEM:** Ogni utente ha una lista degli ordini effettuati che è rappresentata mediante una tabella nel nostro modello e prevede come identificatore lo stesso TDC e il pacchetto ordinato, dispone degli attributi Gifted e Paid: il primo segnala se il pacchetto è stato regalato da un altro TDC, il secondo indica se è stato ricevuto il pagamento nel conto corrente di TravelDream (parametro modificabile dal pannello di controllo dell'amministratore TDE).
- **GIFT LIST ITEM:** E' un elemento della lista desideri di un utente, è identificata dal TDC proprietario e dal pacchetto. Ogni tupla rappresenta un pacchetto inserito nella lista desideri da un utente.
- **PACKAGE:** Rappresenta i pacchetti presenti nel sistema, essi possono essere contenuti o meno in una PACKAGE LIST. Dispone dei seguenti attributi:

Id:	Identificativo del pacchetto, chiave primaria e dunque univoco
Name:	Nome del pacchetto
Reduction:	Sconto applicato al prezzo del pacchetto (calcolato come somma dei prodotti base inseriti)
People:	Numero di persone previsto per un pacchetto, definisce il numero di partecipanti
Image:	Immagine che rappresenta il pacchetto (questo attributo è opzionale)

- **INVITATION:** Identifica gli inviti che vengono effettuati da un TDC verso altri TDF. E' identificato da un Hash, che viene generato dal sistema e resta univoco. Un INVITATION ha uno e un solo TDC, un TDC può creare più inviti.
- **BASE PRODUCT:** E' il prodotto base del sistema, gli unici attributi sono un Id univoco e un nome Name. Un BASE PRODUCT è inoltre in relazione ad una COMPANY (il cui unico attributo è un nome Name) e può avere o meno una personalizzazione PERSONALIZED PRODUCT. Da BASE PRODUCT discendono (con relazione totale ed esclusiva):

- EXCURSION: Entità che modella le escursioni. Una escursione è effettuata in una e una sola città CITY
- HOTEL: Entità che modella gli hotel, dispone dell'attributo Stars che indica le stelle dell'albergo, un hotel è disponibile in una e una sola città CITY
- FLIGHT: Entità che modella i voli aerei del sistema, è in relazione con AIRPORT in quanto ha un aeroporto di arrivo Arrival e un aeroporto di partenza Departure.
- HOTEL e FLIGHT: Dispongono, in aggiunta a quanto detto, di una relazione “Possibile Class Personalization” che, con un attributo di prezzo “Price” definisce il prezzo aggiuntivo da applicare nel caso venga scelta quella personalizzazione.
- EXCURSION e FLIGHT: Dispongono, in aggiunta a quanto detto, di una relazione “Possibile Date Personalization” che contiene anch’essa un attributo “Price” che svolge le stesse funzioni sopra menzionate
- PERSONALIZATION: Dispone di un id per essere identificata, un PERSONALIZED PRODUCT può avere da 0 (nessuna personalizzazione) a 2 personalizzazioni (esse non possono essere entrambe dello stesso tipo, questo vincolo sarà modellato ed evidenziato più avanti durante la traduzione logica). Una PERSONALIZATION può essere:
 - DATE PERSONALIZATION: Definisce le personalizzazioni di data, fornendo un data Date di partenza e una durata del viaggio Duration espressa in minuti
 - CLASS PERSONALIZATION: Indica la personalizzazione di una classe (ad esempio classe “economy” o “business” in un volo aereo e “suite” o “camera di lusso” nel caso di un hotel)

2.2 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 risultante è riassunto tramite le seguenti:

```

USER(Mail, Password*, FirstName*, LastName*)
GROUP(Id)
USER_GROUP(UserId,GroupId)
COMPANY(Name)
EXCURSION(Id, Name, Company)
HOTEL(Id, Name, Company)
CITY(Id,Name)
AIRPORT(Id,CityId)
FLIGHT(Id, Name, Company,Arrival,Departure)
DATE_PERSONALIZATION(Id, Date, Duration)
  
```

```

CLASS_PERSONALIZATION(Id, Class)
POSSIBLE_DATE_PERSONALIZATION_FLIGHT(FlightId,DatePersonalizationId,duration)
POSSIBLE_DATE_PERSONALIZATION_EXCURSION(ExcursionId,DatePersonalizationId)
POSSIBLE_CLASS_PERSONALIZATION_HOTEL(HotelId,ClassPersonalizationId)
POSSIBLE_CLASS_PERSONALIZATION_FLIGHT(FlightId,ClassPersonalizationId)
PERSONALIZED_PRODUCT_HOTEL(Id,HotelId,PackageId,DatePersonalization,ClassPersonalization)
PERSONALIZED_PRODUCT_FLIGHT (Id,FlightId,PackageId,DatePersonalization,ClassPersonalization)
PERSONALIZED_PRODUCT_EXCURSION (Id,ExcursionId,PackageId,DatePersonalization)
PACKAGE(Id,Reduction,People, name, image)
GIFT_LIST_ITEM(PackageId,UserId)
BUYING_LIST_ITEM(PackageId,UserId,Gifted,Paid)
INVITATION(Hash,PackageId,InviterUserId,InvitedUserId,Accepted)

```

Lo schema di traduzione adottato ha previsto il collasso verso il basso e il collasso verso l'alto di numerose entità, si noti infatti come non siano più presenti l'entità BASE PRODUCT e PERSONALIZATION (entrambe collassate verso il basso).

Abbiamo inserito delle tabelle aggiuntive per tradurre nel modello logico delle associazioni molti a molti, quali ad esempio le “POSSIBLE DATE PERSONALIZATION” per tutte e tre le tipologie di prodotto base, le “POSSIBLE CLASS PERSONALIZATION” per i prodotti base HOTEL e FLIGHT. E’ stato eseguito lo stesso procedimento con la relazione “USER_GROUP”.

Dal modello logico qui descritto abbiamo ottenuto poi il codice SQL ed in seguito l’abbiamo visualizzato tramite MySQL WorkBench 6.0.

Di seguito si riporta il modello ottenuto:

