Ingegneria del Software

Corso di laurea Magistrale in Ingegneria Informatica

Travel Easy: terza iterazione

Pisana Giacomo, Scarpa Eliana, Serio Giuliana

Requisiti

Introduzione

TravelEasy è un'applicazione software pensata per la gestione completa di un'agenzia viaggi.

I clienti possono utilizzare il sistema per cercare pacchetti vacanza inserendo date, destinazioni e preferenze personali, e procedere direttamente alla prenotazione e al pagamento dei servizi scelti.

Gli operatori dell'agenzia, invece, utilizzano TravelEasy per inserire e aggiornare le offerte di viaggio e visualizzare le prenotazioni effettuate dai clienti e monitorare lo stato dei pagamenti.

TravelEasy permette di creare diversi tipi di pacchetti vacanza, ciascuno composto da una combinazione di servizi. Ad esempio, un pacchetto "Weekend Romantico a Parigi" può comprendere volo andata e ritorno, due notti in hotel a quattro stelle con colazione inclusa e una crociera sulla Senna. Un pacchetto "Avventura in Islanda" può invece includere volo, cinque notti in hotel e escursioni guidate. Gli operatori possono aggiungere periodicamente nuovi tipi di pacchetti al catalogo.

Una prenotazione può riguardare anche più persone.

Casi d'uso

Nella seconda iterazione si considerino i seguenti casi d'uso.

UC4: Modifica prenotazione

Attore Primario: Cliente

Scenario Principale:

1. Il sistema mostra l'elenco delle prenotazioni (UC10)
2. Il cliente clicca su "Modifica Viaggiatori"
3. Il sistema mostra le opzioni modificabili (viaggiatori)
4. Il cliente modifica i dati desiderati
5. Il sistema calcola l'eventuale differenza di prezzo (supplemento o rimborso)
6. Il sistema mostra il nuovo totale
7. Il cliente conferma le modifiche
8. Se necessario, il sistema gestisce il pagamento aggiuntivo o il rimborso parziale
9. Il sistema aggiorna la prenotazione con i nuovi dati

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**:

1. Il Cliente aggiorna la pagina
2. Il Cliente effettua nuovamente l'accesso

2a. La prenotazione non è modificabile (partenza entro 7 giorni):

1. Il sistema mostra messaggio informativo

2b. Il Cliente clicca su "Modifica Prenotazione"

1. Il sistema mostra le opzioni modificabili (date viaggio)
2. Il cliente modifica i dati desiderati
3. Il sistema richiama il caso d'uso UC2
4. Si riprende dal passo 5

8a. Pagamento supplemento fallito:

1. La modifica viene annullata, la prenotazione originale rimane valida

8b. Rimborso fallito:

1. La modifica viene annullata, la prenotazione originale rimane valida

Regole di dominio

La modifica della prenotazione può essere effettuata entro 7 giorni dalla data di partenza.

UC5: Elimina prenotazione

Attore Primario: Cliente

Scenario Principale:

1. Il sistema mostra l'elenco delle prenotazioni attive (UC10)
2. Il cliente seleziona la prenotazione da eliminare
3. Il cliente conferma l'eliminazione
4. Il sistema indica la somma del rimborso
5. Il sistema conferma il rimborso
6. Il sistema aggiorna l'elenco delle prenotazioni e il saldo ore a carico del cliente

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

1. Il Cliente aggiorna la pagina
2. Il Cliente effettua nuovamente l'accesso

2a. La prenotazione non è eliminabile (partenza entro 2 giorni)

5a. Rimborso fallito: La prenotazione originale rimane valida

Regole di dominio

La prenotazione può essere eliminata entro 2 giorni dalla data di partenza, in modo che l'operatore possa procedere con il check-in. Il rimborso sarà del 100% se la prenotazione viene annullata entro 7 giorni dalla partenza, del 50% se viene effettuata entro 2 giorni.

UC12: Gestione assistenze speciali

Attore Primario: Cliente

Scenario Principale:

1. Il Sistema riceve richiesta dal caso d'uso **UC3: Prenotazione pacchetto vacanza**
2. Il Sistema seleziona la Prenotazione
3. Il Cliente seleziona il Viaggiatore interessato
4. Il Cliente seleziona la tipologia di assistenza di cui ha bisogno
5. Il Sistema mostra un riepilogo delle tipologie di assistenza selezionate col prezzo totale
6. Il Cliente conferma la richiesta
7. Il Sistema registra la richiesta

I passi 3 e 4 si ripetono finché necessario.

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**:

1. Il Cliente aggiorna la pagina
2. Il Cliente effettua nuovamente l'accesso

3a. Il Cliente non conferma la richiesta:

1. Il caso d'uso termina

4a. La registrazione della richiesta non va a buon fine:

1. Il Sistema mostra il messaggio "errore nella registrazione della richiesta"

2. Si torna al passo 1

UC14: Gestione check-in

Attore Primario: Operatore

Scenario Principale:

1. Si attiva il caso d'uso **UC11: Visualizza prenotazioni dell'agenzia**
2. L'Operatore seleziona la prenotazione su cui effettuare un check-in
3. Il Sistema mostra un riepilogo dei dati dei Viaggiatori
4. L'Operatore conferma i dati
5. Il Sistema registra il completamento del check-in

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**:

1. L'Operatore aggiorna la pagina
2. L'Operatore inserisce nuovamente le sue informazioni, ritornando al passo 1

1a. Non ci sono prenotazioni su cui effettuare un check-in:

1. Il caso d'uso termina

2a. Mancano più di due giorni alla partenza

1. Il Sistema mostra un messaggio "non è possibile effettuare un check-in"
2. L'Operatore sceglie un'altra prenotazione, ripetendo il passo 2

3a. Il recupero dei dati dei Viaggiatori non va a buon fine:

1. Il Sistema mostra un messaggio "errore nel recupero dei viaggiatori"
2. Il passo 2 si ripete

5a. La registrazione del check-in non va a buon fine:

1. Il Sistema ripete l'operazione

UC15: Recensione

Attore Primario: Cliente

Scenario Principale:

1. Il sistema mostra l'elenco delle prenotazioni del cliente (UC10), evidenziando quelle completate per cui è possibile lasciare una recensione
2. Il cliente seleziona la prenotazione completata per cui desidera lasciare una recensione
3. Il sistema presenta il modulo di recensione suddiviso nelle seguenti sezioni:
    - Valutazione agenzia di viaggio:
        - Punteggio da 1 a 5 stelle
        - Campo testo libero per commenti
    - Valutazione compagnia di trasporto:
        - Punteggio da 1 a 5 stelle
        - Campo testo libero per commenti
    - Valutazione struttura/e di alloggio:
        - Punteggio da 1 a 5 stelle
        - Campo testo libero per commenti
4. Il cliente compila le sezioni del modulo, inserendo i punteggi e i commenti
5. Il cliente seleziona "Conferma"
6. Il sistema valida i dati inseriti verificando che i campi obbligatori siano compilati
7. Il sistema registra la recensione nel database collegandola alla prenotazione di riferimento
8. Il sistema mostra al cliente un messaggio di conferma

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

1. Il Cliente aggiorna la pagina
2. Il Cliente effettua nuovamente l'accesso

1a. Nessuna prenotazione completata disponibile

1. Il sistema mostra il messaggio "Non hai prenotazioni completate per cui lasciare una recensione"
2. Il caso d'uso termina

1b. Recensione già presente per la prenotazione

1. Il sistema offre la possibilità di visualizzare la recensione precedentemente inviata
2. Il caso d'uso termina

6a. Validazione fallita

1. Il sistema rileva che uno o più campi obbligatori non sono stati compilati
2. Il sistema visualizza un messaggio di errore
3. Il sistema rimanda al passo 4

7a. Errore nel salvataggio

1. Il sistema non riesce a salvare la recensione
2. Il sistema mostra il messaggio "Errore durante il salvataggio. Riprova più tardi"
3. Il sistema non salva la recensione
4. Il caso d'uso termina

UC18: Visualizzazione recensioni

Attore Primario: Operatore

Scenario Principale:

1. Il sistema recupera tutte le recensioni associate all'agenzia
2. Il sistema visualizza la dashboard delle recensioni con un riepilogo generale:
    - Punteggio medio complessivo dell'agenzia
    - Punteggio medio compagnie di trasporto
    - Punteggio medio strutture di alloggio
    - Numero totale di recensioni ricevute
3. Il sistema visualizza l'elenco completo delle recensioni visualizzando:
    - Data di invio della recensione
    - Nome cliente
    - Prenotazione di riferimento (codice e destinazione)
    - Punteggio agenzia
    - Punteggio compagnia di trasporto
    - Punteggio struttura di alloggio
    - Pulsante per visualizzare tutti i dettagli della recensione

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**:

3. L’Operatore aggiorna la pagina
4. L’Operatore inserisce nuovamente le sue informazioni, ritornando al passo 1

1a. Nessuna recensione presente

1. Il sistema mostra il messaggio "Nessuna recensione disponibile al momento"
2. Il caso d'uso termina

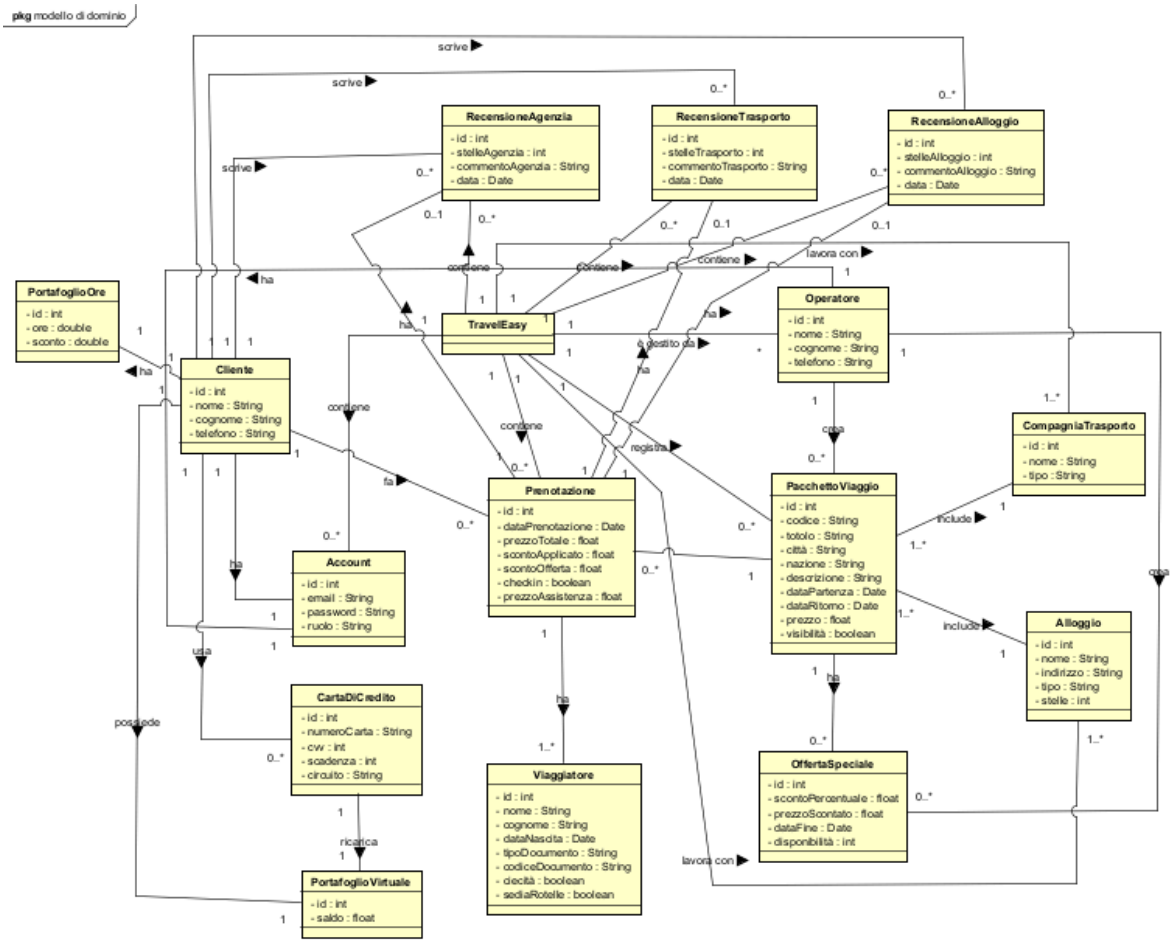
Analisi

Introduzione

Questo capitolo descrive l'analisi svolta nell'iterazione 3, mentre il capitolo successivo descrive l'attività di progettazione.

Modello di dominio

Il Modello di Dominio della terza iterazione rappresenta l'evoluzione finale della struttura informativa del sistema **TravelEasy**. Rispetto alle versioni precedenti, il modello introduce entità e attributi specifici per supportare le logiche di assistenza, check-in e recensioni.



Entità Principali e Responsabilità

L'entità ***Prenotazione*** è stata arricchita per supportare il ciclo di vita completo del viaggio:

- **Stato del Check-in**: L'attributo booleano ***checkin*** permette di tracciare l'avvenuta registrazione del viaggiatore (UC14).
- **Persistenza e Modifica**: La relazione tra *Cliente, TravelEasy e Prenotazione* garantisce che le operazioni di modifica (UC4) ed eliminazione (UC5) possano essere eseguite mantenendo la coerenza con i dati finanziari del cliente.

Per rispondere ai requisiti di inclusività e personalizzazione del viaggio (UC12 e UC14), l'entità ***Viaggiatore*** è stata estesa con attributi specifici:

- **Dettagli Medici e Logistici**: Sono stati inseriti i campi ***cecità: boolean*** e ***sediaRotelle: boolean***.
- **Impatto Economico**: La classe *Prenotazione* include ora l'attributo ***prezzoAssistenza: float***, che permette di contabilizzare separatamente i costi derivanti dai servizi speciali richiesti.

Il modello introduce un sottosistema di feedback (UC15 e UC18) che permette al cliente di valutare ogni aspetto dell'esperienza di viaggio:

- **Specializzazione delle Recensioni**: Sono state definite tre classi distinte per gestire i diversi domini di feedback:
    - ***RecensioneAgenzia***: Valuta l'operato di *TravelEasy*.
    - ***RecensioneTrasporto***: Valuta la compagnia di trasporto.
    - ***RecensioneAlloggio***: Valuta l'alloggio.
- **Attributi di Valutazione**: Ogni classe di recensione condivide una struttura basata su ***stelle*** (valutazione numerica), ***commento*** (testuale) e ***data***.
- **Relazioni**: Il ***Cliente*** "scrive" le recensioni (UC15), mentre ***all'Operatore*** è consentita la visualizzazione e l'analisi dei feedback ricevuti (UC18).

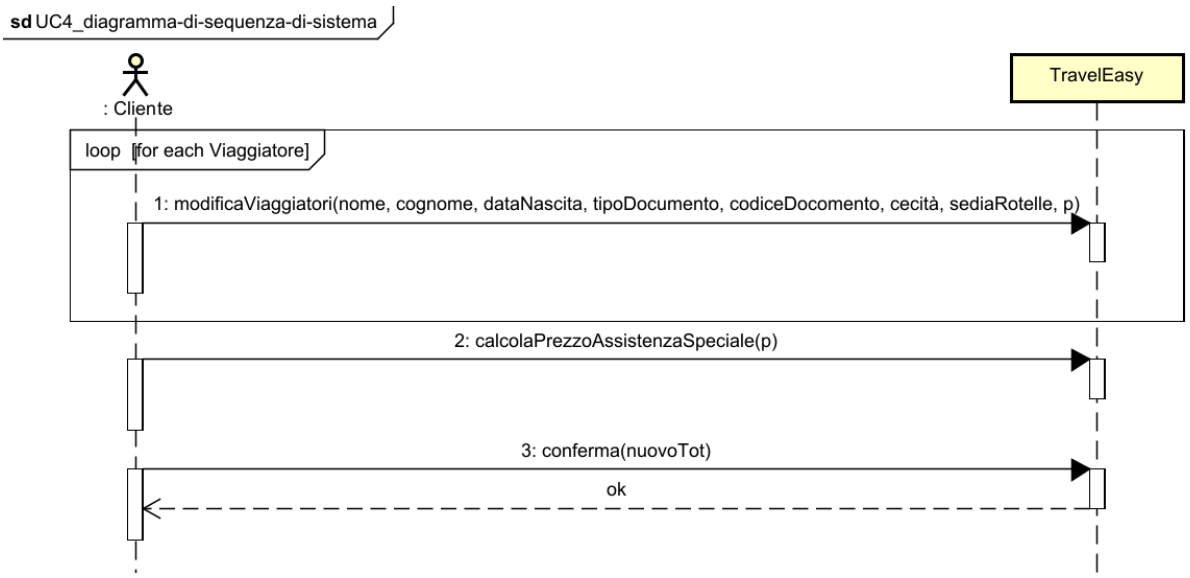
Il modello conferma la solidità della gestione utente e dei pagamenti già delineata nelle iterazioni precedenti:

- **Profilazione**: Ogni ***Cliente*** possiede un ***Account*** (per le credenziali) e può gestire più ***CartaDiCredito***.
- **Sistemi di Credito**: La separazione tra ***PortafoglioVirtuale*** (saldo monetario per ricariche e pagamenti) e ***PortafoglioOre*** (per accumulo punti fedeltà) permette una gestione flessibile delle promozioni.

Caso d'uso UC4 - Modifica Prenotazione

Diagramma di sequenza di sistema

Il diagramma di sistema per l'UC4 illustra l'interazione tra il **Cliente** e il sistema **TravelEasy** per l'aggiornamento dei dati relativi ai partecipanti di un viaggio già selezionato. L'operazione è particolarmente rilevante poiché gestisce non solo l'anagrafica, ma anche le necessità di assistenza speciale che possono influenzare il costo finale.



1. **Iterazione Dati Viaggiatori (*loop [for each Viaggiatore]*):** L'interazione inizia con un blocco ciclico che permette al Cliente di invocare il messaggio ***modificaViaggiatori(...)*** per ogni passeggero.
    - **Parametri inviat**i: Il Cliente fornisce i dati identificativi (*nome, cognome, dataNascita, tipoDocumento, codiceDocumento*) e specifica eventuali necessità di assistenza tramite i flag *cecità e sediaRotelle* per il pacchetto p.
    - **Obiettivo**: Garantire che il sistema disponga di informazioni accurate per le prenotazioni dei trasporti e degli alloggi associati.
2. **Ricalcolo Costi Assistenza (*calcolaPrezzoAssistenzaSpeciale(p)*):** Una volta terminato l'aggiornamento dei dati per tutti i viaggiatori, il Cliente richiede il calcolo di eventuali costi aggiuntivi. Il sistema valuta se le opzioni di assistenza speciale selezionate comportino una variazione del preventivo originale per il pacchetto p.
3. **Conferma Finale (*conferma(nuovoTot)*):** L'ultima fase prevede l'invio del messaggio di conferma da parte del Cliente, che accetta il *nuovoTot* calcolato dal sistema. Il sistema risponde con un messaggio di *ok*, finalizzando la modifica e aggiornando lo stato della prenotazione nel database.

Contratti delle operazioni

modificaViaggiatori

L'operazione di sistema ***modificaViaggiatori*** consente al **Cliente** di aggiornare o definire i dettagli dei partecipanti associati a una prenotazione all'interno del sistema **TravelEasy**.

Operazione	modificaViaggiatori(nome: String, cognome: String, dataNascita:Date, tipoDocumento: String, codiceDocumentoi: String, cecità: boolean, sediaRotelle: boolean, p:Prenotazione)
Riferimenti	UC4: Modifica prenotazione
Precondizioni	- Il Cliente ha effettuato l'accesso -è stata effettuata almeno una prenotazione

Postcondizioni	- Gli attributi del Viaggiatore associato alla prenotazione p sono stati aggiornati con i nuovi valori forniti

calcolaPrezzoAssistenzaSpeciale

L'operazione di sistema ***calcolaPrezzoAssistenzaSpeciale*** consente al **Cliente** di richiedere il ricalcolo automatico del costo complessivo della prenotazione all'interno del sistema **TravelEasy**.

