Università degli studi di Salerno
Corso di Laurea in Informatica

INGEGNERIA DEL SOFTWARE

Object Design Document

"UNI-AirLines"

Studenti:

Nome	***Matricola***
Santoro Mario	0512104850
Marino Raffaele	0512104508
Pastore Matteo	0512104724
Fortunato Angelo	0512104532

Anno Accademico: 2018/19

SOMMARIO

1. Introduction

1.1 Object design trade-offs

Dopo la realizzazione del documento RAD (Requirement Analysis Document) e SDD (System Design Document), abbiamo descritto in linea di massima, quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti implementativi.

- *Prestazioni VS Costi*. Considerando il sistema che stiamo realizzando, possiamo dire che il non eccessivo budget a nostra disposizione ci ha consentito di realizzare il prodotto utilizzando materiale open source partendo da zero minimizzando così i costi e rendendo il sistema più che soddisfacente.
- *Interfaccia vs. Tempo di risposta*. Il tempo di risposta tra server e interfaccia sono più che sufficienti (rapidi) a soddisfare le esigenze dei vari utenti collegati al Sistema. Ovviamente maggiore sarà la grandezza del database e maggiore sarà il tempo di risposta e ricerca nel database.
- *Interfaccia vs. Easy-use*. L'interfaccia, grazie all'utilizzo delle form e di una impostazione semplice e intuitiva, permette un uso facile (Easy-Use) della gestione del sistema di database prodotti anche considerevolmente grande così da rendere immediata l'attività anche ai meno esperti col computer.
- *Costi vs. Mantenimento*. Grazie a un uso di materiale open source e l'utilizzo del linguaggio javadoc il sistema può essere facilmente modificato, implementato con nuove funzioni o corretto in presenza di errori.

1.2 Component off-the-shelf

Per lo sviluppo del sistema è previsto l'uso di diversi componenti off-the-shelf, ovvero componenti software messi a disposizione dal mercato che offrono pacchetti di soluzioni che possono essere utili a risolvere degli specifici problemi.

Le componenti previste per la realizzazione del sistema sono le seguenti:

- Bootstrap, un toolkit open source utilizzato per lo sviluppo di progetti responsive sul web. Il suo utilizzo è previsto insieme a quello di HTML, CSS e JavaScript per realizzare la struttura base della grafica.

- jQuery.js, una libreria JavaScript per applicazioni web il cui obiettivo è quello di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML. Il suo utilizzo ha lo scopo di ridurre la complessità del codice JavaScript durante l'implementazione.

- Javadoc, un tool che permette di generare la documentazione di un programma attraverso l'inserimento di tag specifici nel codice stesso; inoltre, prevede che la documentazione in javadoc produca un insieme di pagine HTML consultabili sul web. Il tool viene utilizzato per documentare il codice Java scritto dagli sviluppatori così da garantire maggiore comprensibilità del codice e rendere più facile la sua manutenzione a sviluppatori futuri.

- AJAX, una tecnica di sviluppo software per la realizzazione di applicazioni web interattive che si basa su uno scambio di dati in background fra web browser e server, che consente

l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. Il suo utilizzo consente di effettuare richieste asincrone al database.

- Selenium, una suite di tool utilizzati per automatizzare i test del sistema eseguendoli su un browser web. Esso viene utilizzato per eseguire le attività di testing di sistema.

- JUnit, un framework di programmazione Java che viene utilizzato per implementare i test di unità.

1.3 Interface documentation guidelines

Il sistema è multi-utente (può accedervi chiunque, visitatori, utenti e gestore).

- Al semplice visitatore, il sistema fornisce solamente la consultazione dei voli disponibili nelle tratte scelte oppure di consultare le offerte del sito.
- All'utente (visitatore loggato) il sistema fornisce la possibilità di completare l'acquisto dei biglietti scegliendo classe di viaggio, posto e bagaglio, poi di modificare i dati personali, visualizzare lo storico dei voli acquistati e effettuare il check-in online a partire da 3 giorni prima della partenza.
- Il gestore ha la possibilità di vedere tutti i voli in programma e ha la possibilità di inserire, modificare e/o cancellare i voli.

1.4 Definitions, acronyms, and abbreviations

- SDD: Software Design Document;
- ODD: Object Design Document
- RAD: Requirements analysis document.
- BROWSER: Explorer, Chrome, Mozilla.
- WebBrowser: Client (utente che accede al sistema)
- WebServer: Server su cui sono memorizzate le risorse.

1.5 References

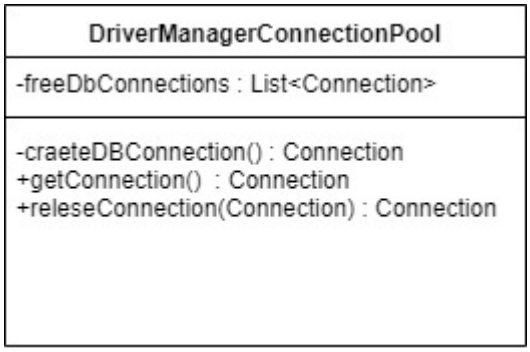
- Problem Statement
- RAD
- SDD
- https://www.easyjet.com/it
- Object-Oriented Software Engineering Using UML, Patterns, and Java™ Third Edition Bernd Bruegge & Allen H. Dutoit
- http://java.sun.com

2. Design Pattern

Singleton

Il pattern Singleton viene utilizzato quando si vuole garantire di avere un unico punto di accesso. Ad esempio, esso viene utilizzato quando si desidera avere un solo Window Manager oppure una sola Coda di Stampa oppure un unico accesso al database.

