UNIVERSITA' DEGLI STUDI DI PALERMO
FACOLTA' DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

ODD
OBJECT DESIGN DOCUMENT

Progetto: ***Magic Travel***

Descrizione:
Il documento elenca le scelte finali prima dell'implementazione la quale deve rispettare i modelli di analisi, i requisiti non funzionali e i criteri di design.

A chi è rivolto:
Analisti e sviluppatori.

Membri del gruppo:

Sciarrabba Enrico Maria
Solaro Giuliano
Taglialavoro Francesco
Zafonte Francesca

Sommario

1 Introduzione

In questa breve introduzione verranno descritti i mezzi utilizzati per l'implementazione del progetto riguardo la creazione dell'interfaccia utente e la gestione del server. Verranno inoltre illustrati i cambiamenti principali apportati all'analisi presentata nel RAD, per permetterne la successiva implementazione.

La suddivisione del file rispecchierà la suddivisione in casi d'uso fatta durante l'analisi.

1.1 Implementazione

Poiché si utilizza un'architettura client-server si è cercato di suddividere le due parti fin da subito.

Per quanto riguarda l'interfaccia utente è stata utilizzata la libreria "react" di javascript la quale permette lo sviluppo di un frontend complesso in modo semplice e veloce grazie all'utilizzo di componenti. Questi ultimi saranno ricavati proprio dalle classi qui descritte. I file ottenuti saranno quindi in formato .js e presenteranno al loro interno codici in jsx il quale permette di unire un linguaggio di marcatura come html a un linguaggio interpretato come js.

Per quanto riguarda il backend è stato invece utilizzato il framework expressjs basato su nodejs. Anche le classi lato server che gestiscono tutte la comunicazione con il db verranno tradotte in codice javascript.

1.2 Cambiamenti principali

Tutte le boundary che rappresentavano pulsanti sono state aggregate alle corrispondenti finestre. Per questo motivo, nella maggior parte dei casi, quando i pulsanti vengono utilizzati per inizializzare il caso d'uso questi sono presenti in classe esterne.

Durante l'analisi tutti gli annunci e le prenotazioni sono stati inclusi in un unico grande insieme che adesso verrà scisso in due: case vacanza e b&b. Nella maggior parte dei casi ad un metodo presente nei class diagram ne corrisponderanno due.

I metodi click, conferma e inizia presenti solitamente nelle boundary che rappresentano dei pulsanti, sono stati eliminati per ovvie ragioni (rappresentavano un'interazione dell'utente con il sistema).

Da ogni caso d'uso verrà generata una classe contenente i metodi principali per gestire il frontend. Solitamente questa è derivata dalla corrispondente classe di tipo finestra. Sarà anche affiancata da altre classi per rifinirne le funzioni.

Tutti i metodi di tipo creaFinestra sono stati eliminati poiché in fase di implementazione verranno rappresentati da link che genereranno l'apertura di nuove finestre.

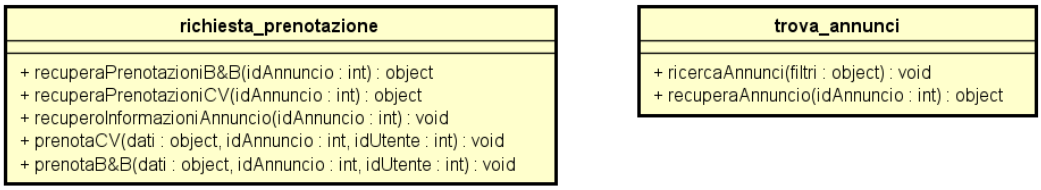
Tutti i metodi stampaEsito verranno sostituiti e specializzati secondo lo specifico caso d'uso d'appartenenza.

Tali modifiche non verranno più illustrate nel paragrafo successivo ma verranno sottintese.