Operazione	calcolaPrezzoAssistenzaSpeciale(p: Prenotazione)
Riferimenti	UC4: Modifica prenotazione
Precondizioni	- Almeno un viaggiatore associato alla prenotazione p ha uno o più flag di assistenza speciale attivi
Postcondizioni	- Il sistema ha calcolato il supplemento per assistenza speciale

conferma

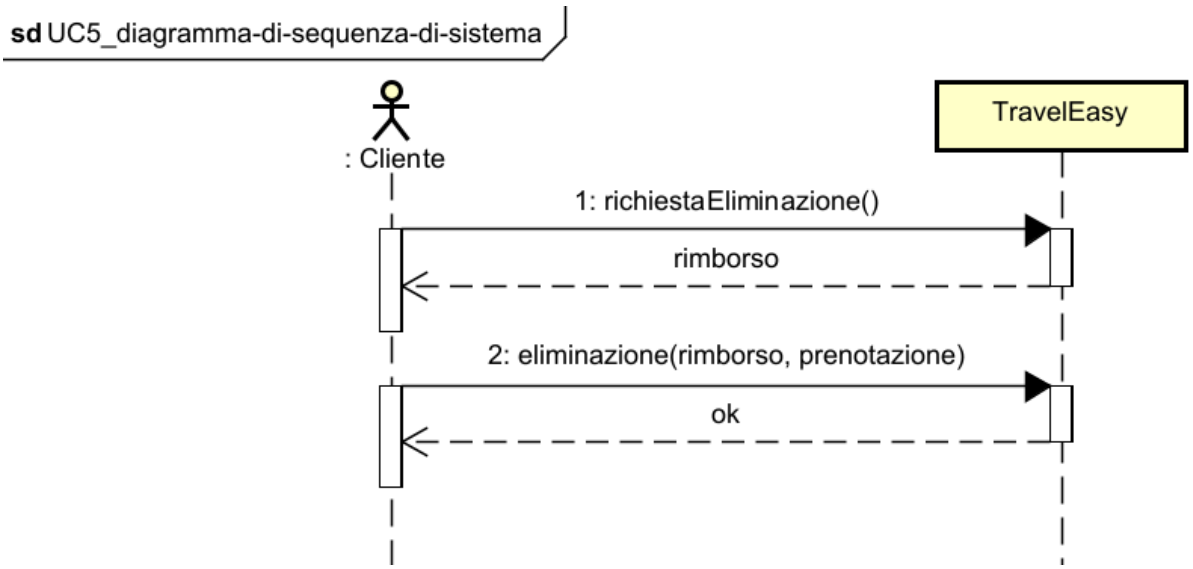
L'operazione di sistema ***conferma*** consente al **Cliente** di finalizzare la procedura di aggiornamento dei dati relativi ai viaggiatori all'interno del sistema **TravelEasy**.

Operazione	conferma(nuovoTot: float)
Riferimenti	UC4: Modifica prenotazione
Precondizioni	- Il nuovo totale nuovoTot è stato calcolato con successo
Postcondizioni	- Il totale della prenotazione p è stato aggiornato al valore nuovoTot. - Le modifiche ai dati dei viaggiatori sono state salvate nel sistema.

Caso d’uso UC5 - Elimina Prenotazione

Diagramma di sequenza di sistema

Il diagramma di sistema per l'UC5 descrive l'interazione tra il **Cliente** e il sistema **TravelEasy** finalizzata alla cancellazione di una prenotazione esistente e alla gestione della relativo rettifica economica.



1. **Richiesta di Calcolo Rimborso *(richiestaEliminazione()*)**: L'interazione ha inizio con l'invio del messaggio ***richiestaEliminazione()*** da parte del Cliente.
    - **Logica di Sistema**: Il sistema riceve la richiesta e calcola l'importo del ***rimborso*** spettante all'utente. Questo calcolo è influenzato dalle politiche di cancellazione dell'agenzia.
    - **Risposta**: Il sistema restituisce al Cliente l'entità del rimborso calcolato, permettendogli di valutare la convenienza dell'operazione prima di procedere in modo definitivo.
2. **Conferma ed Eliminazione *(eliminazione(rimborso, prenotazione)*)**: Una volta preso atto della quota rimborsabile, il Cliente invoca il messaggio finale di ***eliminazione***.
    - **Parametri inviati**: L'attore passa al sistema i riferimenti necessari, ovvero l'identificativo della ***prenotazione*** da cancellare e il valore del ***rimborso***.
    - **Esito**: Il sistema provvede alla rimozione della prenotazione e al contestuale riaccredito dei fondi. Il processo si conclude con un messaggio di ***ok***, che conferma il successo della transazione e il ripristino della disponibilità dei posti nel catalogo.

Contratti delle operazioni

richiestaEliminazione

L'operazione di sistema ***richiestaEliminazione*** consente al **Cliente** di avviare la procedura di cancellazione di una prenotazione esistente all'interno del sistema **TravelEasy**.

Operazione	richiestaEliminazione()
Riferimenti	UC5: Elimina prenotazione
Pre-condizioni	- Il Cliente è autenticato - Esiste almeno una prenotazione effettuata
Post-condizioni	- Il sistema ha calcolato l'importo di rimborso spettante al cliente in base alla policy di cancellazione

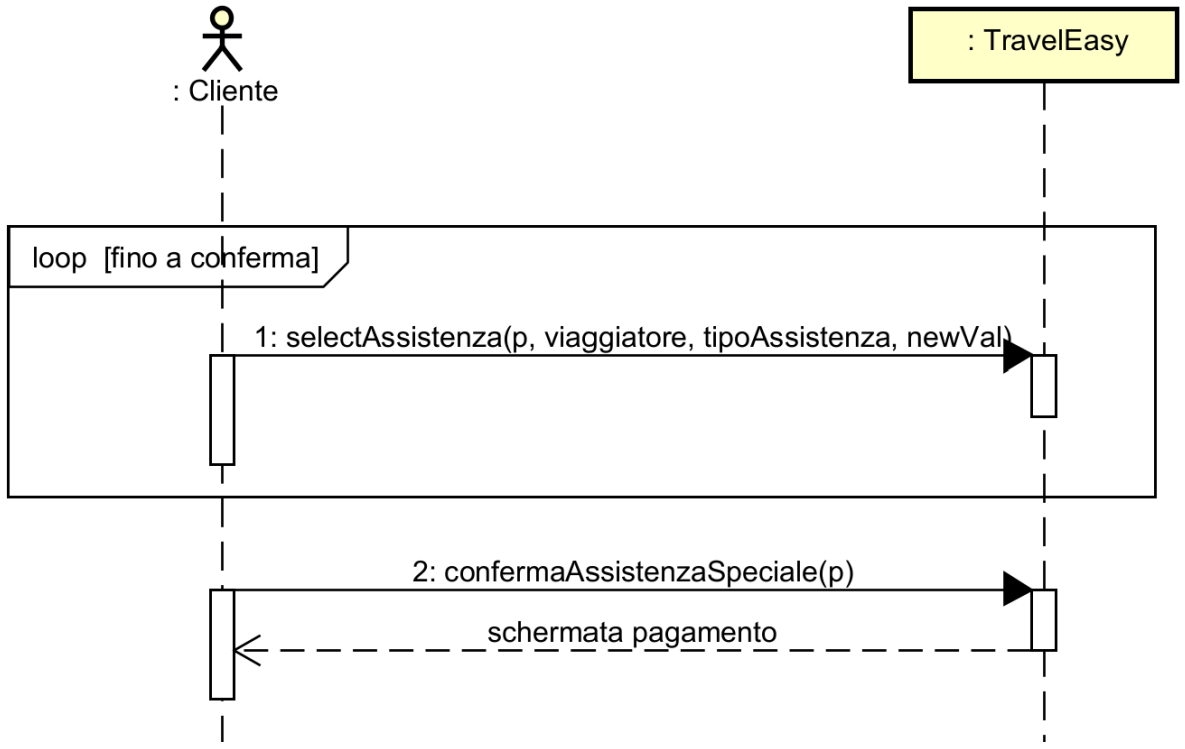
eliminazione

L'operazione di sistema ***eliminazione*** consente al **Cliente** di completare definitivamente la cancellazione della prenotazione selezionata all'interno del sistema **TravelEasy**.

Operazione	eliminazione(rimborso: float, prenotazione: Prenotazione)
Riferimenti	UC5: Elimina prenotazione
Pre-condizioni	- Il cliente ha preso visione del rimborso
Post-condizioni	- L'istanza di Prenotazione identificata da prenotazione è stata eliminata dal sistema - Il valore rimborso è stato accreditato al cliente

Caso d'uso UC12 - Gestione Assistenze Speciali

Diagramma di sequenza di sistema



Contratti delle operazioni

selectAssistenza

Operazione	selectAssistenza(p: Prenotazione, viaggiatore: Viaggiatore, tipoAssistenza: String, newVal: boolean)
Riferimenti	Caso d'uso: *Gestione assistenze speciali*

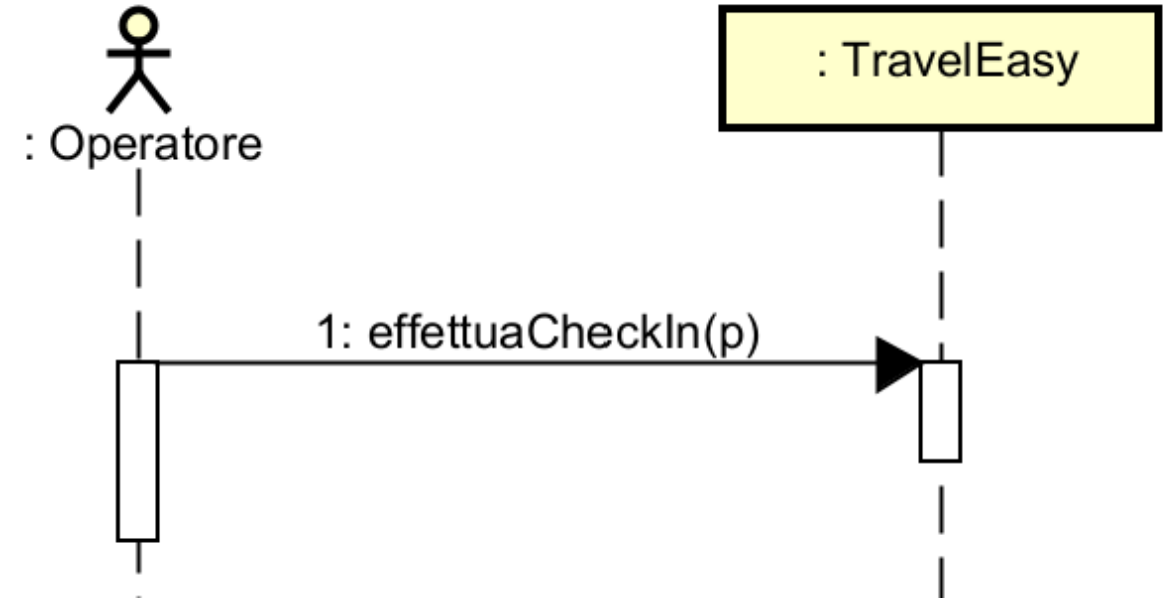
Precondizioni	- il Cliente è autenticato - è in corso la prenotazione di un pacchetto
Postcondizioni	- è stata identificata l'istanza *v* di Viaggiatore - gli attributi di *v* sono stati aggiornati

confermaAssistenzaSpeciale

Operazione	confermaAssistenzaSpeciale(p: Prenotazione)
Riferimenti	Caso d'uso: *Gestione assistenze speciali*
Precondizioni	- il Cliente è autenticato - è in corso la prenotazione di un pacchetto
Postcondizioni	- gli attributi dell'istanza *p* di Prenotazione sono stati aggiornati