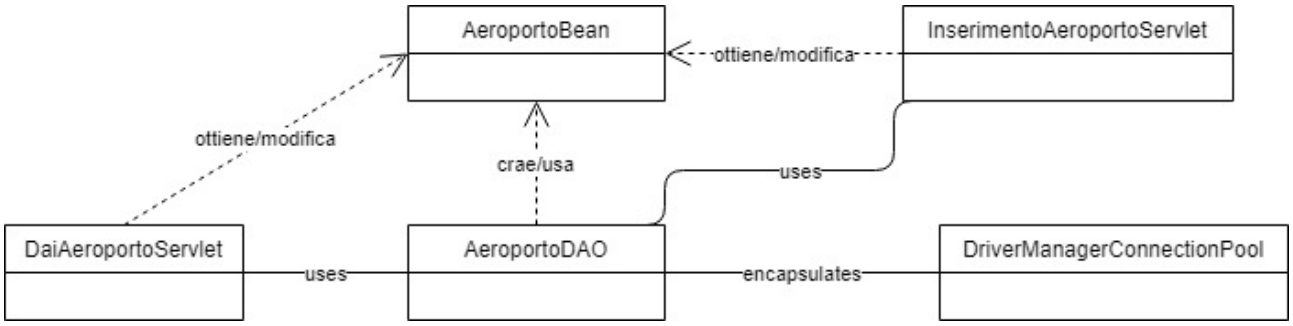
Un oggetto Singleton viene inizializzato nel momento in cui la classe viene invocata attraverso la definizione di un oggetto statico. La visibilità del suo costruttore viene modificata da public a private, così che non sia possibile istanziare la classe dall'esterno e fare in modo che soltanto la classe può istanziare sé stessa.



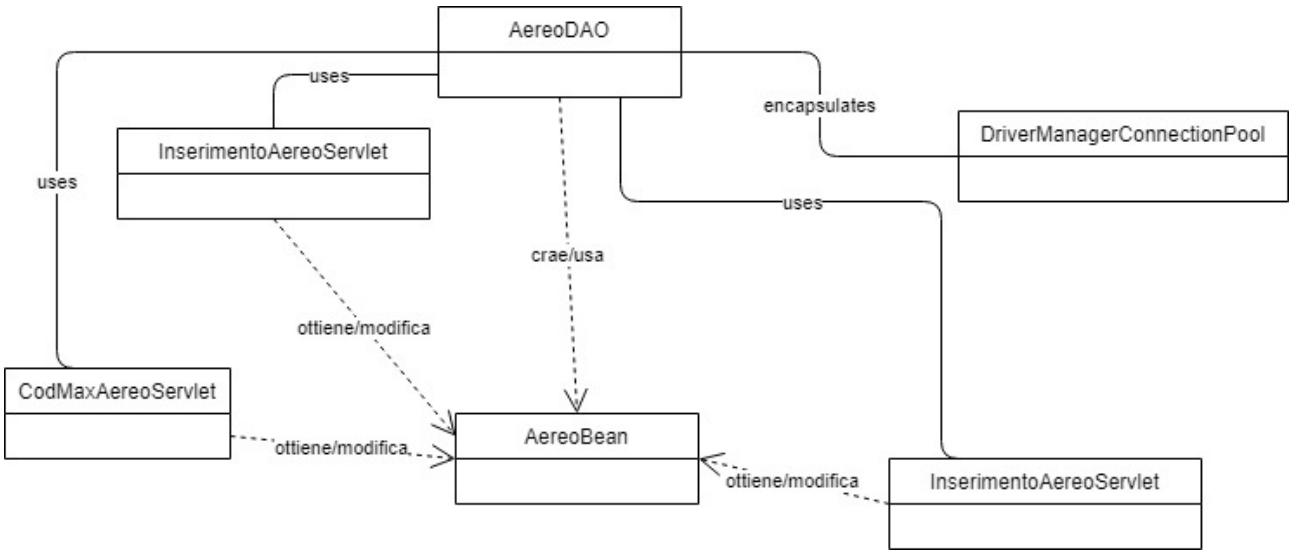
MVC
Siccome in fase di system design si è stabilito che il sistema proposto presenta l'architettura Model - View - Controller (o MVC), in fase di implementazione è previsto l'utilizzo dell'omonimo pattern.
Il pattern MVC viene utilizzato in un contesto dove l'applicazione deve fornire un'interfaccia grafica costituita da più schermate che mostrano vari dati all'utente, i quali devono risultare aggiornati in qualunque momento. Tale pattern viene spesso utilizzato nei casi in cui l'applicazione presenti una natura modulare basata sulle responsabilità, al fine di ottenere un sistema basato sulle componenti.
La soluzione fornita da questo pattern risulta essere particolarmente adatta al sistema proposto, poiché prevede che l'applicazione debba separare i componenti software che implementano le funzionalità di business dai componenti che implementano la logica di presentazione e di controllo, i quali utilizzano tali funzionalità.
I componenti previsti dal pattern MVC sono descritti di seguito nel dettaglio:
- Il Model definisce le regole di business per l'interazione con i dati, esponendo alla View ed al Control rispettivamente le funzionalità per l'accesso e l'aggiornamento dei dati.
- Il Control realizza la corrispondenza tra l'input dell'utente e i processi eseguiti dal Model, oltra a selezionare le schermate della View richieste ed implementare la logica di controllo dell'applicazione.