2 Package

2.1 Ricerca e Prenota

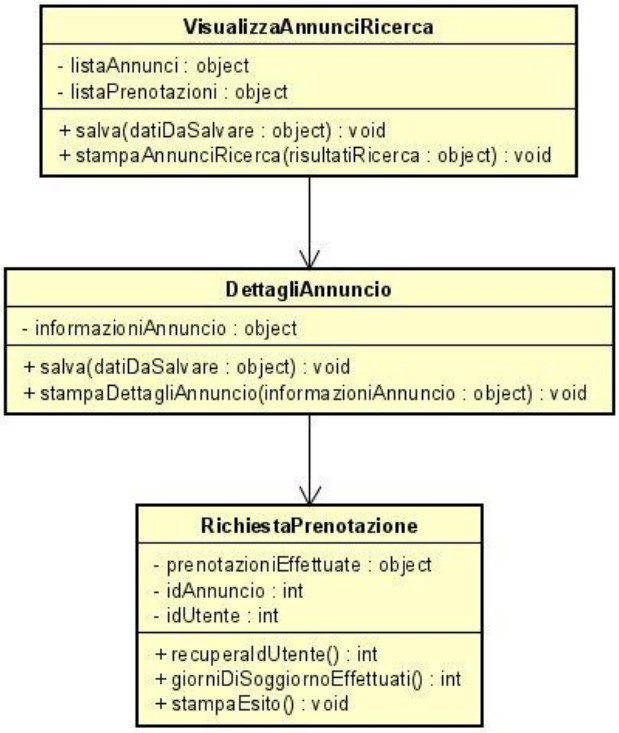
Lato Server: modifica della boundary DBMSRicercaEPrenota:



Questi saranno i metodi per gestire la comunicazione tra server e il database. Oltre ai cambiamenti già dichiarati nell'introduzione notiamo un'altra suddivisione logica in due sottoclassi. Attraverso questa è possibile distinguere due fasi principali: Ricerca, in cui andiamo ad inglobare anche DettagliAnnunciRicerca, rappresentata da trova_annunci; Prenotazione rappresentata dalla classe richiesta_prenotazione.

- In *DBMSRicercaEPrenota* il metodo 'ricerca' diventa 'ricercaAnnunci' mentre il metodo 'recuperaDettagliAnnuncio' viene sostituito da 'recuperaAnnuncio'.
  Eventuali differenze tra un annuncio di casa vacanze e un annuncio B&B verranno poi eventualmente valutate e gestite in fase di implementazione.
- In *richiesta_prenotazione* il metodo 'creaRichiesta' appartenente alla boundary DBMSRicercaEPrenota viene sostituito con due metodi 'prenotaCV' e 'prenotaB&B' che si occuperanno, oltre che ad effettuare la prenotazione, di inserire nel database i dati relativi ai partecipanti ad un soggiorno, nel caso in cui non siano già presenti nel database.
  I metodi 'recuperoInformazioniAnnuncio', 'recuperaPrenotazioniB&B' e 'recuperaPrenotazioniCV' vengono aggiunti per poter gestire, in fase di implementazione, tutti i controlli necessari.

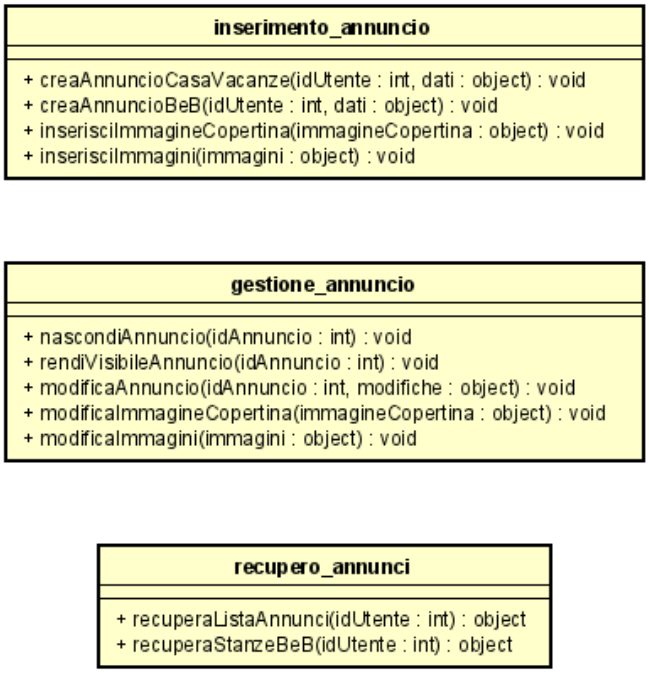
Lato Client:



- *VisualizzaAnnunciRicerca* e *Dettagli Annuncio* ereditano il metodo 'salva', necessario per memorizzare i dati recuperati dal database, dalle rispettive entity Dati e DettagliAnnuncio. Entrambe posseggono dei metodi per stampare i dati recuperati dal database, in particolare 'stampaAnnunciRicerca' e 'stampaDettagliAnnuncio' che si occuperanno mostrarli secondo logiche che verranno scelte in fase implementativa.
- Sono stati aggiunti degli attributi come listaAnnunci, listaPrenotazioni, informazioniAnnuncio, prenotazioniEffettuate, idAnnuncio, idUtente dove sarà possibile memorizzare i dati recuperati dal database per poi poterli elaborare.
  La RichiestaPrenotazione eredita inoltre il metodo 'recuperaIdUtente' dalla entity Utente.
- Sono stati inoltre aggiunti i metodi 'stampaEsitoRicerca' che specializza il metodo stampaEsito e 'giorniDiSoggiornoEffettuati' utilizzato per effettuare il controllo sul massimo numero di giorni in cui è possibile prenotare una casa vacanze.

2.2 Gestione Annunci

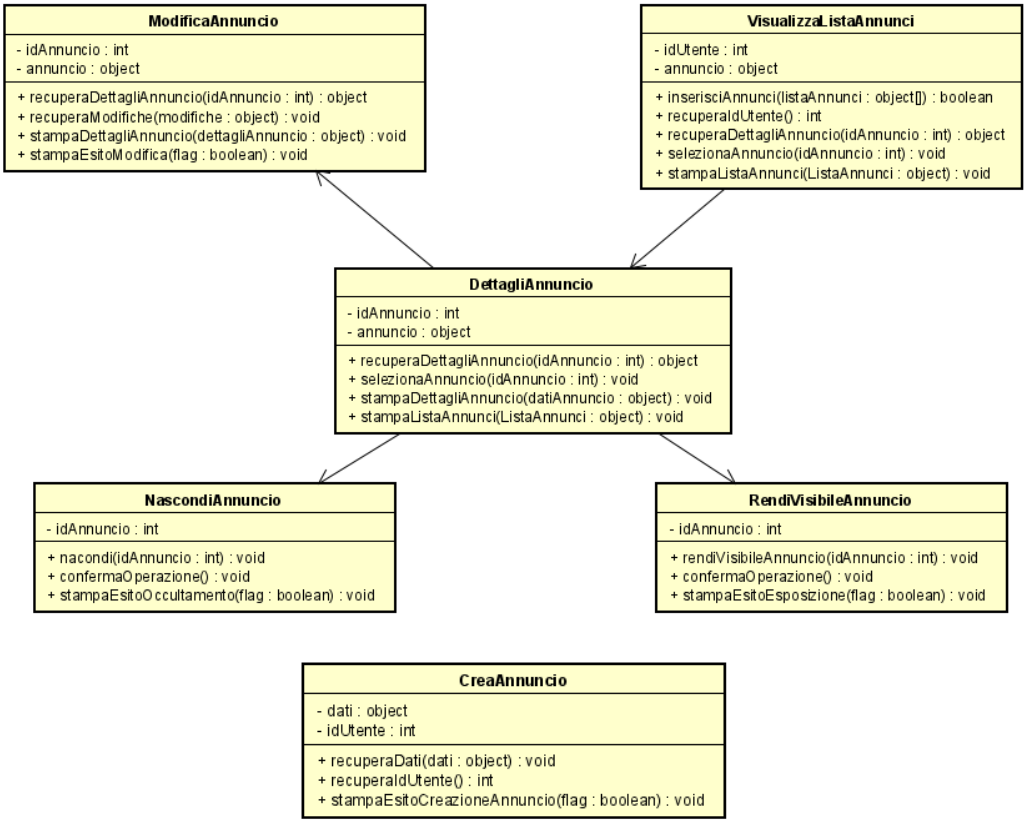
Lato Server: modifica della boundary DBMSGestioneAnnunci.