Caso d'uso UC14 - Gestione check-in

Diagramma di sequenza di sistema



Contratti delle operazioni

effettuaCheckIn

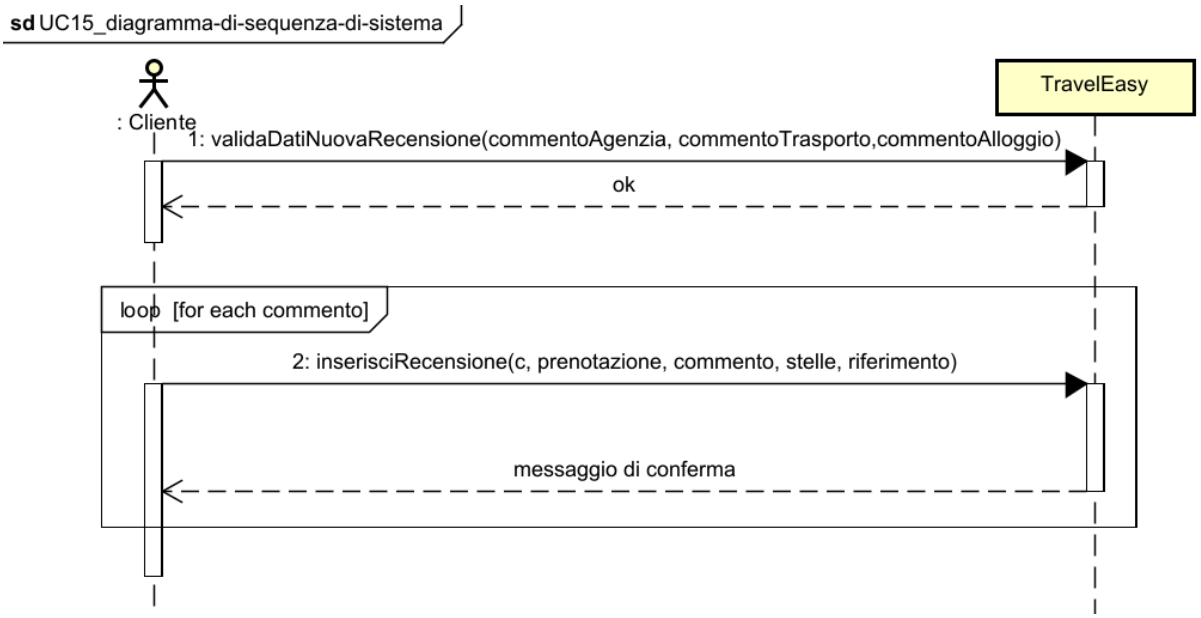
Operazione	effettuaCheckIn(p: Prenotazione)
Riferimenti	Caso d'uso: *Gestione check-in*

Precondizioni	-	l'Operatore è autenticato - l'Operatore ha selezionato una Prenotazione *p* - mancano meno di due giorni alla partenza
Postcondizioni	-	gli attributi di *p* sono stati aggiornati

Caso d'uso UC15 - Recensione

Diagramma di sequenza di sistema

Il diagramma di sistema per l'UC15 illustra il protocollo di interazione tra il **Cliente** e il sistema **TravelEasy** per il rilascio di feedback dettagliati a seguito di un'esperienza di viaggio. Il processo è progettato per garantire la qualità dei contenuti e la granularità del giudizio.



1. **Validazione Preliminare (*validaDatiNuovaRecensione(...)*)**: L'interazione ha inizio con l'invio simultaneo dei commenti testuali relativi alle tre componenti principali del pacchetto: l'agenzia (*commentoAgenzia*), il trasporto (*commentoTrasporto*) e l'alloggio (*commentoAlloggio*).
    - **Logica di Sistema**: Il sistema esegue un controllo formale sulla validità dei dati.
    - **Esito**: Una volta superata la validazione, il sistema risponde con un messaggio di ***ok***, abilitando l'utente alla fase successiva di inserimento dei punteggi.
2. **Iterazione Feedback Dettagliato *(loop [for each commento]*)**: Per garantire una valutazione precisa di ogni fornitore, il sistema adotta un blocco ciclico che permette al Cliente di invocare il messaggio ***inserisciRecensione(...)*** per ogni singola entità valutata.
    - **Parametri inviati**: Il Cliente specifica il riferimento al profilo (**c**), alla **prenotazione** effettuata, al testo del **commento**, al punteggio espresso in **stelle** e al fornitore di **riferimento**.
    - **Risultato**: il sistema restituisce un **messaggio di conferma**, assicurando che ogni componente del feedback sia stata correttamente acquisita.

Contratti delle operazioni

validaDatiNuovaRecensione

L'operazione di sistema ***validaDatiNuovaRecensione*** consente al **Cliente** di sottoporre a una verifica preliminare i contenuti testuali della propria valutazione all'interno del sistema **TravelEasy**.

Operazione	validaDatiNuovaRecensione(commentoAgenzia: String, commentoTrasporto: String, commentoAlloggio: String)
Riferimenti	UC15: Recensione
Pre-condizioni	- Il cliente deve essere autenticato nel sistema - Il viaggio deve essere effettivamente concluso (data di rientro superata)
Post-condizioni	- I dati inseriti sono stati validati

inserisciRecensione

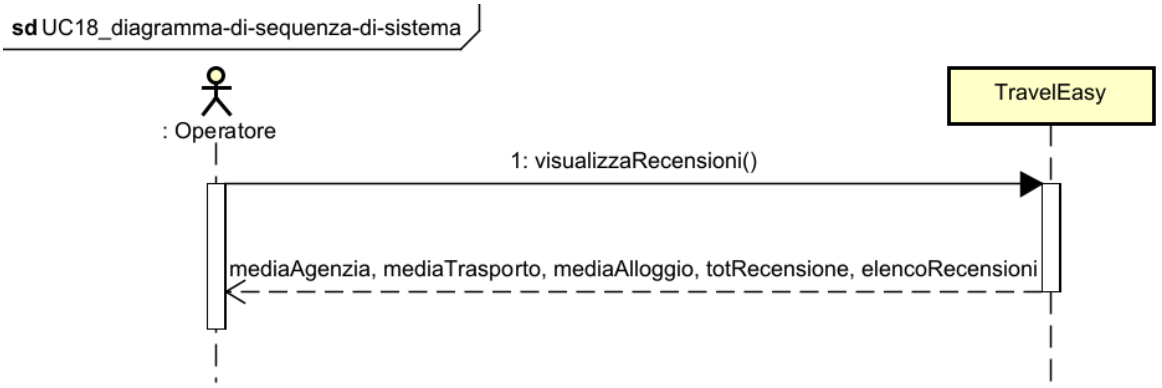
L'operazione di sistema ***inserisciRecensione*** consente al **Cliente** di registrare nel sistema **TravelEasy** il dettaglio qualitativo di ogni singola componente del viaggio acquistato.

Operazione	inserisciRecensione(c: Cliente, prenotazione: Prenotazione, commento: String, stelle: int, riferimento: String)
Riferimenti	UC15: Recensione
Pre-condizioni	- I dati sono stati validati
Post-condizioni	- La recensione viene registrata e collegata alla prenotazione di riferimento

Caso d’uso UC18 - Visualizzazione Recensioni

Diagramma di sequenza di sistema

Il diagramma di sistema per l'UC18 descrive l'interazione tra l'**Operatore** e il sistema **TravelEasy** finalizzata al monitoraggio della qualità dei servizi attraverso l'analisi dei feedback lasciati dai clienti.



1. **Richiesta Dati *(visualizzaRecensioni()*)**: L'interazione è attivata dall'Operatore tramite il messaggio ***visualizzaRecensioni()***. Questa operazione non richiede parametri di input espliciti, poiché il sistema è progettato per interrogare globalmente il proprio database delle recensioni.
2. **Output Analitico del Sistema**: In risposta alla richiesta, il sistema **TravelEasy** elabora e restituisce un set di dati aggregati e analitici di fondamentale importanza per la gestione aziendale:
    - **Indicatori di Qualità**: Vengono fornite le medie dei punteggi (stelle) per le tre categorie principali: ***mediaAgenzia***, ***mediaTrasporto*** e ***mediaAlloggio***.
    - **Volume dei Feedback**: Viene restituito il valore ***totRecensione***, che indica il numero complessivo di valutazioni ricevute, utile per pesare statisticamente le medie calcolate.
    - **Dettaglio Testuale**: Il sistema fornisce ***l'elencoRecensioni*** completo, permettendo all'Operatore di consultare i singoli commenti validati e inseriti dai clienti nelle fasi precedenti (UC15).

Contratti delle operazioni

visualizzaRecensioni

L'operazione di sistema ***visualizzaRecensioni*** consente all'**Operatore** di accedere al modulo di monitoraggio e analisi dei feedback dei clienti all'interno del sistema **TravelEasy**.

Operazione	visualizzaRecensioni()
Riferimenti	UC18: Visualizzazione recensioni
Precondizioni	- L'Operatore è autenticato nel sistema
Postcondizioni	- L'operatore visualizza le recensioni dei clienti - È stato calcolato: - La media degli attributi stelleAgenzia di tutte le recensioni - La media degli attributi stelleTrasporto di tutte le recensioni - La media degli attributi stelleAlloggio di tutte le recensioni. - Il conteggio totale delle recensioni.