- La View si occupa della logica di presentazione dei dati

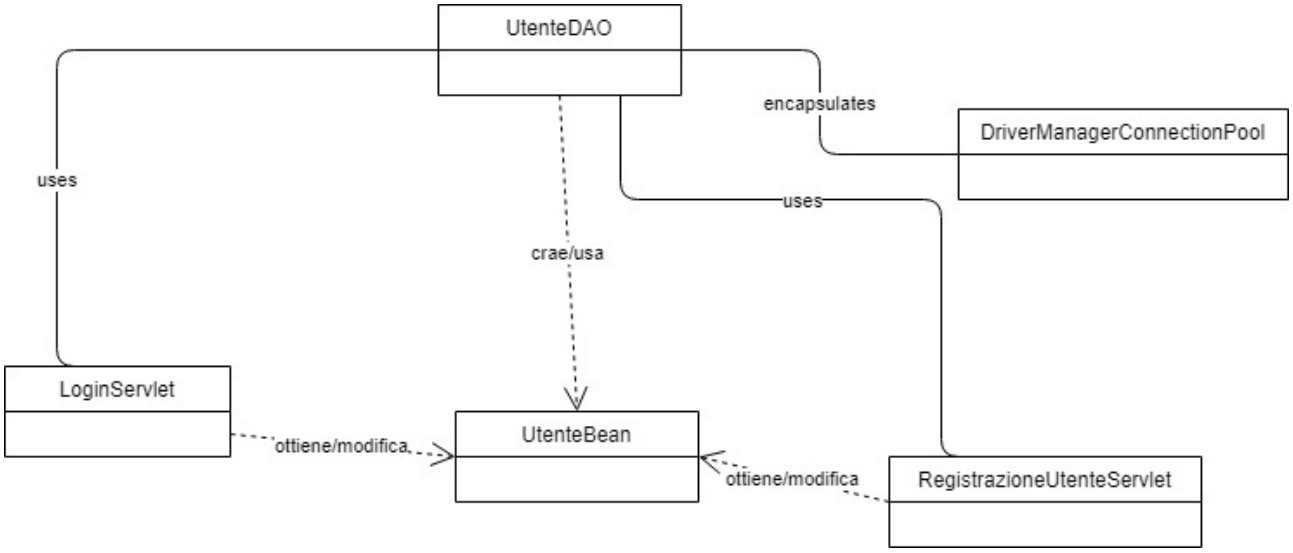
patter DAO per aeroporto



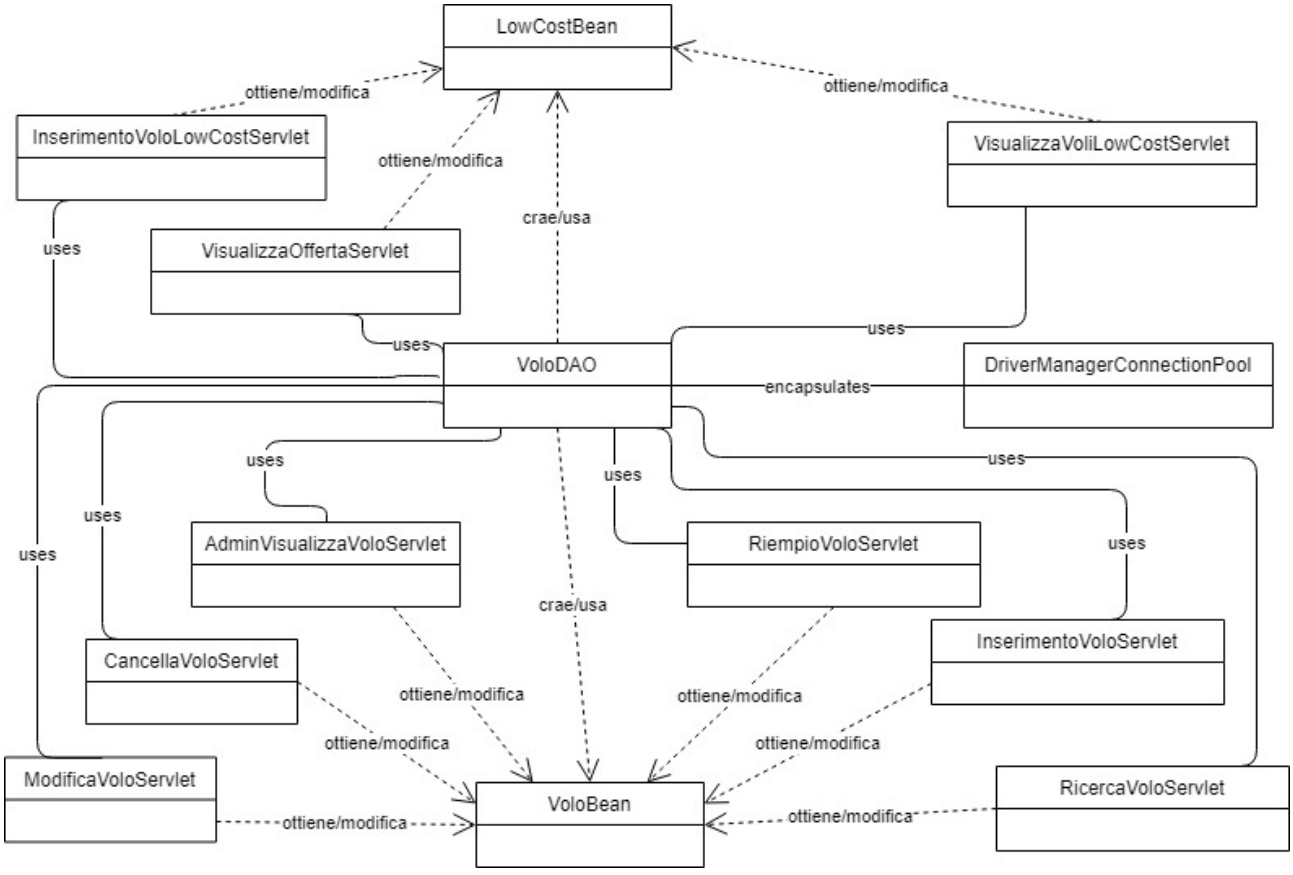