Oltre a quanto descritto nell'introduzione sono state apportate altre modifiche:

- La boundary DBMSGestioneAnnunci è stata divisa il 3 classi:
  - *inserimento_annuncio* che fornisce i metodi utili all'inserimento dell'annuncio e delle immagini;
  - *gestione_annuncio* che fornisce i metodi per modificare, nascondere e rendere visibile l'annuncio;
  - *recupero_annunci_proprietario* che consente di recuperare tutti gli annunci relativi ad un proprietario.
- In *inserimento_annuncio* e *gestione_annuncio* sono stati aggiunti i metodi necessari all'inserimento e alla modifica delle immagini relative ad un annuncio. Essendo queste operazioni importanti ai fini implementativi, risulta necessario inserirle nel corrente documento.

Lato Client:

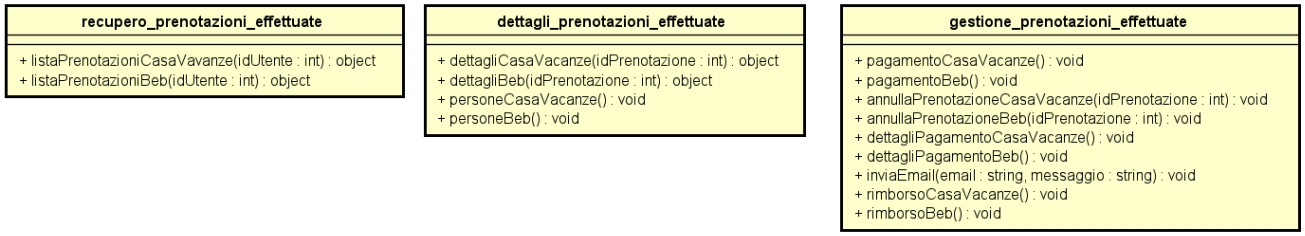


Nel diagramma sono state apportate delle modifiche riguardanti i metodi e degli accorpamenti di più classi:

- *CreaAnnuncio:* i metodi 'riceviDati' e 'inserisciDati', presenti nel class diagram del RAD, sono stati inglobati in un unico metodo generale 'recuperaDati' che si occupa di recuperare i dati inseriti nel form di creazione annuncio; inoltre è stato ereditato dalla entity utente il metodo 'recuperaIdUtente' che consente di recuperare l'id dell'utente autenticato.
- *ListaAnnunci:* sono stati accorpati gli oggetti entity nell'unica classe, con conseguente inserimento dei metodi 'inserisciAnnunci' per inserire la lista degli annunci recuperati dal database, 'recuperaDettagliAnnuncio' per recuperare i dettagli di un qualsiasi annuncio e 'recuperaIdUtente' per recuperare l'id dell'utente autenticato.
- *DettagliAnnuncio:* è stato inserito il nuovo metodo 'recuperaDettagliAnnuncio' ereditato dalla entity listaAnnunci.
- *ModificaAnnuncio:* è stato inserito il nuovo metodo 'recuperaDettagliAnnuncio' ereditato dalla entity listaAnnunci.
- *NascondiAnnuncio:* i metodi 'conferma' e 'annulla' operazione, presenti nel class diagram del RAD, sono stati inglobati in un unico metodo 'confermaOperazione' in quanto sufficiente a rappresentarli entrambi. Il metodo 'stampaEsitoOccultamento' è invece riferito all'operazione di nascondere l'annuncio.
- *RendiVisibileAnnuncio:* i metodi 'conferma' e 'annulla' operazione, presenti nel class diagram del RAD, sono stati inglobati in un unico metodo 'confermaOperazione' in quanto sufficiente a rappresentarli entrambi. Il metodo 'stampaEsitoEsposizione' è invece riferito all'operazione di rendere visibile l'annuncio.