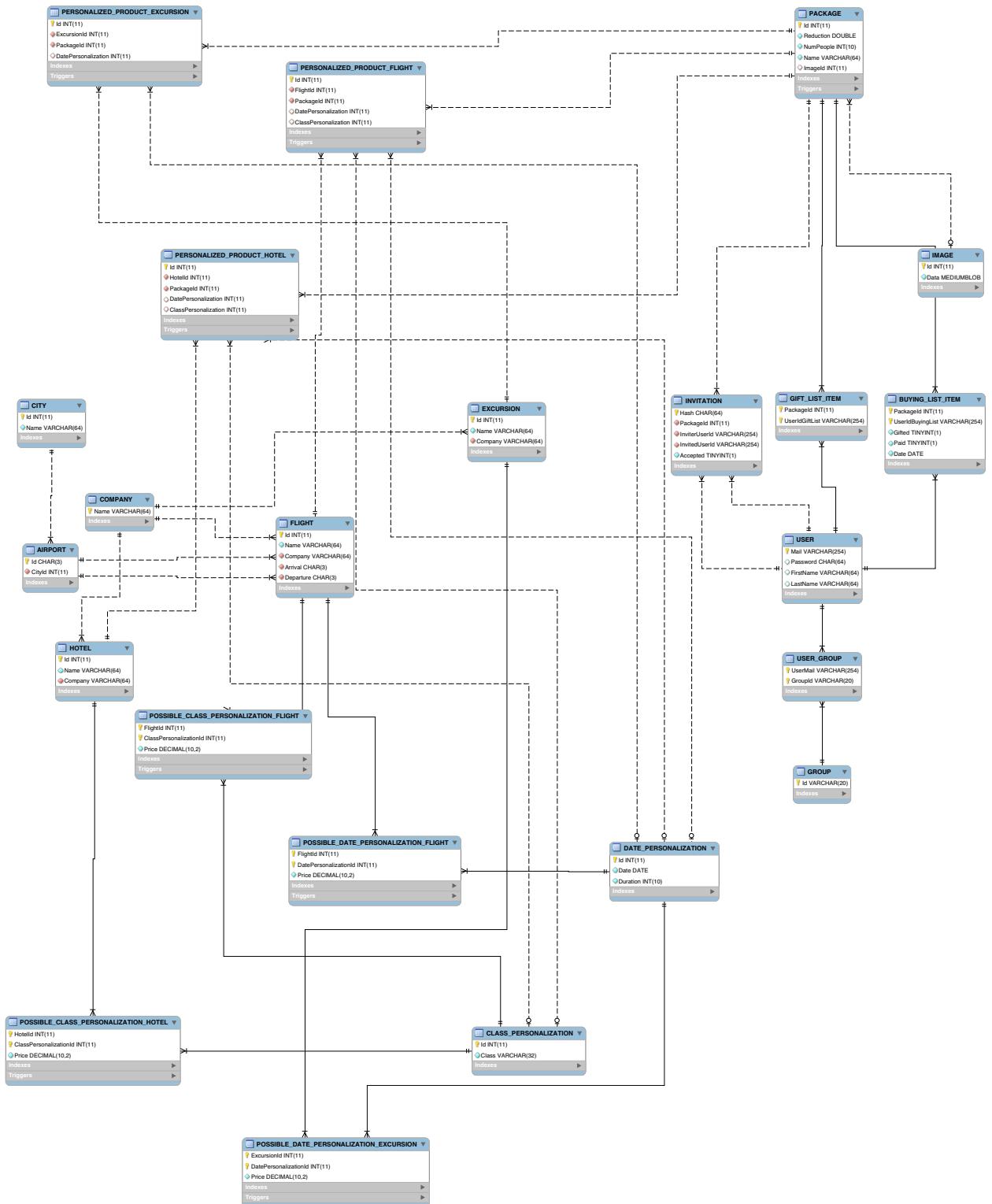


Figure 3: Schema ottenuto con MySQL WorkBench

3 Visione d'insieme

Si è voluto organizzare l'intero progetto completo di business e client tier per avere una visione d'insieme di quello che andremo a implementare. Non ci si è soffermati, in questo schema, a descrivere le singole classi e le varie componenti della struttura ma si sono semplicemente evidenziate le relazioni tra le varie componenti (sono raccolte dunque le JPA, i vari EJB e le pagine JSF con i relativi managed beans) che verranno poi descritte in maniera approfondita nelle sezioni seguenti del documento.

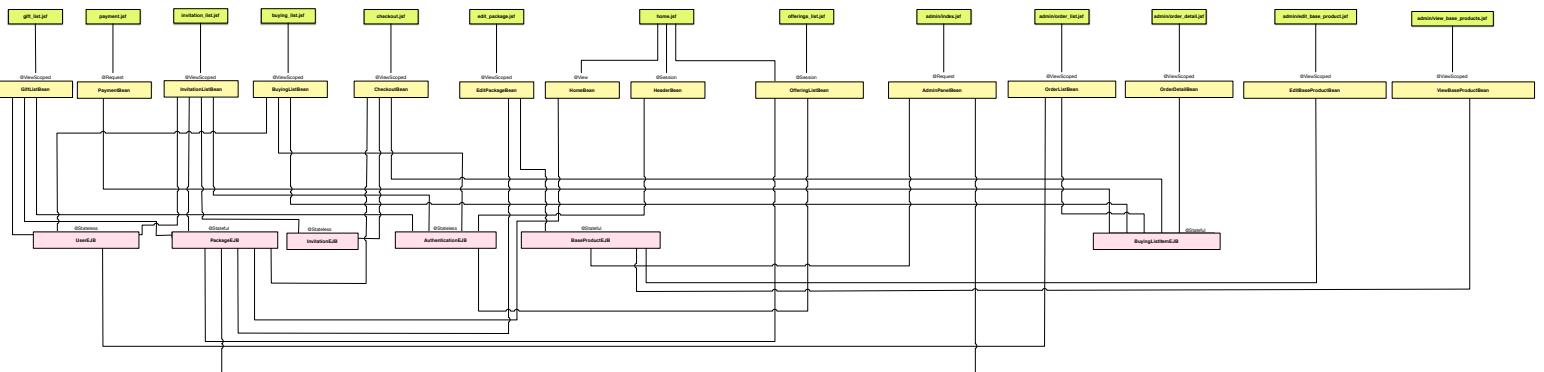


Figure 4: Schema generale dell'architettura

4 Progetto del business tier

Una volta definita la base dati del nostro sistema e l'architettura a tier, abbiamo progettato le componenti software da realizzare per implementare la logica applicativa. Il business tier contiene sia le componenti software che descrivono le entità JPA (e quindi il modello dei dati) che la logica applicativa, implementata tramite gli EJB.

4.1 Modello (Entity Beans, JPA)

L'insieme delle classi JPA Entity, chiamate anche Entity Beans, è direttamente collegato al modello logico dei dati prima discusso, conseguente alla realizzazione del modello ER. Gli Entity Beans sono utili in quanto introducono un livello di astrazione che può essere utilizzato all'interno del progetto Java. Ogni Entity Bean corrisponde in maniera univoca ad una entità sullo schema del progetto logico, e quindi al database.

La realizzazione di tale struttura ci permette di facilitare le operazioni sulla base dati e le varie JPA che sono introdotte sono schematizzate nella Figura 5. In particolare si tenga presente come tali entità siano sempre implicitamente costituite da metodi setter e getter per essere facilmente manipolabili all'interno dell'applicativo.