Pattern DAO per aereo



Pattern DAO per Utente



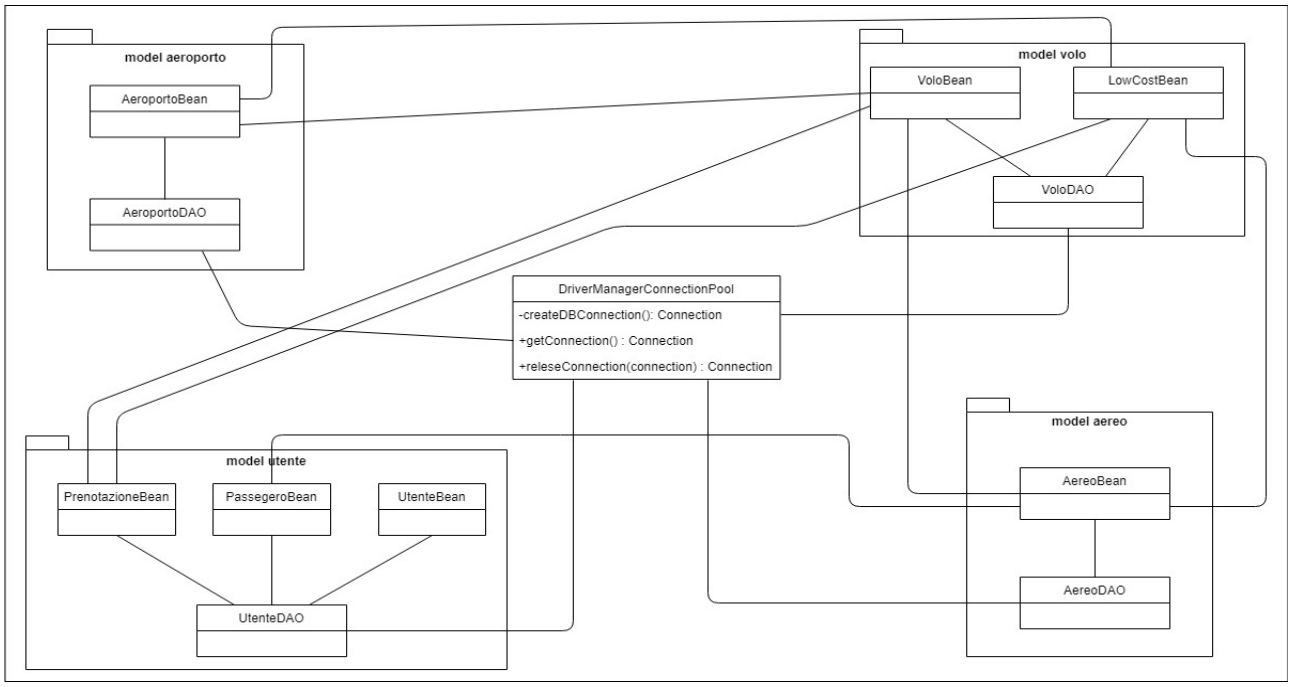
Pattern DAO per Volo

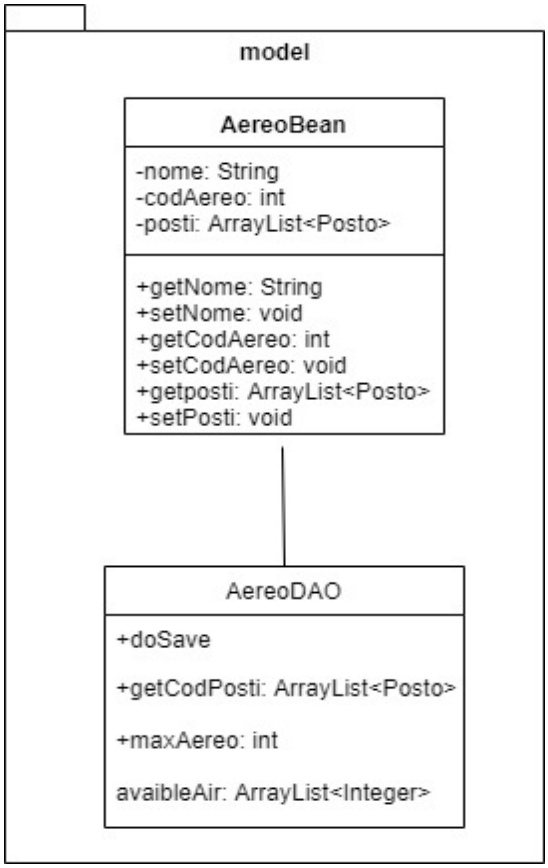