2.3 Prenotazioni Effettuate

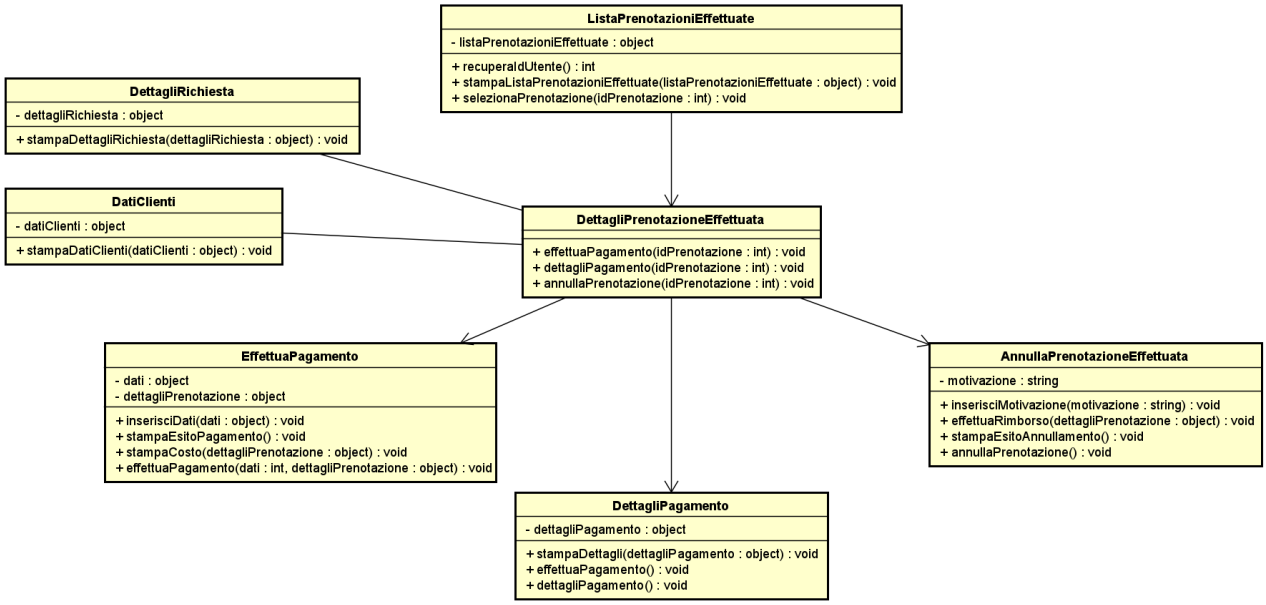
Lato Server: modifica della boundary DBMSPrenotazioniEffettuate.



Oltre ai cambiamenti già dichiarati nell'introduzione sono state apportate le seguenti modifiche:

- la boundary DBMSPrenotazioniEffettuate è stata suddivisa in tre classi:
  - *recupero_prenotazioni_effettuate* si occupa di richiedere la lista delle prenotazioni effettuate,
  - *dettagli_prenotazioni_effettuate* consente di recuperare i dettagli di ogni singola prenotazione,
  - *gestione_prenotazioni_effettuate* si occupa dei metodi necessari per effettuare l'annullamento, il pagamento di una prenotazione o visualizzare i dettagli di un pagamento effettuato.
- in *gestione_prenotazioni_effettuate* sono stati aggiunti i metodi necessari per effettuare il rimborso ed il metodo inviaEmail ereditato dall'oogetto control. Quest'ultimo sarà richiamato ogni volta che bisognerà inoltrare una e-mail ad un proprietario
- il metodo che gestiva le transazioni è stato inglobato dalle corrispondenti funzioni del caso d'uso
- il metodo che gestiva l'aggiornamento dello stato di una prenotazione è stato inglobato dalle corrispondenti funzioni del caso d'uso

Lato client:

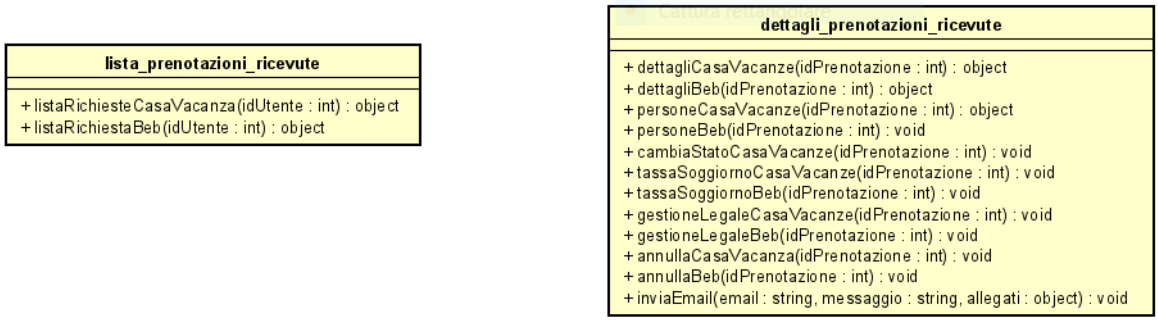


- In ListaPrenotazioniEffettuate sono stati inglobati i metodi per stampare la lista secondo precise logiche che verranno scelte in fase di implementazione (quali per esempio l'ordine di stampa), il recupero dell'idUtente (metodo acquisito dall'entity) e la possibilità di selezionare i dettagli di una prenotazione.
- In DettagliPrenotazioneEffettuata è stata apportata una separazione in due classi con i rispettivi metodi stampa.
- I metodi effettuaPagamento, dettagliPagamento e annullaPrenotazione saranno implementati come pulsanti per l'avvio dei casi d'uso.

Come detto prima, tutti i metodi la cui funzione consisteva nell'invio di e-mail sono state assorbiti da funzioni che richiamano il metodo inviaEmail di gestione_prenotazioni_effettuate.

2.4 Prenotazioni Ricevute

Lato Server: modificando la boundary DBMSPrenotazioniRicevute.

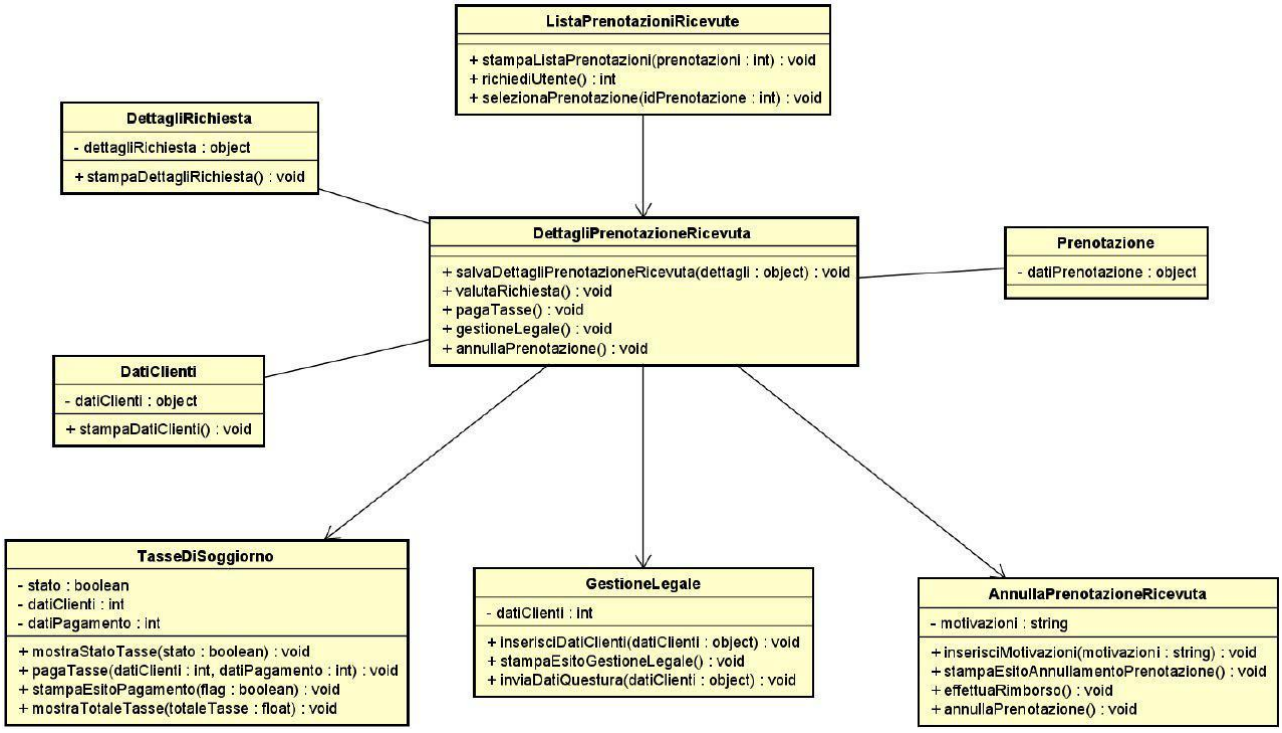


Questi saranno i metodi per gestire la comunicazione tra server e il database.