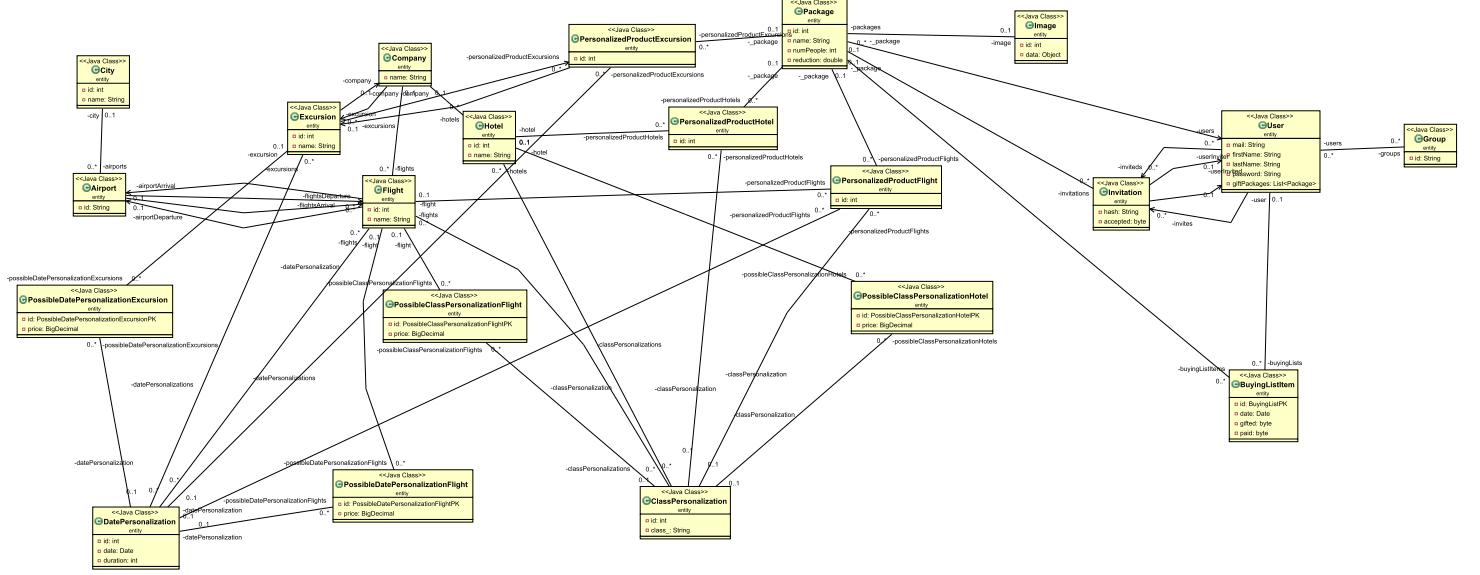


Figure 5: Diagramma UML delle JPA

4.2 Logica applicativa (EJB)

Per sviluppare la logica applicativa del progetto, si sono utilizzati i Session Bean messi a disposizione da Java Enterprise. Ricordiamo come questa tipologia di componenti sia il nesso tra le Entity Beans e il web client.

All'interno dei Session Bean, si noti come siano state usate classi stateful per i sistemi di autenticazione gestiti tramite “realm” (l'EJB denominato “AuthenticationEJB”) mentre negli altri casi si sono utilizzati componenti stateful e stateless a seconda delle necessità.

Il diagramma delle classi UML di Figura 6 mostra il design dei Enterprise Java Beans che verranno implementati.

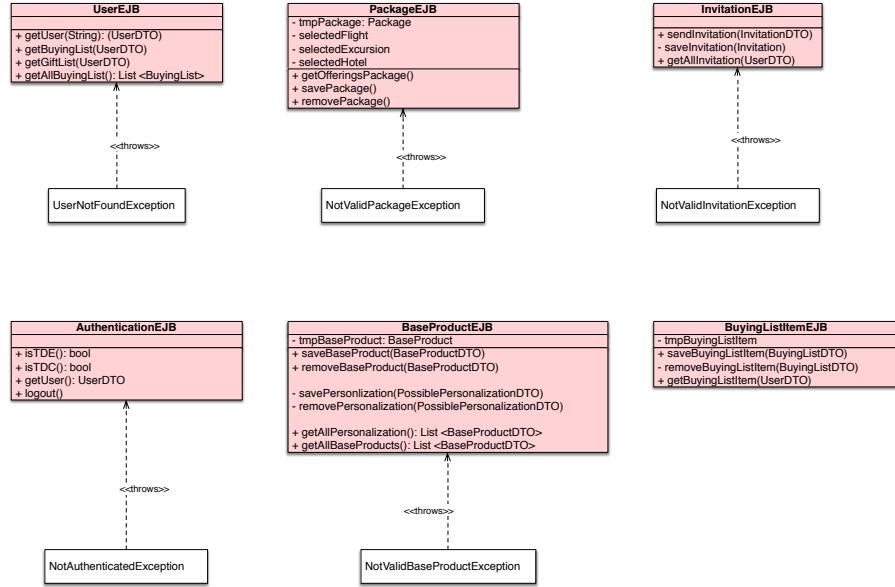


Figure 6: Schema UML per gli EJB individuati nel sistema

5 Progetto del client

Una volta determinate le funzionalità offerte dallo strato di logica applicativa, abbiamo progettato il web tier, lo strato che garantisce l'interazione con l'utente che è collegato al sistema tramite un browser web. Abbiamo quindi progettato la struttura dell'interfaccia tramite diagrammi concettuali, che descrivono i percorsi di navigazione all'interno dell'applicativo, per poi definire la struttura logica con la quale verranno implementati. Abbiamo anche ritenuto conveniente disegnare la struttura di molte delle pagine visualizzabili dagli utenti (sia clienti che amministratori del sistema) in maniera da facilitare il nostro lavoro in fase di sviluppo.

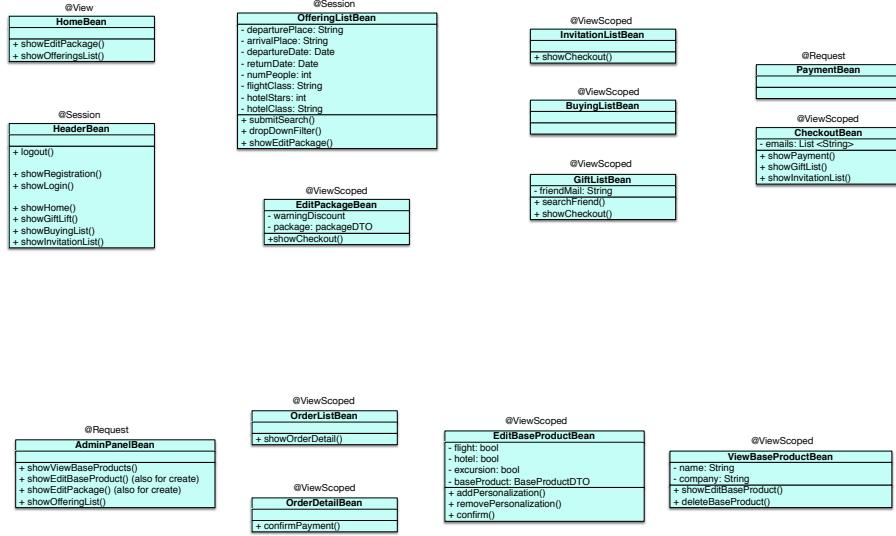


Figure 7: Schema UML per i Managed Beans del sistema

Si elencano ora alcune annotazioni prese durante la fase di stesura di questa sezione:

HeaderBean Le funzioni di “show” redirigono alle rispettive pagine, controllando che l’utente abbia i permessi necessari per accedervi.

EditPackageBean La funzione “showCheckout()” viene eseguita solo se sono rispettati tutti i vincoli sul pacchetto (verificati dal TDE), altrimenti si mostrerà l’errore relativo.

OfferingListBean Contiene una funzione che genera la tabella con tutti i pacchetti disponibili, secondo i criteri di ricerca selezionati tramite la funzione `dropDownList()` e `submitSearch()`.

CheckoutBean Prevede le tre modalità di conclusione dell’ordine, quali acquisto, inserimento nella lista regali e invito di amici ad unirsi al pacchetto.

5.1 Progetto della navigazione

Per rappresentare i percorsi di navigazione attraverso l’interfaccia utente dell’applicazione web abbiamo sfruttato come tipo di diagramma il modello UX, la cui notazione è basata sui diagrammi delle classi UML.

In particolare abbiamo deciso, per semplicità e chiarezza di lettura, di suddividere l’UX in 4 diagrammi che rappresentano funzionalità e/o ruoli diversi degli utenti che possono svolgere il flusso di lavoro evidenziato. Va notato che la copertura delle funzionalità ci pare esaustiva nei suoi aspetti principali, possono mancare tuttavia alcuni aspetti secondari che sono intuitivi e avrebbero solo appesantito la notazione.

Nel diagramma di Figura 8 si mostra la parte di applicazione il cui flusso di attività principale ha origine nella schermata HOMEPAGE (evidenziata in arancione). Questa porzione di applicazione è quella accessibile a tutti gli utenti che non hanno effettuato il login. Va notato che dalla homepage (come anche da tutte le altre schermate) è presente anche un header nella parte superiore della pagina che contiene vari link e al quale dedichiamo un UX diagram ad hoc.