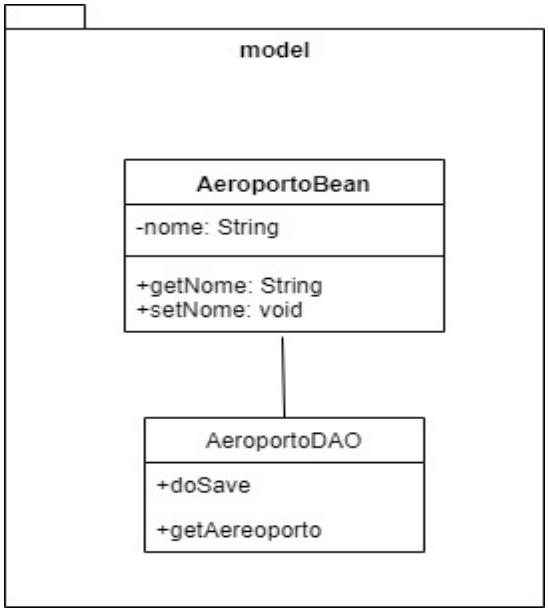
3.Package

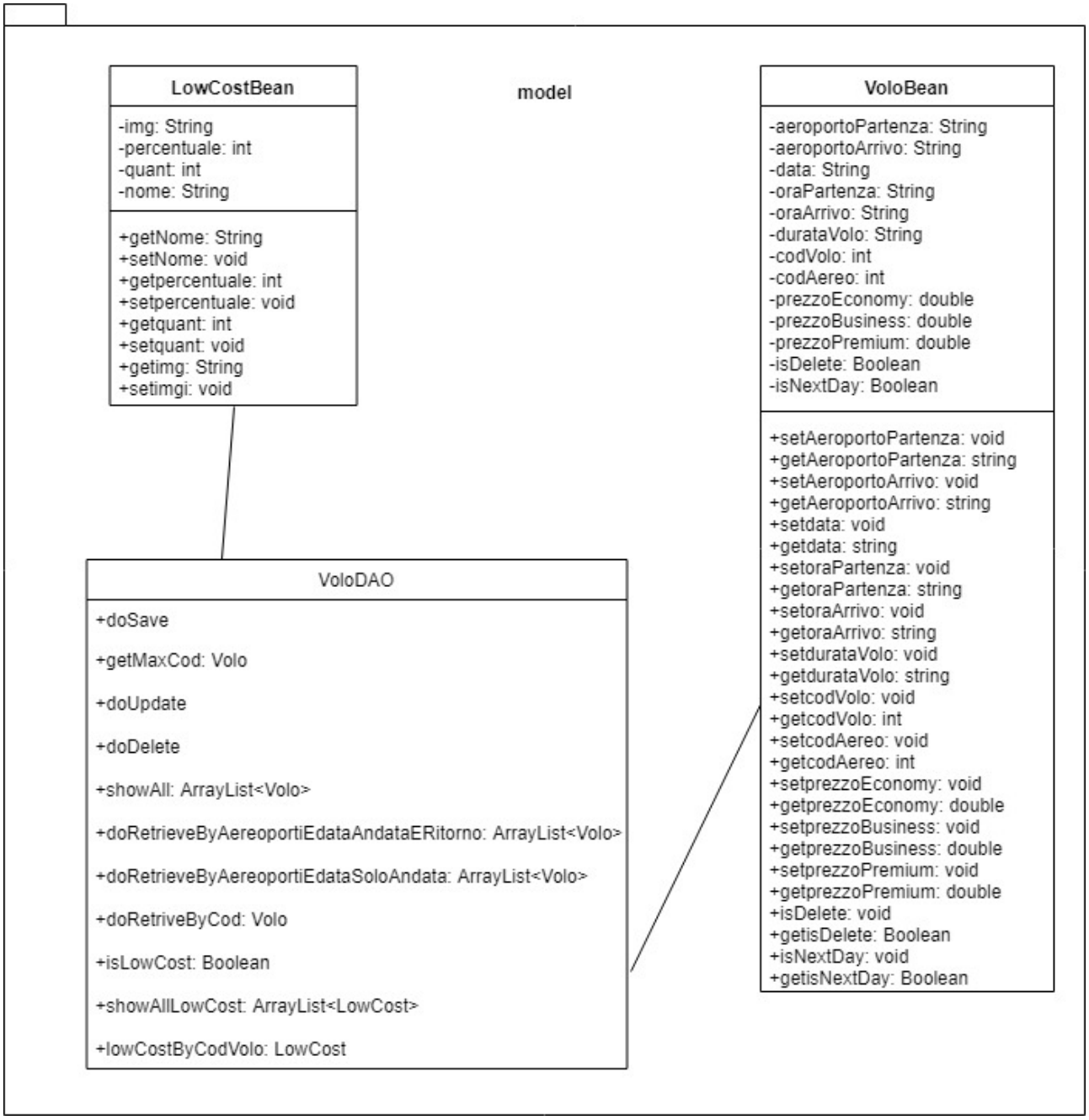
Di seguito, vengono illustrati i diagrammi dei package previsti dall'implementazione del sistema.