Progettazione

Caso d’uso UC4 - Modifica Prenotazione, diagrammi di interazione

modificaViaggiatori

L'operazione ***modificaViaggiatori*** gestisce l'aggiornamento dei profili dei passeggeri associati a una specifica prenotazione.

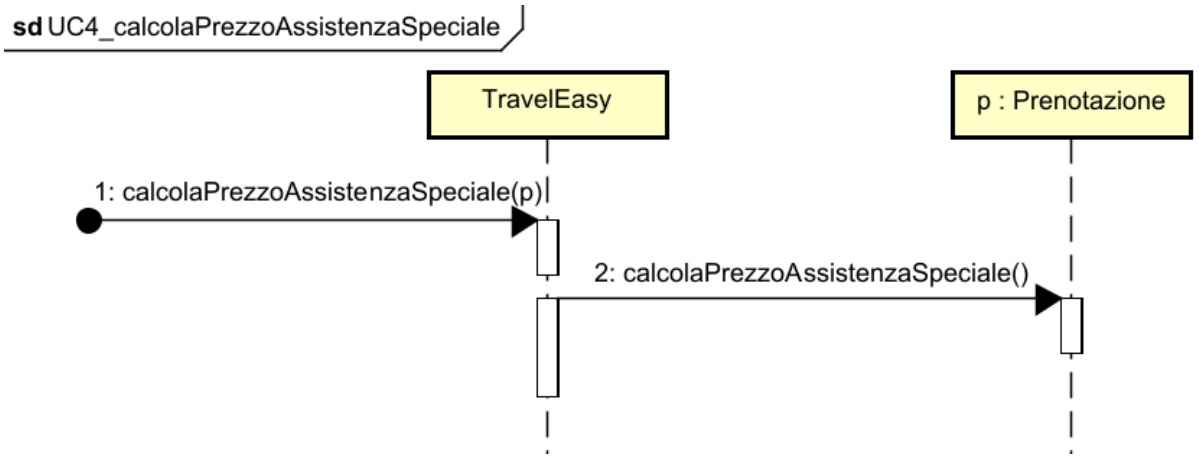
Flusso delle Operazioni e Logica Interna

1. **Ricezione e Validazione**: Il controller ***TravelEasy*** riceve il messaggio contenente l'anagrafica completa del viaggiatore, gli estremi del documento e i flag di assistenza speciale (*cecità*, *sediaRotelle*) riferiti alla prenotazione **p**. Immediatamente, il controller esegue un'operazione interna di ***ValidaDati()*** per assicurarsi che i campi siano formalmente corretti.
2. **Delega alla Prenotazione (Information Expert)**: Una volta validati i dati, il controller delega l'aggiornamento all'istanza ***p : Prenotazione*** tramite il messaggio ***replaceViaggiatori(...)***. Questa scelta di design rispetta il principio **Expert**, poiché solo l'oggetto *Prenotazione* possiede il riferimento diretto all'elenco dei partecipanti da modificare.
3. **Aggiornamento dell'Elenco**: L'oggetto *Prenotazione* interagisce con la collezione ***elencoViaggiatori*** (implementata come una *List<Viaggiatore>*) invocando il metodo ***set(...)***. Questo passaggio sostituisce o aggiorna l'oggetto *Viaggiatore* corrispondente con i nuovi parametri forniti, inclusi quelli necessari per il calcolo dell'assistenza speciale.

Elementi di Design

- **Coerenza dei Dati**: L'operazione assicura che ogni modifica anagrafica sia istantaneamente riflessa nella lista dei viaggiatori della prenotazione, garantendo che i messaggi successivi (come il calcolo del prezzo) operino su dati aggiornati.
- **Separazione delle Responsabilità**: Il controller gestisce la logica di interfaccia e validazione iniziale, mentre la persistenza in memoria dei dati del viaggiatore è affidata alla gerarchia delle classi di dominio.

calcolaPrezzoAssistenzaSpeciale



L'operazione ***calcolaPrezzoAssistenzaSpeciale(p)*** ha il compito di determinare l'impatto economico delle necessità specifiche dichiarate per i viaggiatori sul costo totale della prenotazione.

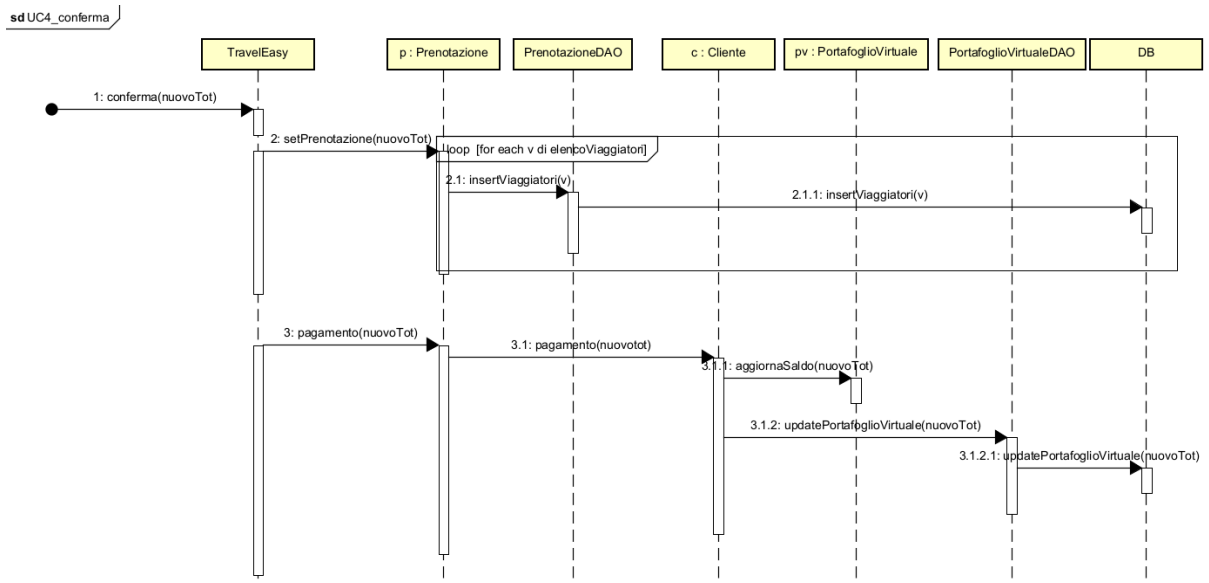
Flusso delle Operazioni e Logica Interna

1. **Innesco del Calcolo**: Il controller ***TravelEasy*** riceve la richiesta di ricalcolo per la prenotazione identificato dal riferimento **p**.
2. **Delega all'Expert (Information Expert)**: Seguendo i principi del design orientato agli oggetti, il controller non esegue direttamente il calcolo algoritmico. Esso delega invece la responsabilità all'istanza ***p : Prenotazione*** invocando il metodo ***calcolaPrezzoAssistenzaSpeciale().***
3. **Logica di Business**: L'oggetto *Prenotazione* è l'entità esperta in quanto aggrega ***l'elencoViaggiatori*** aggiornato nella fase precedente. Esso analizza i flag di ogni viaggiatore (come *cecità* o *sediaRotelle*) e somma gli eventuali costi accessori al prezzo base del pacchetto.

Elementi di Design

- **Basso Accoppiamento**: Il controller rimane agnostico rispetto alle tariffe specifiche per l'assistenza, limitandosi a coordinare la comunicazione tra l'attore e il modello di dominio.
- **Coesione**: La logica di prezzo è incapsulata all'interno delle classi di dominio che gestiscono i dati della prenotazione, garantendo che future modifiche alle politiche tariffarie dell'agenzia non influenzino la struttura del controller.

conferma



L'operazione ***conferma(nuovoTot)*** rappresenta l'atto finale del caso d'uso, in cui il sistema rende persistenti le modifiche ai viaggiatori e procede all'aggiornamento contabile della prenotazione.

Flusso delle Operazioni e Logica Interna

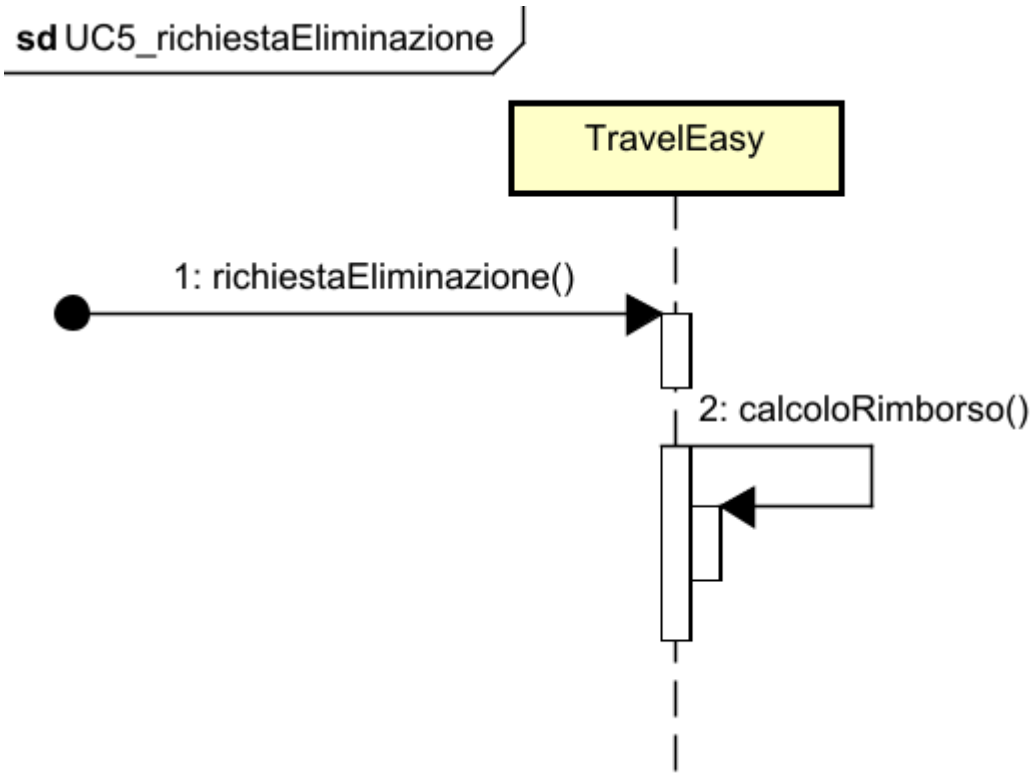
1. **Chiusura della Transazione**: Il controller ***TravelEasy*** riceve il messaggio di conferma dall'attore, comprensivo dell'importo finale ricalcolato (*nuovoTot*).
2. **Aggiornamento dello Stato della Prenotazione**: Il controller invoca il metodo ***setPrenotazione(nuovoTot)*** sull'istanza ***p : Prenotazione***. Questo passaggio è fondamentale per allineare formalmente il costo del pacchetto nel database alle modifiche logistiche effettuate (es. costi di assistenza speciale). Serve inoltre ad aggiungere al Database tutti i viaggiatori di questa specifica prenotazione.
3. **Gestione del Pagamento (Flusso Delegato)**:
    - Il controller avvia la procedura di addebito chiamando ***pagamento(nuovoTot)*** sulla prenotazione **p**.
    - L'oggetto *Prenotazione*, agendo come coordinatore del pagamento per i propri servizi, delega a sua volta l'operazione all'oggetto ***c : Cliente***.
    - Infine, l'oggetto *Cliente* richiede al proprio ***pv : PortafoglioVirtuale*** di eseguire l'aggiornamento del saldo tramite il metodo ***aggiornaSaldo(nuovoTot)*** e aggiornare il Database.