Inoltre le altre convenzioni di colore delle classi adottate sono: *giallo* per le pagine (screen), *verde* per le input form, *bianco* per le classi di dati. Particolare importanza hanno nel nostro modello le classi di colore *azzurro*, che indica che esse non sono dettagliate nell’UX corrente, ma fanno da passaggio ad un altro diagramma UX nel quale sono specificate adeguatamente.

Si nota, ad esempio, che l’UX di Figura 8 si ferma alla pagina EDIT_PACKAGE, questa infatti non è raggiungibile da un utente non loggato.

Alla pagina EDIT_PACKAGE dedichiamo invece l’UX di Figura 9: qui abbiamo sia i metodi e le funzionalità svolti dal TDC (ad esempio filtrare i prodotti a scelta, fare drag&drop, confermare per andare al checkout), ma anche quelli del TDE che dal pannello di amministrazione arrivano qui per modificare/eliminare un pacchetto.

In Figura 10 viene mostrato l’UX che riassume la navigazione a partire dall’header presente in tutte le pagine. Sui collegamenti delle pagine dopo lo slash viene indicato l’attore che può accedere a quel collegamento a partire dal proprio header, infatti i link contenuti nell’header variano a seconda che l’utente sia non loggato oppure loggato come TDC oppure loggato come TDE.

Infine in Figura 11 si descrive in dettaglio la parte relativa all’area del TDE, che parte dal pannello di amministrazione e raggiunge alcune pagine dedicate, mentre per EDIT_PACKAGE e OFFERINGS_LIST (in azzurro) sfruttiamo le stesse già definite nei precedenti diagrammi aggiungendo nel caso di TDE i metodi e le funzionalità proprie di questo attore.