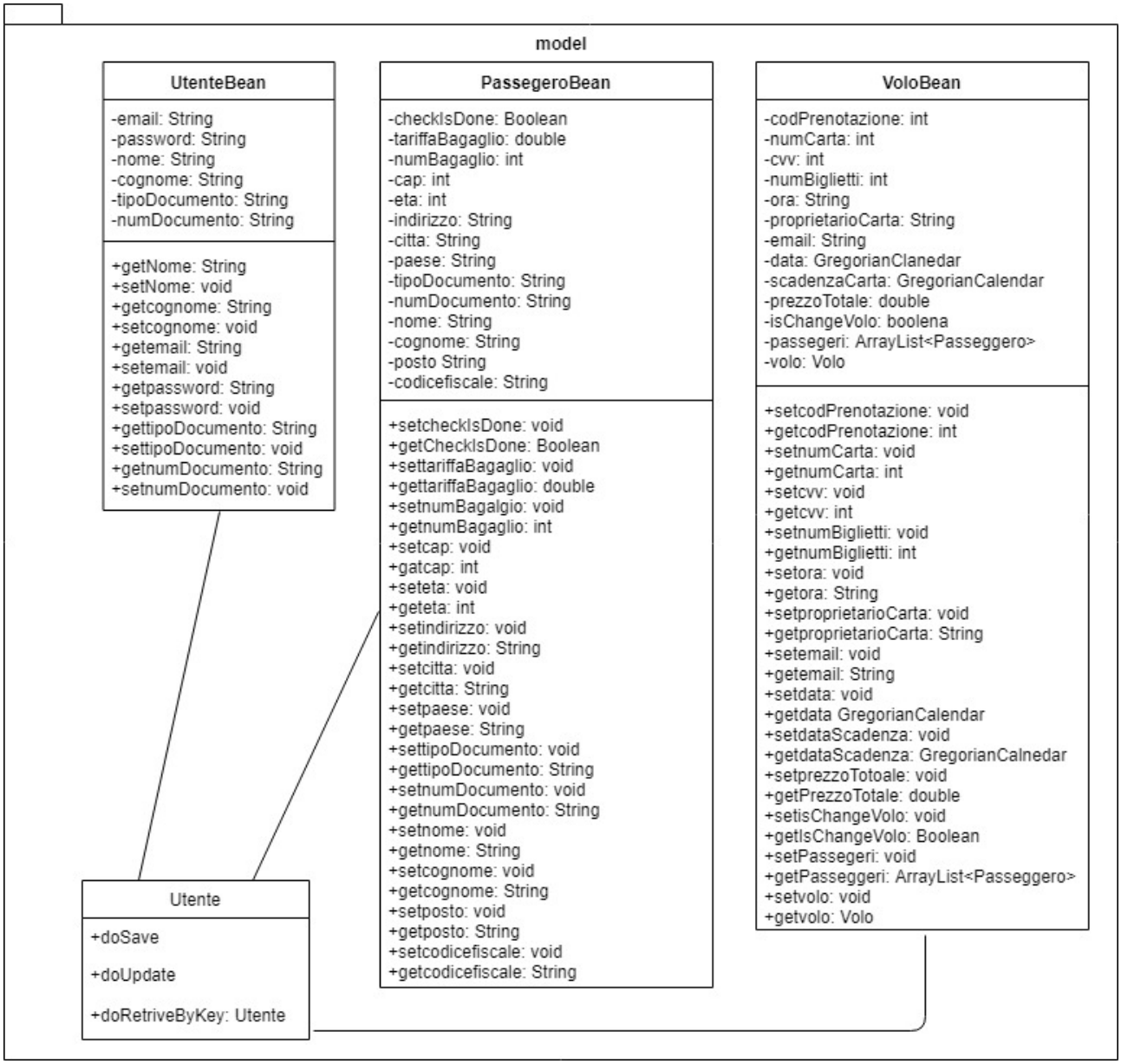
MODEL



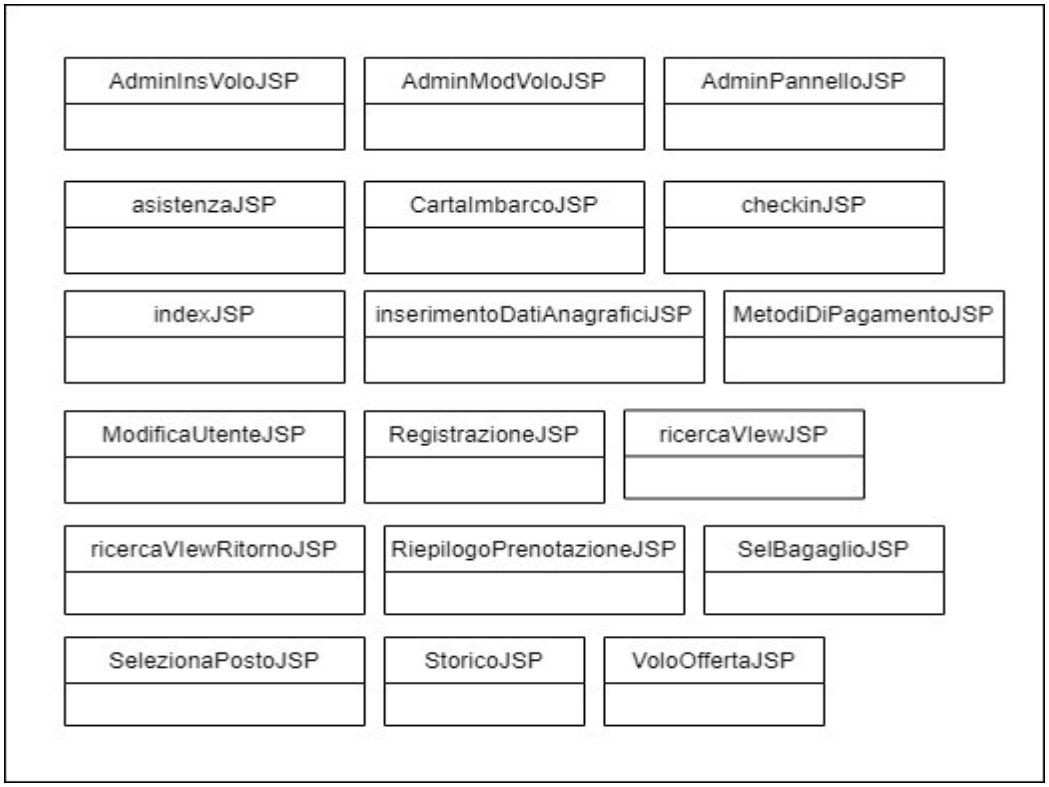

model
AereoBean
-nome: String
-codAereo: int
-posti: ArrayList<Posto>
+getNome: String
+setNome: void
+getCodAereo: int
+setCodAereo: void
+getposti: ArrayList<Posto>
+setPosti: void
AereoDAO
+doSave
+getCodPosti: ArrayList<Posto>
+maxAereo: int
avaibleAir: ArrayList<Integer>


model
AeroportoBean
-nome: String
+getNome: String
+setNome: void
AeroportoDAO
+doSave
+getAereoporto