Elementi di Design

- **Delega a Catena**: Il diagramma mostra una corretta applicazione della delega delle responsabilità. Il controller non manipola direttamente il saldo del cliente; l'operazione passa attraverso gli oggetti di dominio (*Prenotazione -> Cliente -> PortafoglioVirtuale*) fino a raggiungere l'**Expert** che possiede l'attributo *saldo*.
- **Integrità Finanziaria**: L'associazione biunivoca tra la conferma dei dati e l'aggiornamento del saldo garantisce che non vi siano discrepanze tra i passeggeri registrati e l'importo effettivamente pagato dal cliente.
- **Persistenza dei dati:** L’utilizzo delle classi DAO è volto a delegare le operazioni di persistenza sul Database e delle classi apposite per aumentare la coerenza e diminuire l’accoppiamento.

Caso d’uso UC5 - Elimina Prenotazione, diagrammi di interazione

richiestaEliminazione



L'operazione ***richiestaEliminazione()*** costituisce la fase preliminare del processo di cancellazione, finalizzata a informare l'utente sugli effetti economici dell'annullamento della prenotazione prima della conferma definitiva.

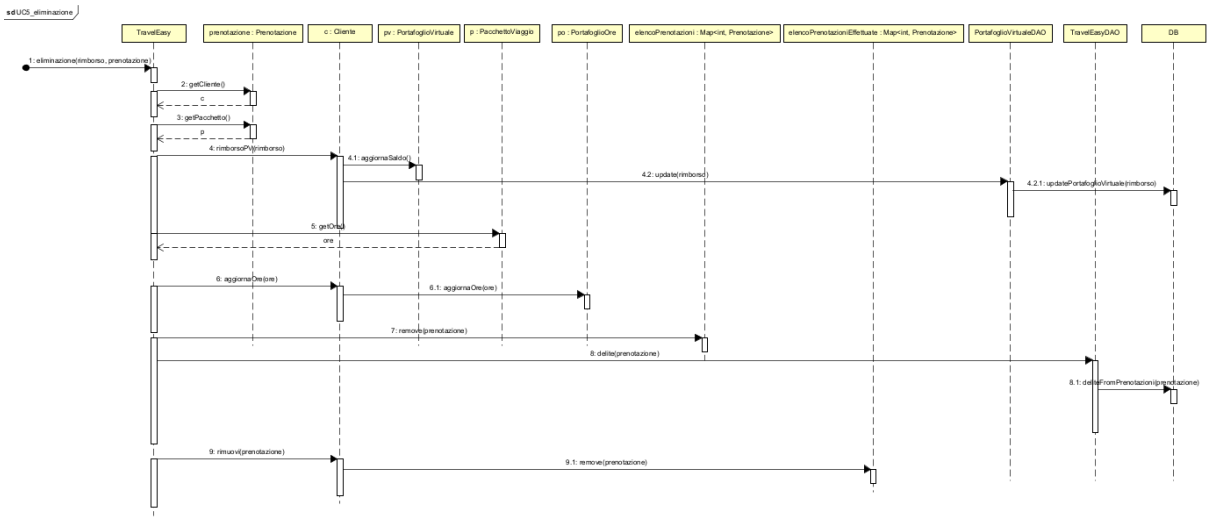
Flusso delle Operazioni e Logica Interna

1. **Ricezione del Messaggio**: Il controller di sistema ***TravelEasy*** intercetta la richiesta di eliminazione proveniente dall'interfaccia del Cliente.
2. **Logica di Calcolo Interna**: Il controller esegue un messaggio riflessivo chiamato ***calcoloRimborso()***.
3. **Determinazione del Valore**: Durante questa fase, il sistema agisce come **Information Expert**, analizzando i dati della prenotazione in oggetto e confrontandoli con le politiche di cancellazione aziendali. Il calcolo deve tenere conto di variabili quali la data corrente rispetto alla data di partenza del pacchetto per quantificare con precisione la quota rimborsabile.

Elementi di Design

- **Basso Accoppiamento**: In questa fase, la logica di calcolo del rimborso è centralizzata nel controller *TravelEasy*. Sebbene questo aumenti leggermente la complessità del controller, garantisce che le regole di business per i rimborsi siano gestite in un unico punto di controllo.
- **Trasparenza verso l’Utente**: La scelta di separare il calcolo dall'eliminazione effettiva rispetta il requisito di trasparenza verso l'utente, fornendo un output (il valore del rimborso) che funge da condizione necessaria per l'attivazione della fase successiva di eliminazione definitiva.

eliminazione



L'operazione ***eliminazione*** rappresenta la fase operativa in cui il sistema **TravelEasy** processa l'annullamento della prenotazione, gestendo contestualmente il riaccredito dei fondi e l'aggiornamento delle disponibilità.

Flusso delle Operazioni e Logica Interna

1. **Recupero Informazioni di Dominio**: Il controller riceve il comando di eliminazione e interroga l'oggetto *prenotazione* per ottenere i riferimenti al ***Cliente (c)*** e al ***PacchettoViaggio (p)*** associati. Questo passaggio è essenziale per identificare correttamente su quali entità applicare i cambiamenti.
2. **Rettifica Finanziaria**: Il sistema attiva la procedura di rimborso delegando al Cliente l'aggiornamento del proprio saldo. Nello specifico, il controller invoca aggiornando anche il Database ***rimborsoPV(rimborso)*** sul Cliente, il quale istruisce il proprio ***PortafoglioVirtuale (pv)*** per eseguire l'operazione di ***aggiornaSaldo()*** con la quota riaccreditata.
3. **Gestione delle Ore**: Oltre al rimborso monetario, il sistema gestisce il ripristino delle risorse. Il controller recupera il valore delle "ore" dal pacchetto **p** e aggiorna il ***PortafoglioOre (po)*** del cliente tramite il metodo ***aggiornaOre(ore)***.
4. **Pulizia delle Collezioni di Sistema**: L'ultima fase prevede la rimozione fisica della prenotazione dalle mappe globali gestite dal controller:
    - Viene invocato ***remove(prenotazione)*** sulla mappa ***elencoPrenotazioni***.
    - Viene invocato ***rimuovi(prenotazione)*** sul Cliente, che a sua volta aggiorna la propria collezione interna ***elencoPrenotazioniEffettuate*** tramite un'operazione di ***remove***.
    - Viene inoltre eliminata la prenotazione dal Database.

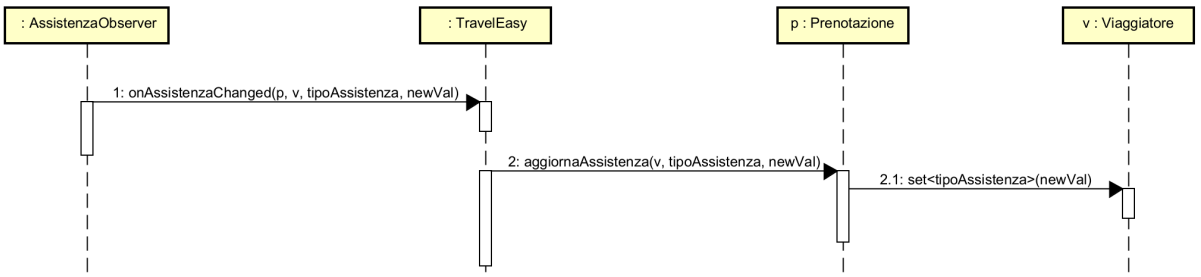
Elementi di Design

- **Integrità Transazionale**: Il diagramma mostra come l'eliminazione non sia una semplice cancellazione di un record, ma un processo coordinato che garantisce la coerenza tra il saldo del cliente, la gestione delle ore e le liste storiche delle prenotazioni.
- **Applicazione dei Principi GRASP**: Il design sfrutta il principio **Information Expert**, delegando al Cliente e al suo Portafoglio la logica di aggiornamento dei valori numerici, mantenendo il controller come puro coordinatore del flusso.

- **Persistenza dei dati**: L'utilizzo delle classi DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.