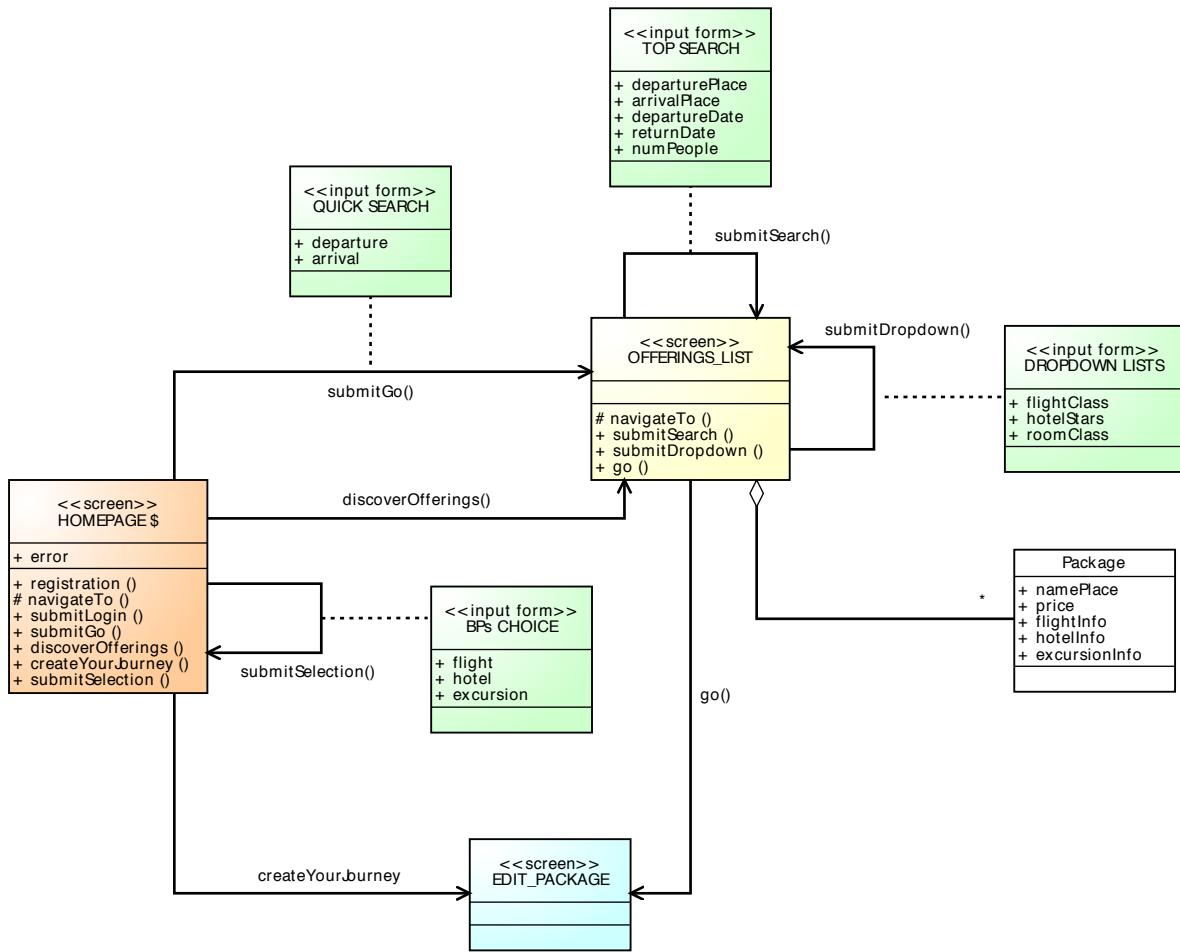


Figure 8: UX: area utente non loggato

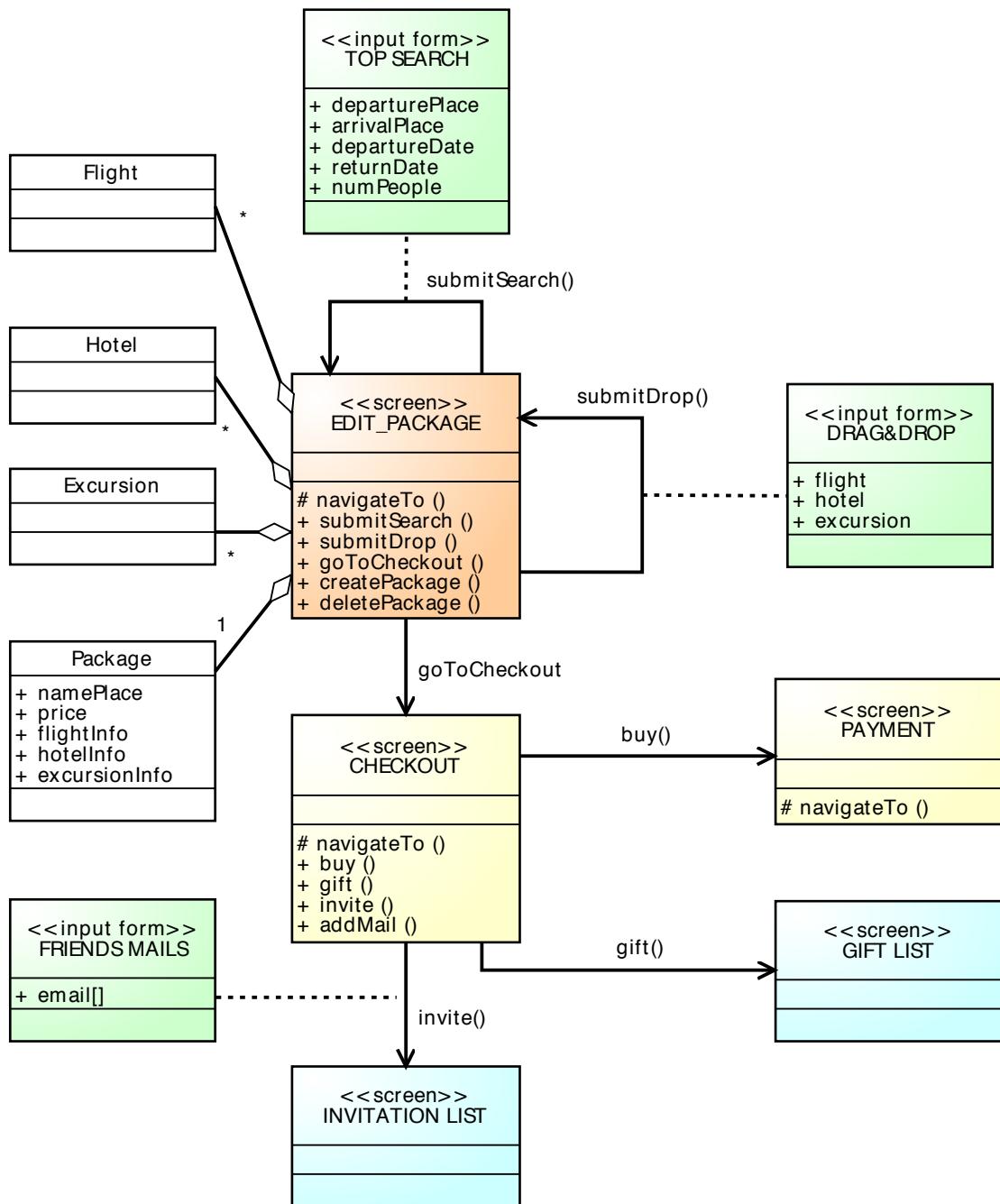


Figure 9: UX: dettaglio navigazione a partire da EDIT_PACKAGE

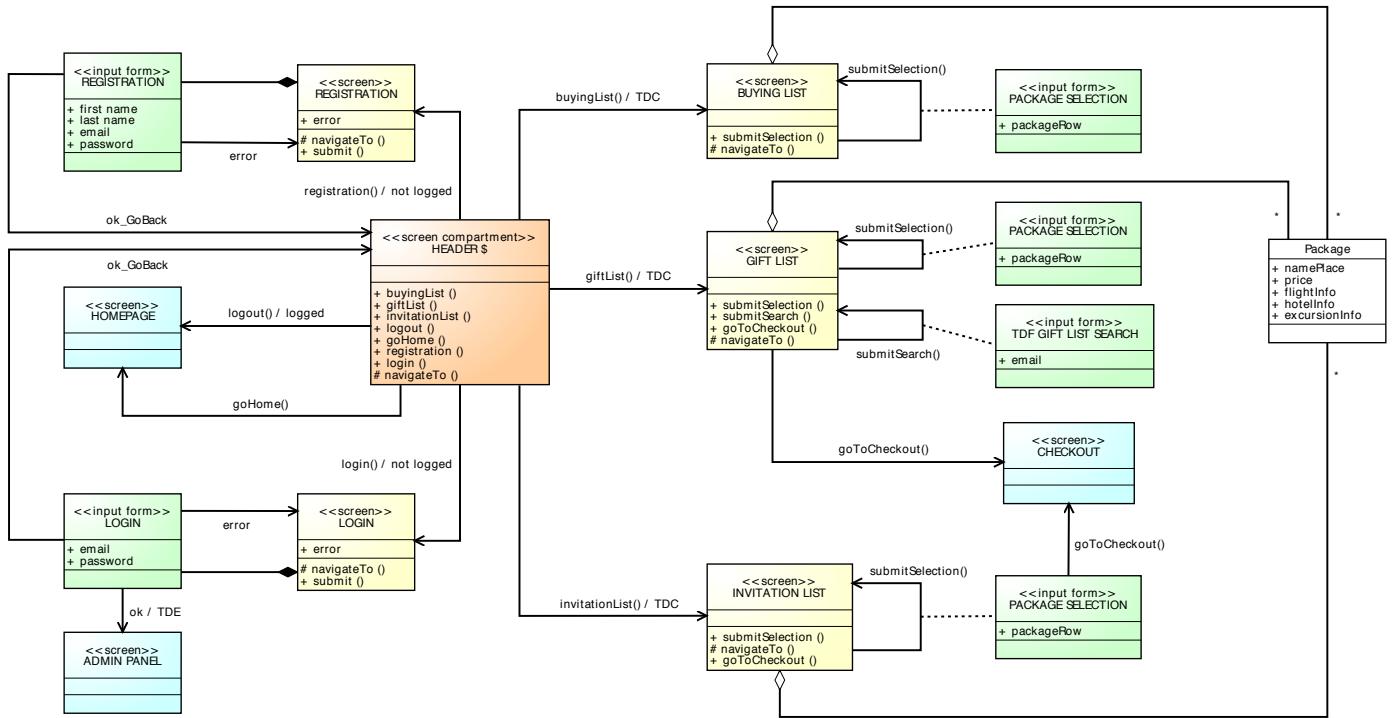


Figure 10: UX: navigazione a partire dall' HEADER

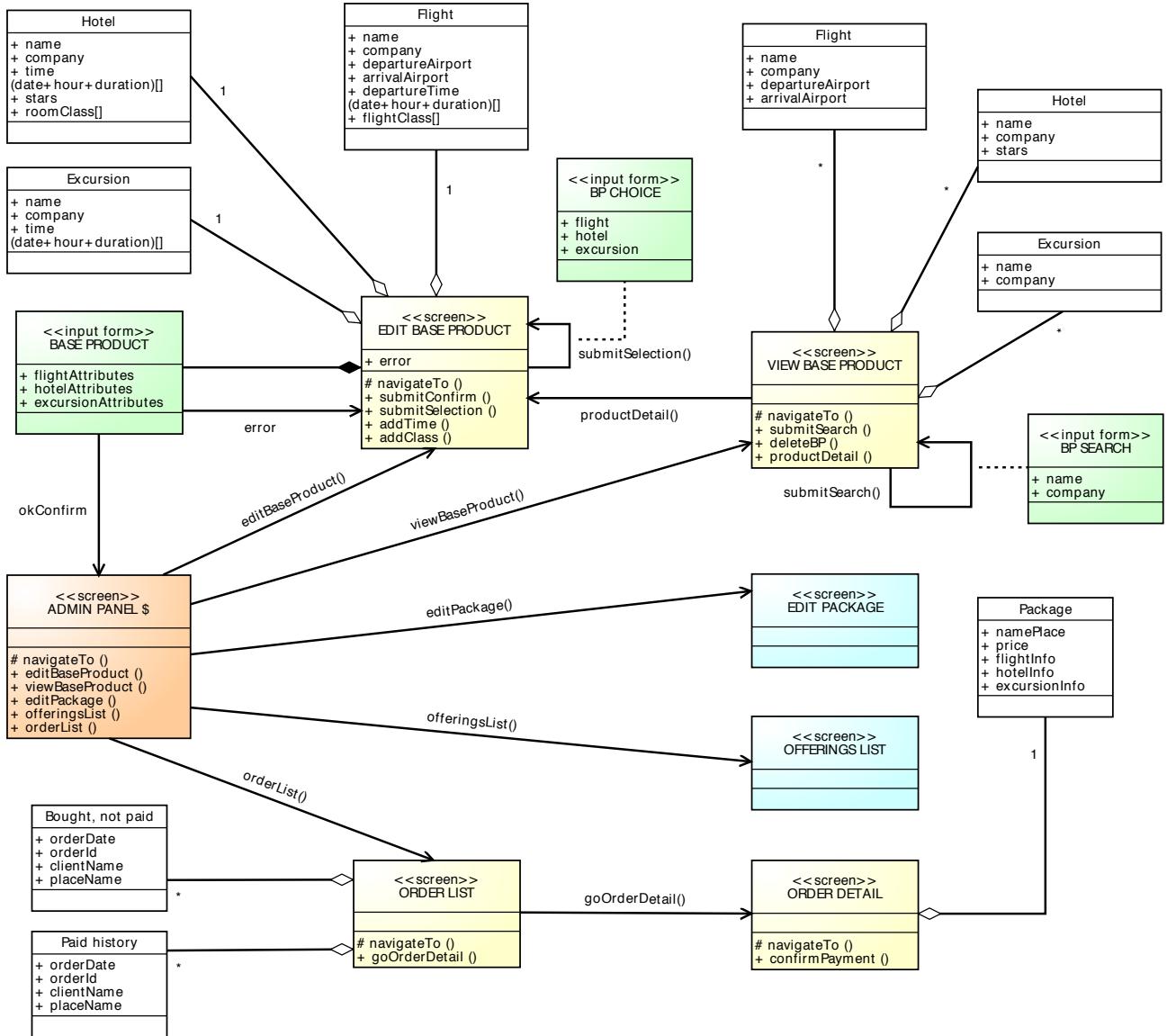


Figure 11: UX: area TDE

5.2 Page sketch

In questa sezione si presentano i mockup delle pagine web che verranno realizzate nel sistema. Le interfacce sono state disegnate prima del progetto della navigazione, per facilitarci a capire cosa realmente servisse all'interno del sistema per soddisfare le necessità da noi pensate nel documento RASD.

Questi schemi ci hanno dunque permesso di individuare meglio i percorsi di navigazione necessari e di poter individuare i dati da presentare nelle varie schermate e ci supporteranno inoltre durante la fase di implementazione, avendo già una base sulla quale lavorare. I layout presentati verranno ulteriormente raffinati in fase di sviluppo.