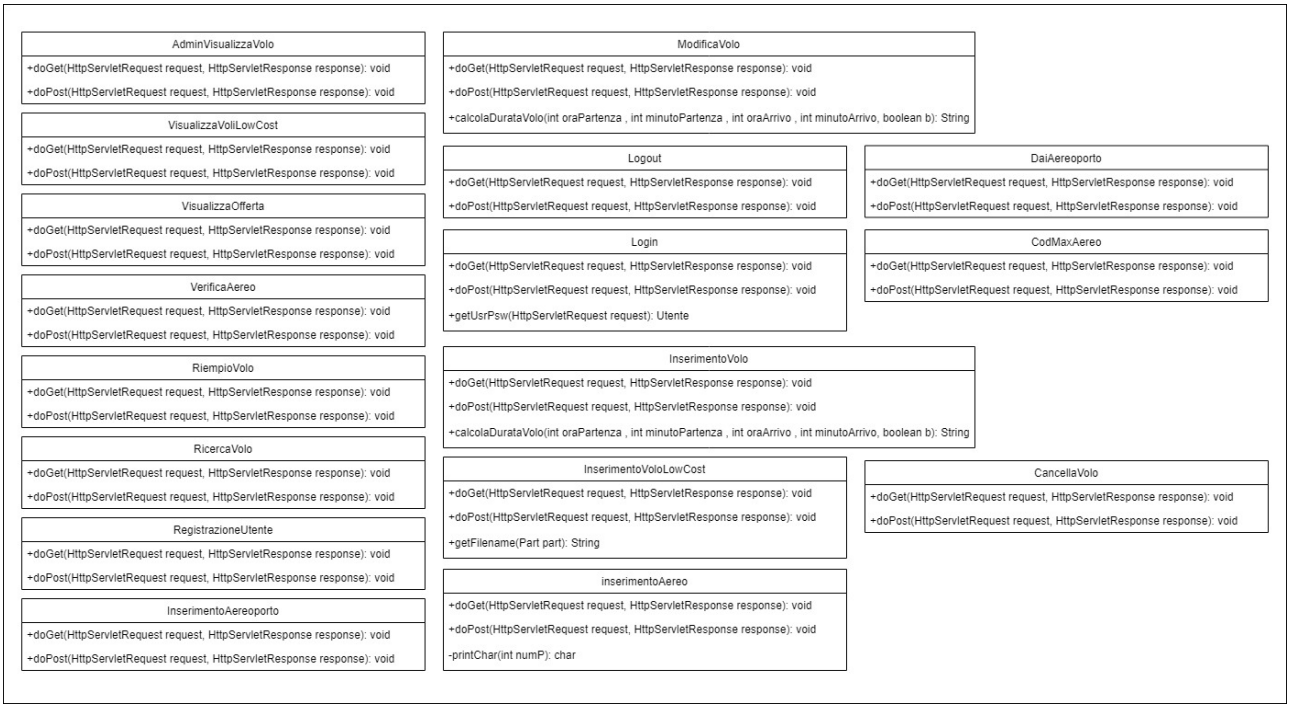

model
LowCostBean
-img: String
-percentuale: int
-quant: int
-nome: String
+getNome: String
+setNome: void
+getpercentuale: int
+setpercentuale: void
+getquant: int
+setquant: void
+getimg: String
+setimgi: void
VoloBean
-aeroportoPartenza: String
-aeroportoArrivo: String
-data: String
-oraPartenza: String
-oraArrivo: String
-durataVolo: String
-codVolo: int
-codAereo: int
-prezzoEconomy: double
-prezzoBusiness: double
-prezzoPremium: double
-isDelete: Boolean
-isNextDay: Boolean
+setAeroportoPartenza: void
+getAeroportoPartenza: string
+setAeroportoArrivo: void
+getAeroportoArrivo: string
+setdata: void
+getdata: string
+setoraPartenza: void
+getoraPartenza: string
+setoraArrivo: void
+getoraArrivo: string
+setdurataVolo: void
+getdurataVolo: string
+setcodVolo: void
+getcodVolo: int
+setcodAereo: void
+getcodAereo: int
+setprezzoEconomy: void
+getprezzoEconomy: double
+setprezzoBusiness: void
+getprezzoBusiness: double
+setprezzoPremium: void
+getprezzoPremium: double
+isDelete: void
+getisDelete: Boolean
+isNextDay: void
+getisNextDay: Boolean
VoloDAO
+doSave
+getMaxCod: Volo
+doUpdate
+doDelete
+showAll: ArrayList<Volo>
+doRetrieveByAereoportiEdataAndataERitorno: ArrayList<Volo>
+doRetrieveByAereoportiEdataSoloAndata: ArrayList<Volo>
+doRetriveByCod: Volo
+isLowCost: Boolean
+showAllLowCost: ArrayList<LowCost>
+lowCostByCodVolo: LowCost