- Oltre ai cambiamenti già dichiarati nell'introduzione notiamo un'altra suddivisione in due sottoclassi in cui la prima si occupa di richiedere la lista delle prenotazioni ricevute mentre la seconda dei dettagli di ogni singola prenotazione.
- Troviamo anche un nuovo metodo inviaEmail ereditato dalla control. Questo metodo sarà richiamato ogni volta che servirà inoltrare una email alla questura, all'ufficio del turismo o ad un generico cliente.

Infine tutti i metodi che gestivano le transazioni sono stati inglobati dalle corrispondenti funzioni del caso d'uso.

Lato Client:



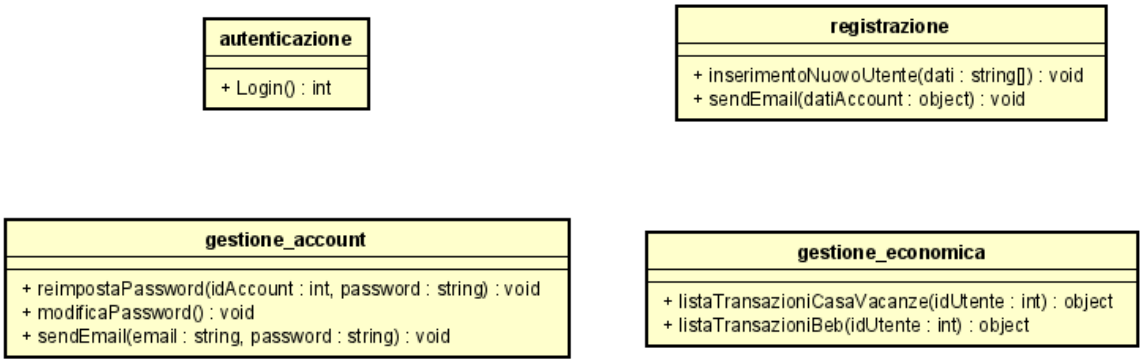
Oltre a quanto descritto nell'introduzione sono state apportate altre modifiche:

- In ListaPrenotazioniRicevute sono stati inglobati i metodi per stampare la lista secondo precise logiche che verranno scelte in fase di implementazione (quali per esempio l'ordine di stampa), la richiesta dell'idUtente (metodo acquisito dall'entity) e la possibilità di selezionare i dettagli di una prenotazione.
- In DettagliPrenotazioni ricevute è stato importato il metodo dalla entity per salvare i dettagli in memoria. Per questo è stata creata anche una classe prenotazione che contiene le informazioni e i dettagli di una prenotazione. Poiché le operazioni che verranno svolte sui dettagli si dividono principalmente in due gruppi, quelle che agiscono sui dati dei clienti e quelle che agiscono sui dettagli della prenotazione, è stata apportata questa separazione in due classi con i rispettivi metodi di stampa.
- Data l'estrema semplicità del caso d'uso valutazione richiesta si è deciso il collasso in un semplice metodo valutaRichiesta il quale andrà a richiamare i metodi presenti in dettagli_prenotazioni_ricevute.
- Gli altri metodi (pagaTasse, gestioneLegale e annullaPrenotazione) saranno implementati come pulsanti o link per l'avvio dei casi d'uso.
- In GestioneLegale il metodo inserisciDatiClienti permette la gestione dei documenti dei clienti.

Come detto prima tutti i metodi la cui funzione consisteva nell'invio di email sono state assorbiti da funzioni che richiamavano il metodo inviaEmail di dettagli_prenotazioni_ricevute.