Le schermate individuate per la realizzazione del progetto sono di seguito riportate con una breve didascalia che contiene il nome che avrà la corrispondente pagina .jsf e ne descrive le funzioni principali.



Figure 12: *home*: Home Page

The screenshot shows the registration page for the Travel Dream website. The title "Registrazione" is at the top. The form consists of five input fields: "Nome" (Name), "Cognome" (Surname), "e-mail" (Email), "password" (Password), and "conferma password" (Confirm Password). Each input field has a placeholder text inside. Below the input fields is a "conferma" (Confirm) button.

Figure 13: *registration*: schermata per la registrazione di un nuovo utente

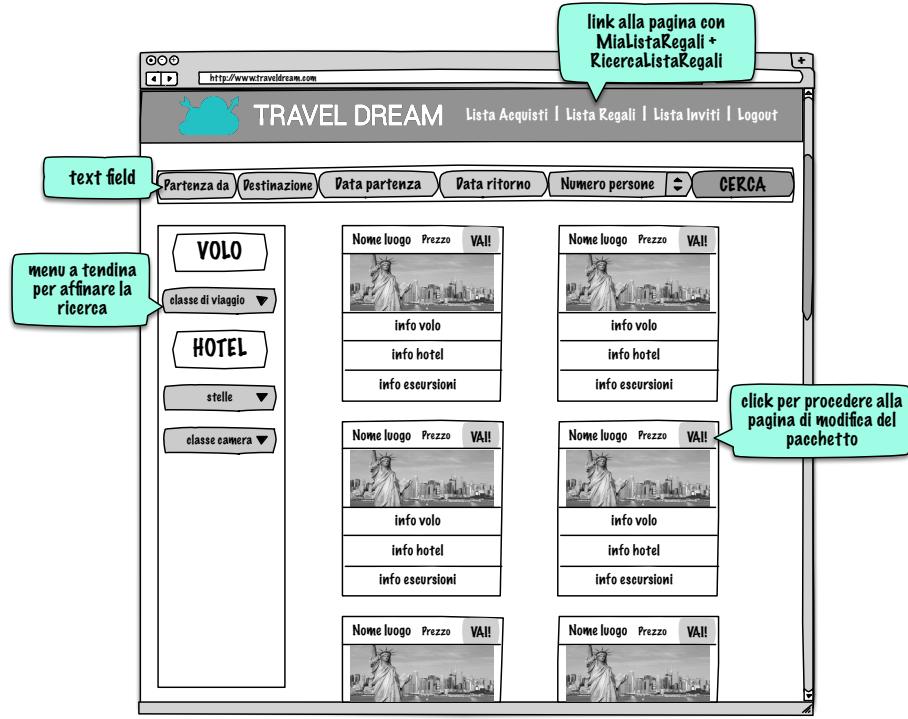


Figure 14: *offerings_list*: catalogo dei pacchetti disponibili, con possibilità di filtrare i risultati