model
UtenteBean
-email: String
-password: String
-nome: String
-cognome: String
-tipoDocumento: String
-numDocumento: String
+getNome: String
+setNome: void
+getcognome: String
+setcognome: void
+getemail: String
+setemail: void
+getpassword: String
+setpassword: void
+gettipoDocumento: String
+settipoDocumento: void
+getnumDocumento: String
+setnumDocumento: void
PassegeroBean
-checkIsDone: Boolean
-tariffaBagaglio: double
-numBagaglio: int
-cap: int
-eta: int
-indirizzo: String
-citta: String
-paese: String
-tipoDocumento: String
-numDocumento: String
-nome: String
-cognome: String
-posto String
-codicefiscale: String
+setcheckIsDone: void
+getCheckIsDone: Boolean
+settariffaBagaglio: void
+gettariffaBagaglio: double
+setnumBagalgio: void
+getnumBagaglio: int
+setcap: void
+gatcap: int
+seteta: void
+geteta: int
+setindirizzo: void
+getindirizzo: String
+setcitta: void
+getcitta: String
+setpaese: void
+getpaese: String
+settipoDocumento: void
+gettipoDocumento: String
+setnumDocumento: void
+getnumDocumento: String
+setnome: void
+getnome: String
+setcognome: void
+getcognome: String
+setposto: void
+getposto: String
+setcodicefiscale: void
+getcodicefiscale: String
VoloBean
-codPrenotazione: int
-numCarta: int
-cvv: int
-numBiglietti: int
-ora: String
-proprietarioCarta: String
-email: String
-data: GregorianClanedar
-scadenzaCarta: GregorianCalendar
-prezzoTotale: double
-isChangeVolo: boolena
-passegeri: ArrayList<Passeggero>
-volo: Volo
+setcodPrenotazione: void
+getcodPrenotazione: int
+setnumCarta: void
+getnumCarta: int
+setcvv: void
+getcvv: int
+setnumBiglietti: void
+getnumBiglietti: int
+setora: void
+getora: String
+setproprietarioCarta: void
+getproprietarioCarta: String
+setemail: void
+getemail: String
+setdata: void
+getdata GregorianCalendar
+setdataScadenza: void
+getdataScadenza: GregorianCalnedar
+setprezzoTotoale: void
+getPrezzoTotale: double
+setisChangeVolo: void
+getIsChangeVolo: Boolean
+setPassegeri: void
+getPasseggeri: ArrayList<Passeggero>
+setvolo: void
+getvolo: Volo
Utente
+doSave
+doUpdate
+doRetriveByKey: Utente

VIEW



CONTROL



4.Class interfaces

Di seguito, vengono specificate le interfacce di tutte le classi del pacchetto control del sistema.

Nome classe	AdminVisualizzaVolo
Descrizione	Si occupa della visualizzazione del volo all'interno della relativa pagina admin, quindi passa tutti i dati del volo.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	VisualizzaVoliLowCost
Descrizione	Si occupa della visualizzazione del VoloLowCost all'interno della relativa pagina index, quindi passa tutti i dati del volo segnato come LowCost.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	VisualizzaOfferta
Descrizione	Si occupa della visualizzazione della pagina offerta, quindi passa tutti i dati relativi a quella offerta.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	VerificaAereo
Descrizione	Si occupa di vedere se un aereo è disponibile in quel giorno e in quell'intervallo orario quando andiamo ad inserire un nuovo Volo.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	RiempioVolo
Descrizione	Serve a caricare i dati all'interno di un oggetto volo.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	RicercaVolo
Descrizione	Si occupa della ricerca di un volo dalla relativa form.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	RegistrazioneUtente
Descrizione	Si occupa della registrazione utente quindi prende i campi inseriti nella form e riempire l'oggetto utente.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	InserimentoAereoporto
Descrizione	Si occupa di inserire un nuovo aeroporto dalla form presente nel pannello dell'admin.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	Logout
Descrizione	Si occupa del logout dell'utente cancellando i cookie.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	Login
Descrizione	Si occupa della verifica dell'utente e al relativo login.
Pre-condizione	HttpServletRequest
Post-Condizione	Utente
Invarianti	

Nome classe	InserimentoVolo
Descrizione	Si occupa di inserire un nuovo volo tramite i valori presi nella form presenti nel pannello dell'admin.
Pre-condizione	Int(oraPartenza),Int(minutoPartenza), int(oraArrivo), Int(MinutoArrivo), boolena(nextDay)
Post-Condizione	String(durata volo)
Invarianti	

Nome classe	ModificaVolo
Descrizione	Si occupa di inserire un nuovo volo tramite i valori presi nella form presenti nel pannello dell'admin.
Pre-condizione	Int(oraPartenza),Int(minutoPartenza), int(oraArrivo), Int(MinutoArrivo), boolena(nextDay)
Post-Condizione	String(durata volo)
Invarianti	

Nome classe	InserimentoVoloLowCost
Descrizione	Si occupa dell'inserimento di un nuovo volo lowCost tramite la form presente nel pannello admin.
Pre-condizione	Part(path file)
Post-Condizione	String(filename)
Invarianti	

Nome classe	InserimentoAereo
Descrizione	Si occupa di inserire un nuovo aereo quindi prendei dati inseriti nella form presente nel pannello dell'admin.
Pre-condizione	Int
Post-Condizione	Char
Invarianti	

Nome classe	DaiAereoporto
Descrizione	Si occupa di prendere tutti gli aeroporti presenti e disponibili ed inserirli nei relativi campi.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	CodMaxAereo
Descrizione	Si occupa di calcolare l'ultimo codice dell'aereo inserito per poter calcolare il codice del nuovo da inserire.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	CancellaVolo
Descrizione	Si occupa della cancellazione di un volo inserito tramite la form presente nel pannello admin.
Pre-condizione	
Post-Condizione	
Invarianti	

Nome classe	ModificaUtente
Descrizione	Si occupa della modifica dei dati di un utente già registrato.
Pre-condizione	
Post-Condizione	
Invarianti	