Caso d'uso UC12 - Gestione Assistenze Speciali, diagrammi di interazione

selectAssistenza



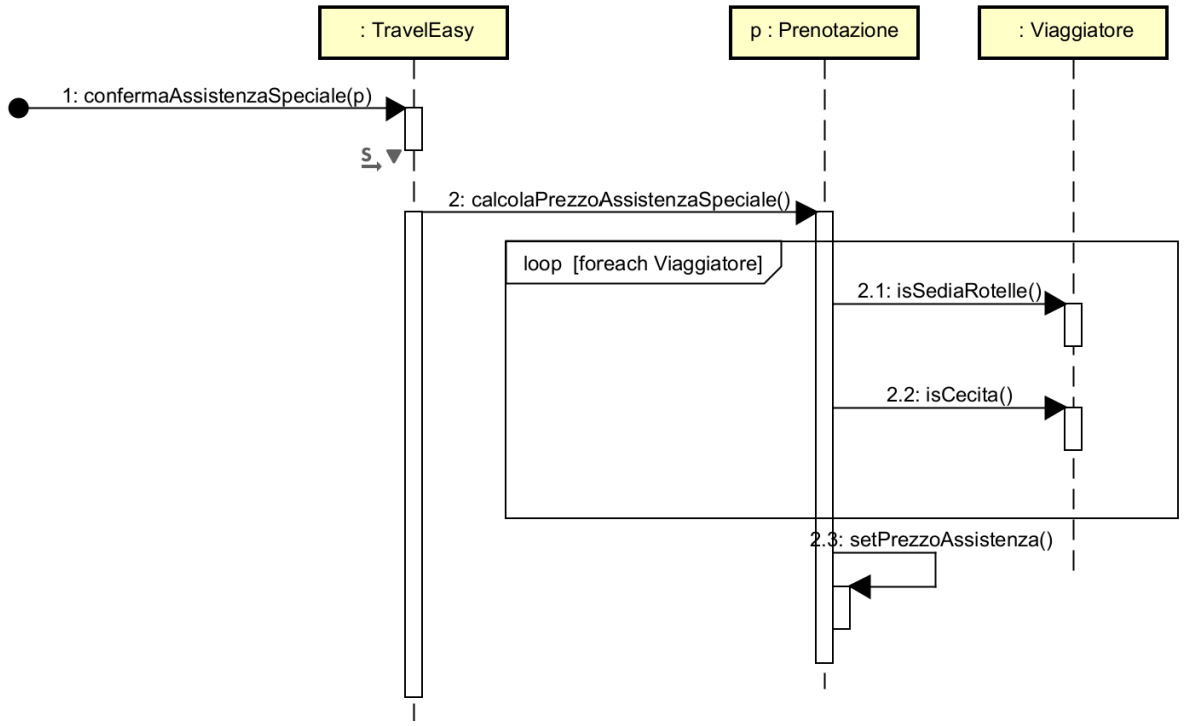
Flusso delle Operazioni e Logica Interna

1. **Notifica di cambiamento**: L'osservatore ***AssistenzaObserver*** notifica, con un messaggio *onAssistenzaChanged(...),* il controller ***TravelEasy*** di un cambiamento nel tipo di assistenza selezionata dal Cliente tramite l'interfaccia grafica.
2. **Aggiornamento del Viaggiatore**: Una volta notificato, il controller provvede a mandare un messaggio *aggiornaAssistenza(...)* alla ***Prenotazione*** che sta venendo definita. La ***Prenotazione*** procede dunque a settare l'attributo del ***Viaggiatore*** selezionato, aggiornandolo al nuovo valore di quella tipologia di assistenza.

Elementi di Design

- **Pattern Observer**: L'utilizzo del Pattern Observer rende l'aggiornamento dei ***Viaggiatori*** istantaneo, notificando il Sistema appena il Cliente richiede un cambiamento.
- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di aggiornamento e archiviazione.

confermaAssistenzaSpeciale



Flusso delle Operazioni e Logica Interna

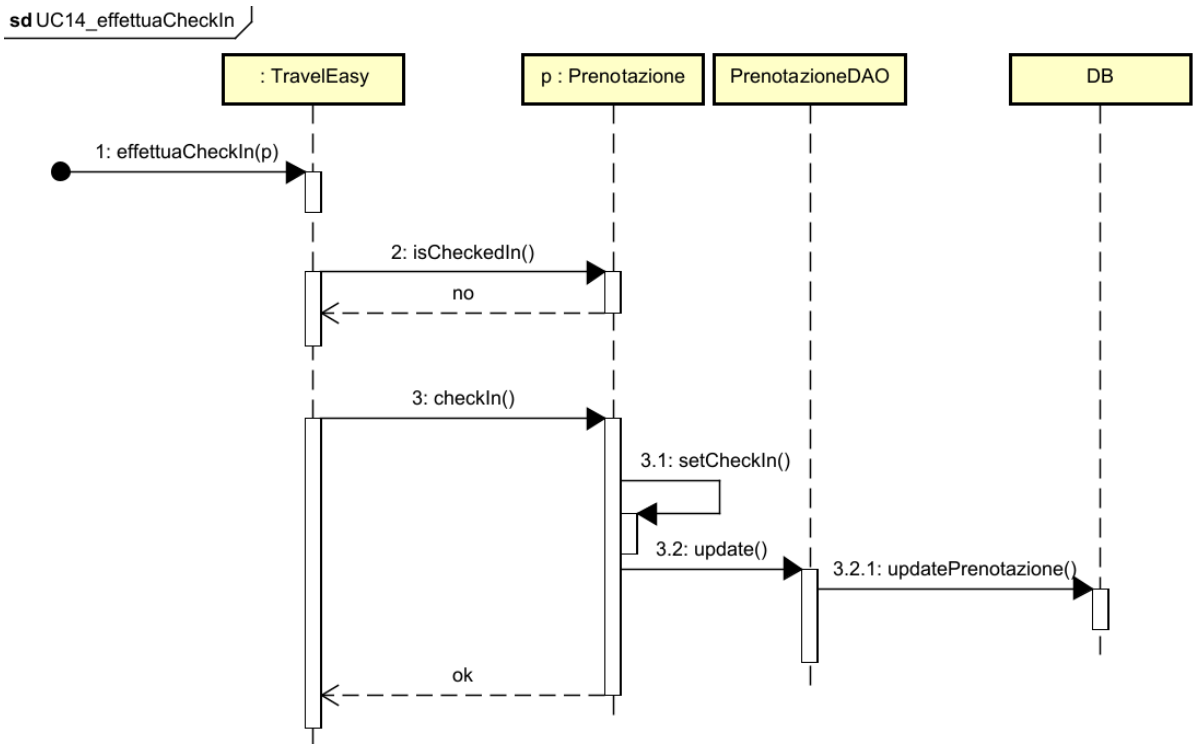
1. **Conferma**: Una volta che la selezione dei tipi di assistenza per ogni Viaggiatore è stata completata il controller ***TravelEasy*** viene notificato tramite un messaggio *condermaAssistenzaSpeciale(p)*.
2. **Aggiornamento della Prenotazione**: Una volta notificato, il controller provvede a mandare un messaggio *calcolaPrezzoAssistenzaSpeciale()* alla ***Prenotazione*** che sta venendo definita. La ***Prenotazione*** procede dunque a scorrere il proprio ***elencoViaggiatori*** controllando per ogni oggetto ***Viaggiatore*** se i suoi attributi relativi alle assistenze sono impostati su *true*, aggiungendo nel caso il prezzo dell'assistenza al totale. Una volta concluso di scorrere la lista, la ***Prenotazione*** procede ad aggiornare il proprio attributo relativo al prezzo totale delle assistenze richieste.

Elementi di Design

- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di aggiornamento e archiviazione.
- **Gestione dei Dati**: L'uso di una ***List<Viaggiatori>*** per *elencoViaggiatori* suggerisce una scelta di design volta a memorizzare tutti i ***Viaggiatori*** associati ad una determinata ***Prenotazione***.

Caso d’uso UC14 - Gestione Check-in, diagrammi di interazione

effettuaCheckIn



Flusso delle Operazioni e Logica Interna

1. **Controllo**: Il controller ***TravelEasy*** viene notificato tramite un messaggio *effettuaCheckIn(p)* dopo che l’operatore ha selezionato la ***Prenotazione***. Provvede dunque a verificare che il check-in non sia già stato precedentemente effettuato sull’istanza selezionata.
2. **Esecuzione del check-in**: Il controller procede a mandare un messaggio *checkIn()* alla ***Prenotazione***, la quale controlla che manchino non più di due giorni alla partenza, seguendo le regole di dominio, per poi aggiornare il database segnando che il check-in è stato effettuato.

Elementi di Design

- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di aggiornamento e archiviazione.
- **Persistenza dei dati:** L’utilizzo delle classi DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l’accoppiamento.

Caso d’uso UC15 - Recensione, diagramma di interazione

validaDatiNuovaRecensione

L'operazione ***validaDatiNuovaRecensione*** costituisce la fase di controllo preventivo necessaria a garantire l'integrità dei feedback inseriti dai clienti nel sistema **TravelEasy**.

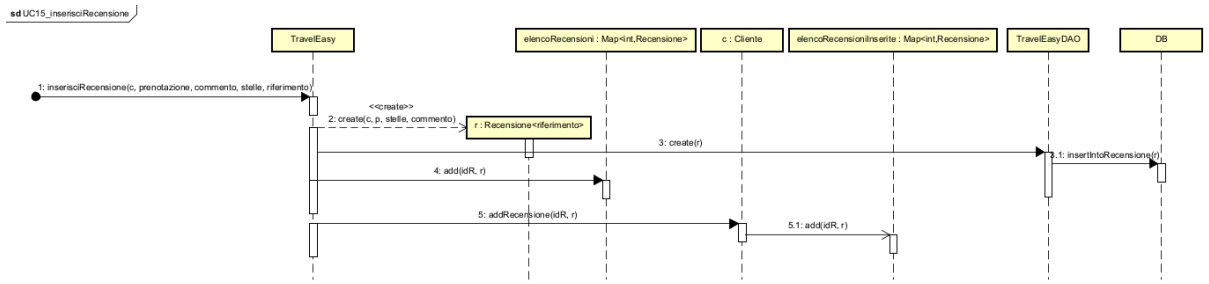
Flusso delle Operazioni e Logica Interna

1. **Ricezione dei Parametri**: Il controller ***TravelEasy*** riceve i tre blocchi testuali distinti che compongono la valutazione complessiva dell'esperienza: *commentoAgenzia, commentoTrasporto e commentoAlloggio.*
2. **Logica di Validazione Centralizzata**: Il controller esegue un'auto-delega invocando il metodo interno ***valida()***. In questa fase, il sistema verifica che i testi non siano vuoti.
3. **Esito della Procedura**:
    - Se i dati sono conformi, il controller abilita il flusso verso la fase successiva di inserimento dei punteggi numerici.
    - Questa scelta di design permette di intercettare eventuali errori di inserimento prima che il sistema provi a istanziare oggetti di dominio, riducendo il rischio di dati inconsistenti nel database.

Elementi di Design

- **Responsabilità del Controller**: In linea con il pattern **Controller**, *TravelEasy* assume il compito di orchestrare la validazione iniziale, fungendo da filtro tra l'attore esterno e lo strato di logica business.
- **Separazione**: Separando la validazione del testo dalla registrazione del punteggio, il sistema risulta più robusto.

inserisciRecensione



L'operazione ***inserisciRecensione*** gestisce la creazione materiale del feedback e la sua corretta archiviazione all'interno di **TravelEasy**, assicurando che ogni commento sia ricollegabile al cliente che lo ha generato.