2.5 Gestione Account

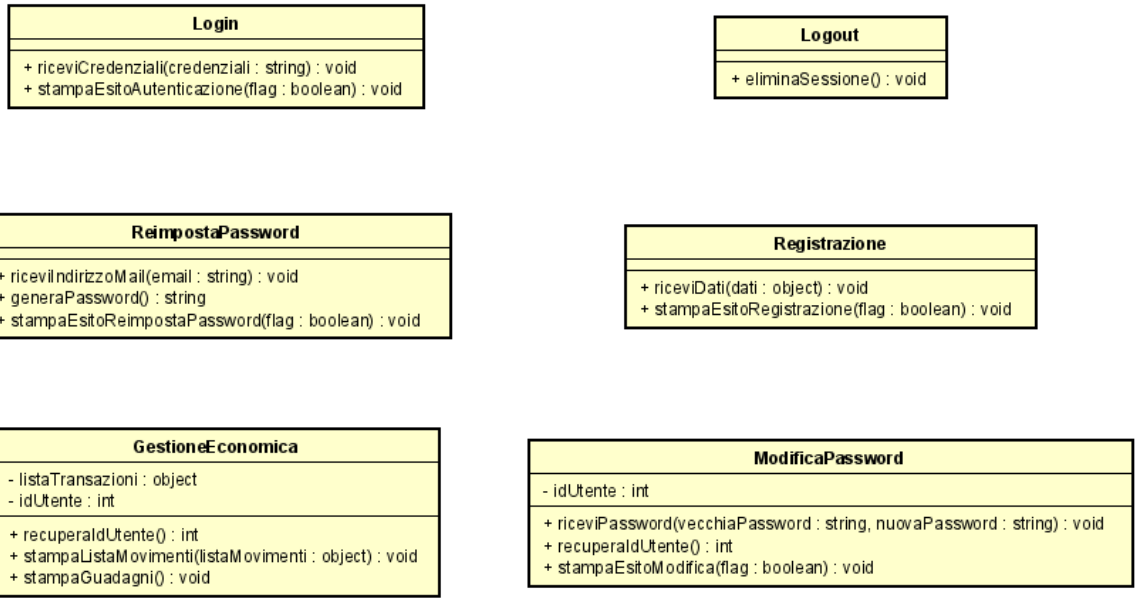
Lato Server: modifica della boundary DBMSGestioneAccount.



Oltre a quanto descritto nell'introduzione sono state apportate altre modifiche:

- La boundary DBMSGestioneAccount è stata divisa in 4 classi:
  - *autenticazione* che consente di effettuare il login;
  - *registrazione* che permette ad un utente di creare un account;
  - *gestione_account* che permette di modificare o reimpostare la password;
  - *gestione_economica* che permette di tener traccia di tutte le transazioni relative all'account.
- Nella classe *autenticazione* il metodo 'Login' sostituisce il metodo 'recuperaCredenziali' della boundary DBMSGestioneAccount.
- In *registrazione* il metodo 'inserimentoNuovoUtente' sostituisce 'controlloEsistenzaDati' e 'salvaDati' e si occupa di compiere tutte le operazioni necessarie alla creazione dell'account. Il metodo 'sendEmail' viene ereditato dalla control sistemaRegistrazione e si occupa di inviare una mail di riepilogo.
- In *gestione_account* il metodo 'reimpostaPassword' sostituisce 'verificaIndirizzoMail' e 'aggiornaPassword'.
  Il metodo 'modificaPassword' sostituisce 'verificaVecchiaPassword' e 'aggiornaPassword'. Inoltre è stato inserito il metodo sendEmail ereditato dalla control sistemaReimpostaPassword e si occupa di inviare una mail contenente le modifiche apportate.
- In *gestione_economica* i metodi 'listaTransazioni' di casa vacanze o b&b derivano dal metodo 'listaMovimenti' presente nel documento di analisi.

Lato Client:



- In *Logout* il metodo ‘eliminaSessione’ sostituisce il metodo elimina della entity utente e consente di eliminare i dati relativi alla sessione dell’utente.
- In *GestioneEconomica* e in *ModificaPassword* il metodo ‘recuperaIdUtente’ viene ereditato dalla entity utente.
- In *ModificaPassword* i metodi ‘inserisciPassword’ e ‘riceviPassword’ vengono inglobati in un unico metodo ‘riceviPassword’.
- In *Login* i metodi ‘inserisciCredenziali’ e ‘riceviCredenziali’ vengono inglobati in un unico metodo ‘riceviCredenziali’.
- In *Registrazione* i metodi ‘inserisciDati’ e ‘riceviDati’ vengono inglobati in un unico metodo ‘riceviDati.
- In *ReimpostaPassword* i metodi ‘inserisciIndirizzoMail’ e ‘riceviIndirizzoMail’ vengono inglobati in un unico metodo ‘riceviIndirizzoMail’.