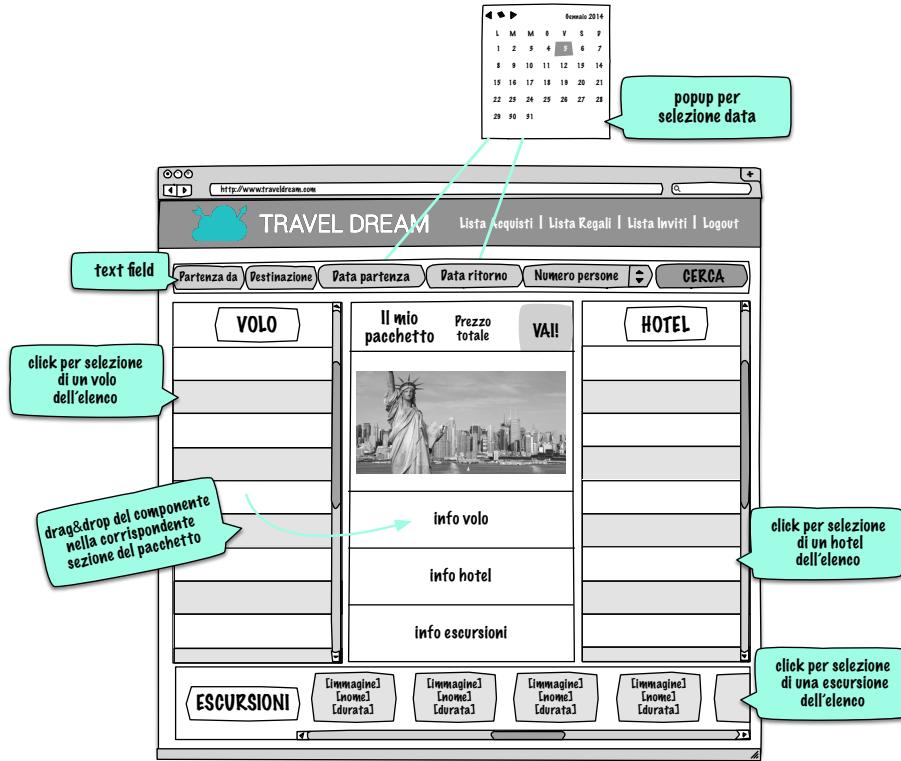


Figure 15: *edit_package*: schermata per la creazione di un nuovo pacchetto



Figure 16: *edit_package*: schermata per la modifica di un pacchetto esistente

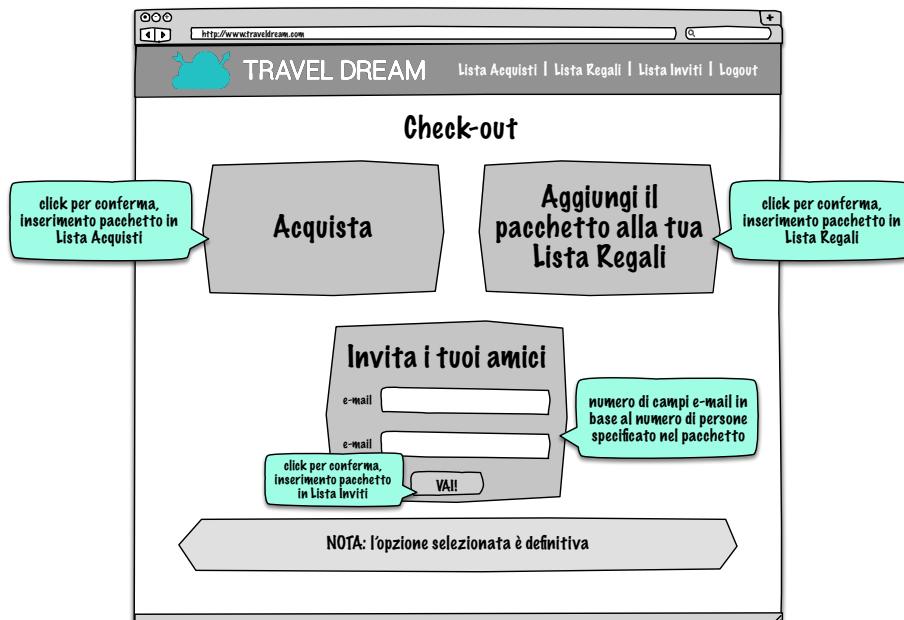


Figure 17: *checkout*: mostra le 3 diverse opzioni per la finalizzazione dell'ordine



Figure 18: *payment*: schermata di riepilogo dell'ordine con dati per effettuare il pagamento



Figure 19: *gift_list*: lista regali dell’utente registrato, vi si accede tramite link dell’header

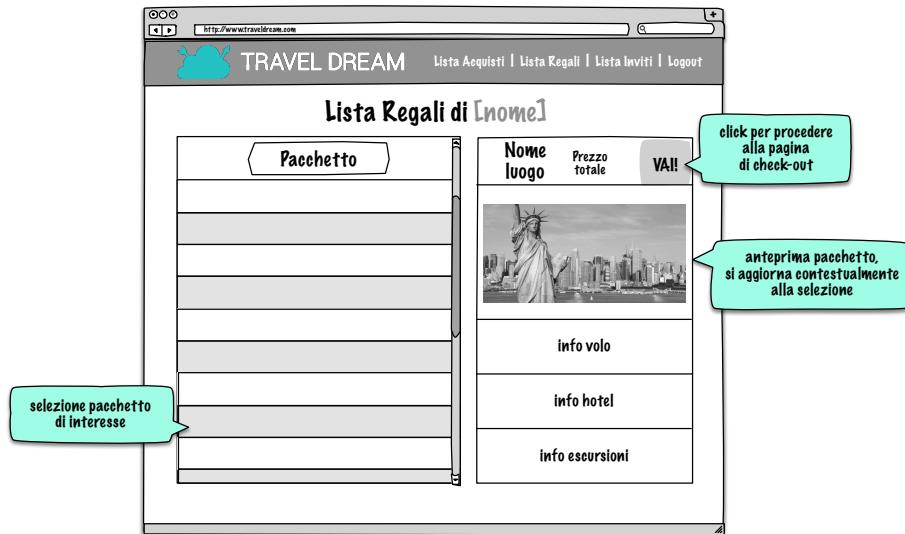


Figure 20: *gift_list* (dopo ricerca di amico): lista regali di un altro utente



Figure 21: *buying_list*: lista degli ordini di un utente

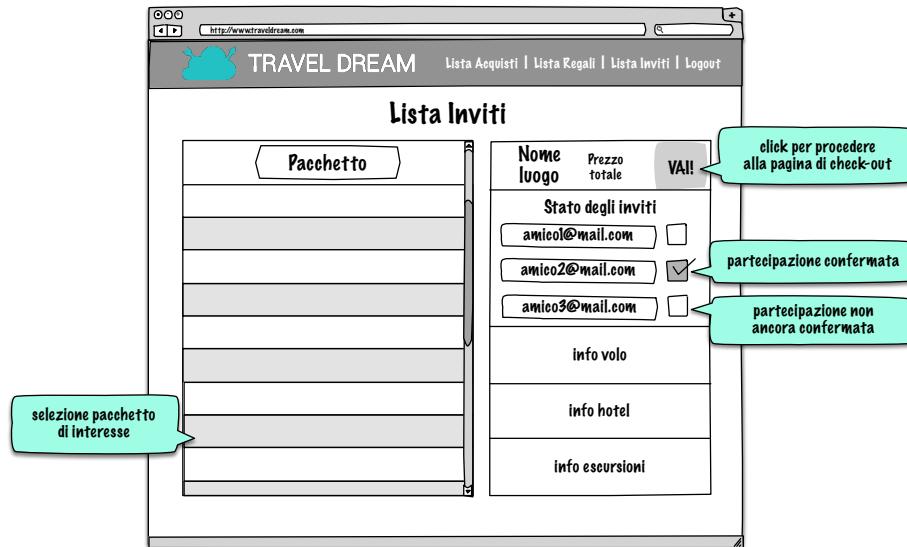


Figure 22: *invitation_list*: Lista di riepilogo degli inviti effettuati da un utente per i vari pacchetti

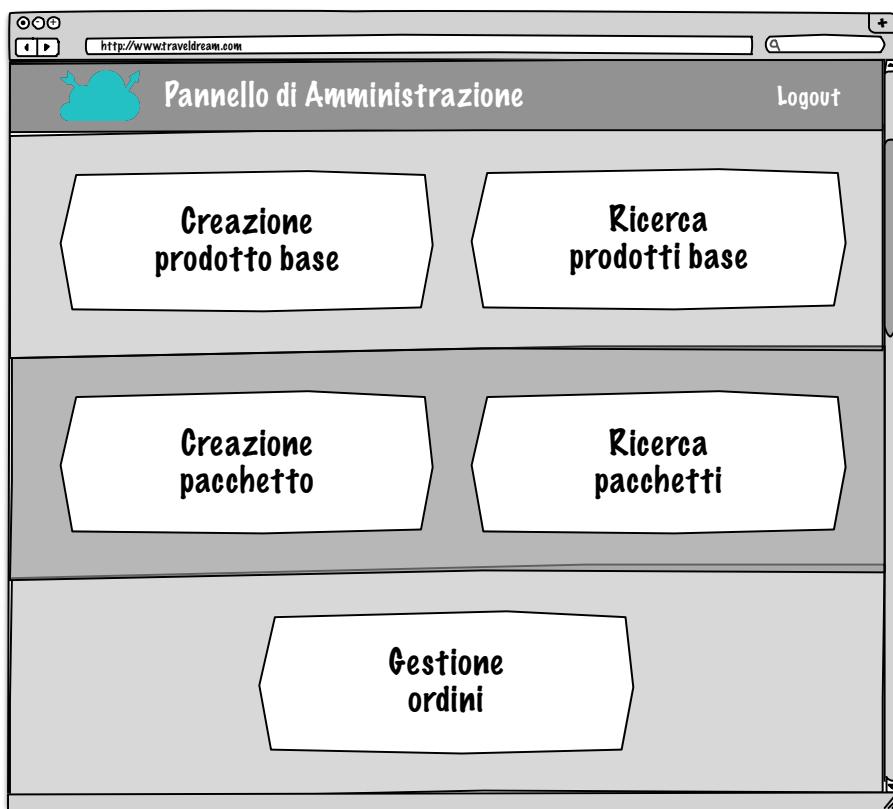


Figure 23: *admin/index*: pannello di amministrazione per i TDE

click per selezione/deselezione del componente

Logout

Crea/Modifica Prodotto Base

VOLO HOTEL ESCURSIONE

nome text-field

azienda

aeroporto di partenza menu a tendina

aeroporto di arrivo

partenza data ora text-field
durata min

classe volo

+ partenza **+ classe** click per aggiungere nuova coppia partenza-durata oppure classe volo

conferma

Figure 24: *edit_base_product*: pagina TDE per creazione e modifica di un pacchetto

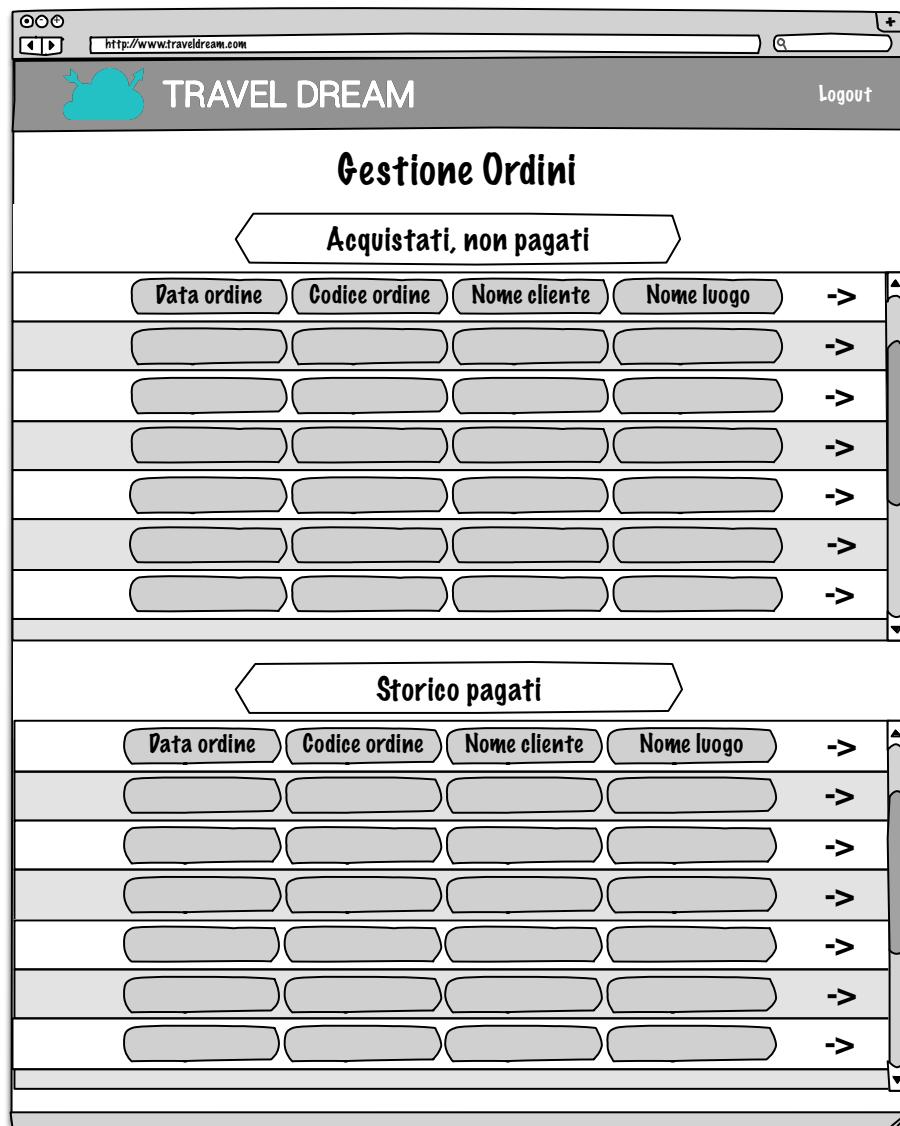


Figure 25: *order_list*: riepilogo degli ordini inseriti nel sistema



Figure 26: *order_detail*: pagina di dettaglio per un ordine con pulsante di conferma di ricevuta del pagamento

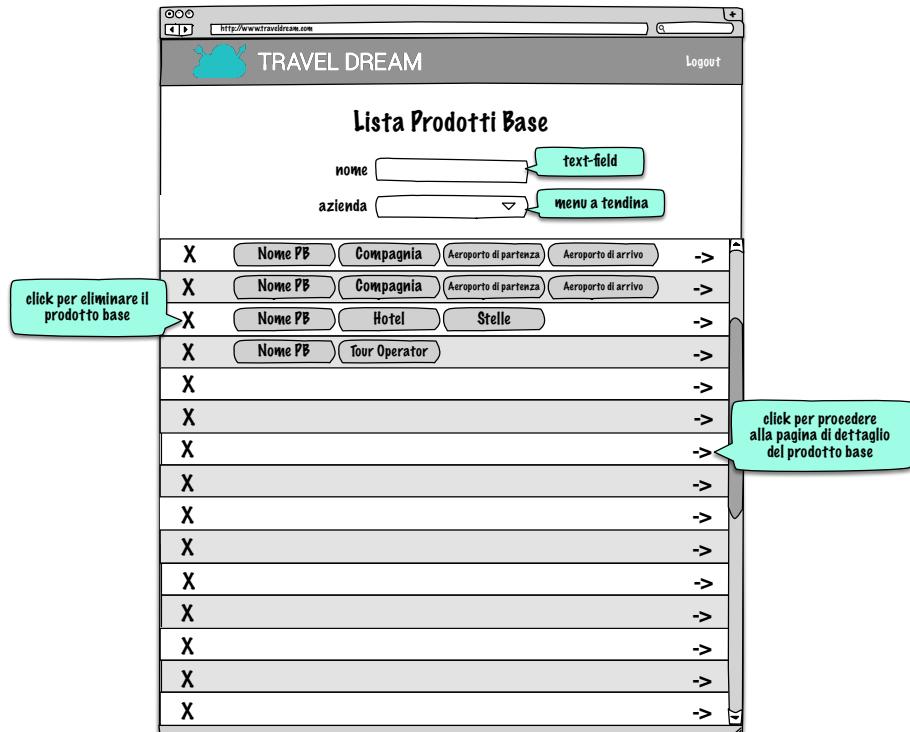


Figure 27: *view_base_products*: schermata di ricerca dei prodotti base per eliminarli/vedere i dettagli

6 Rettifiche al RASD

Durante la stesura di questo documento ci siamo accorti di voler modificare alcune considerazioni fatte nel documento RASD per risolvere alcuni problemi riscontrati solo durante questa fase del progetto:

- Si specifica, come considerazione, che gli hotel siano aperti tutto l'anno e quindi non dispongano di un attributo “DATE PERSONALIZATION”
- Si sono modificati alcuni UI Sketch precedenti con l'introduzione di tutte le funzioni pensate in fase di design e alcune correzioni minori. Per completare la copertura di tutto il sistema sono stati inoltre aggiunti tutti gli sketch relativi all'area riservata al TDE.

7 Ore di lavoro complessive

Sono state conteggiate le ore di lavoro per la stesura di questo documento. Queste sono state suddivise tra ore condivise tra i vari componenti del gruppo, per la definizione in comune accordo dei punti più importanti del documento e di carattere più generale, e ore di lavoro individuale aggiuntive per la realizzazione più pratica di questo Design Document.

Ore di lavoro in gruppo	21
-------------------------	----

Persona	Ore di lavoro individuale
Palvarini	15
Venturini	17
Zerbinati	12