Flusso delle Operazioni e Logica Interna

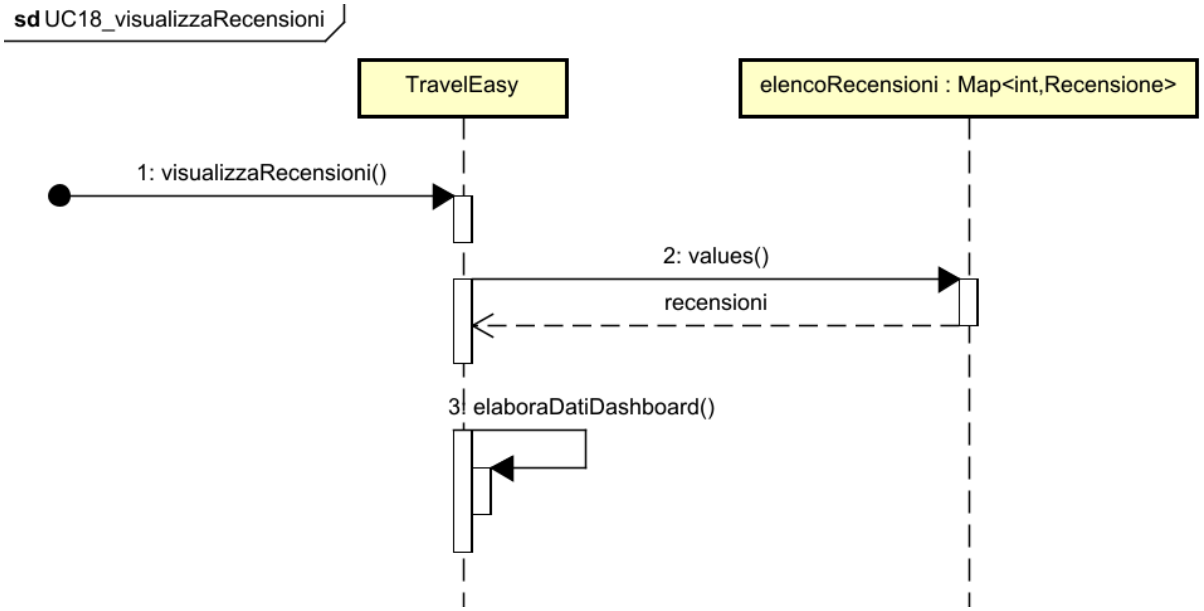
1. **Istanziamento dell'Oggetto *(<<create>>)***: Il controller ***TravelEasy*** riceve i parametri (cliente *c*, prenotazione *p*, stelle, commento e riferimento) e istanzia un nuovo oggetto ***r : Recensione***. L'istanza viene creata specificatamente per il "riferimento" indicato, che rappresenta l'entità valutata (Agenzia, Alloggio o Trasporto) e aggiunta al Database.
2. **Archiviazione Globale**: Una volta creato l'oggetto, il controller lo aggiunge alla collezione di sistema ***elencoRecensioni*** (una *Map<int, Recensione>*) tramite il messaggio ***add(idR, r).*** Questo passaggio garantisce la persistenza della recensione a livello di database globale del sistema.
3. **Aggiornamento del Profilo Cliente (Information Expert)**: Per mantenere la tracciabilità dei feedback rilasciati da ogni utente, il controller invoca il metodo ***addRecensione(idR, r)*** sull'oggetto ***c : Cliente***.
    - L'oggetto *Cliente*, agendo come **Information Expert** del proprio storico, inserisce il riferimento della recensione nella sua mappa interna ***elencoRecensioniInserite***.

Elementi di Design

- **Integrità dei Legami**: L'operazione consolida l'associazione tra l'esperienza di viaggio (prenotazione *p*) e il giudizio finale, impedendo che vengano inserite recensioni "orfane" o non verificate.
- **Principio di Creazione**: Il controller si assume la responsabilità di creare l'oggetto *Recensione* poiché possiede tutte le informazioni necessarie provenienti dall'interfaccia, agendo come orchestratore del processo.
- **Persistenza dei dati**: L'utilizzo delle classi DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.
- **Pattern Observer**: Per migliorare l'esperienza utente (UX) e garantire la coerenza visiva immediata, è stato implementato il **Pattern Observer** all'interno del flusso di inserimento recensioni. Ricevuta la notifica di "inserimento avvenuto", l'interfaccia intercetta l'evento e attiva una procedura di aggiornamento locale con cui il pulsante **"Inserisci Recensione"** diventa **"Visualizza Recensione"** in modo da permettere al Cliente di poter visualizzare la recensione inserita.

Caso d’uso UC18 - Visualizzazione Recensioni, diagramma di interazione

visualizzaRecensione



L'operazione ***visualizzaRecensioni()*** descrive il processo logico attraverso cui il sistema aggrega i feedback puntuali per fornire all'Operatore una visione analitica e statistica della qualità dei servizi.

Flusso delle Operazioni e Logica Interna

1. **Accesso alla Persistenza**: Il controller ***TravelEasy*** riceve la richiesta e interroga l'oggetto ***elencoRecensioni***, una mappa che contiene tutte le istanze delle recensioni registrate nel sistema. Attraverso il messaggio ***values()***, il controller recupera l'intera collezione di oggetti *Recensione*.
2. **Elaborazione Analitica**: Una volta ottenuta la lista delle recensioni, il controller esegue un'operazione interna chiamata ***elaboraDatiDashboard()***.
    - Durante questa fase, il sistema itera su ogni recensione per estrarre i punteggi relativi ad Agenzia, Trasporto e Alloggio.
    - Vengono calcolate le medie aritmetiche e viene conteggiato il numero totale di feedback (*totRecensione*) per popolare i parametri richiesti dal diagramma di sistema.
3. **Presentazione dei Risultati**: Il risultato dell'elaborazione è la produzione di un set di dati pronto per essere visualizzato dall'interfaccia dell'Operatore, garantendo che le decisioni di business siano basate su dati aggregati in tempo reale.

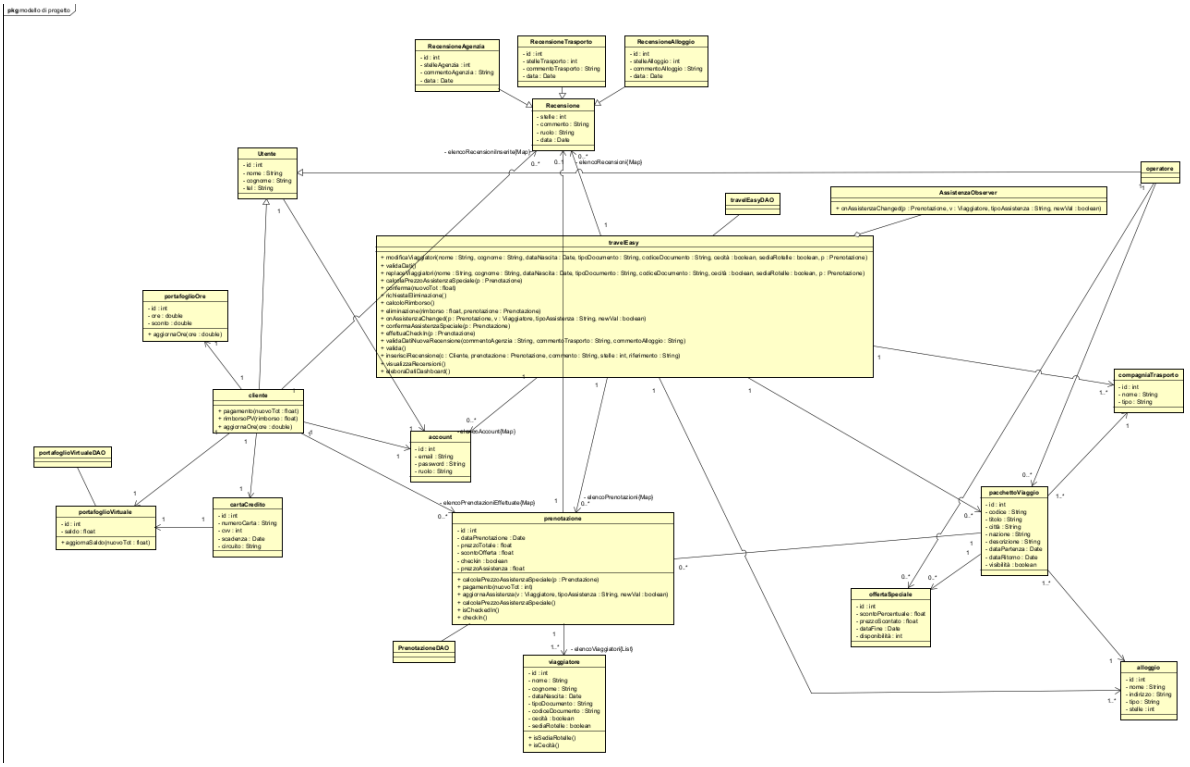
Elementi di Design

- **Separazione tra Dati e Logica**: Il design mantiene una netta separazione tra la memorizzazione (la mappa *elencoRecensioni*) e l'elaborazione dei dati (il metodo del controller). Questo permette di modificare gli algoritmi di calcolo delle medie (ad esempio introducendo pesi diversi per recensioni più recenti) senza alterare la struttura dei dati.
- **Efficienza**: Recuperando tutti i valori in un'unica operazione ***(values()***), il sistema ottimizza l'accesso alla memoria, riducendo il numero di chiamate necessarie per generare il report complessivo.

- **Centralizzazione delle Responsabilità**: Il controller *TravelEasy* agisce come coordinatore, assicurando che la trasformazione dai dati grezzi (le singole recensioni) alle informazioni di business (le medie della dashboard) avvenga in modo coerente e centralizzato.

Diagramma delle classi di progetto

Il diagramma delle classi di progetto rappresenta la struttura statica del sistema **TravelEasy**, definendo le entità di dominio, le loro responsabilità (metodi) e le relazioni che permettono di soddisfare i requisiti funzionali.



Per gestire in modo granulare i feedback degli utenti (UC15 e UC18), è stata implementata una gerarchia di specializzazione:

- **Classe Base *Recensione***: Contiene gli attributi comuni come *stelle, commento, ruolo e data.*
- **Specializzazioni**: Le classi ***RecensioneAgenzia, RecensioneTrasporto e RecensioneAlloggio*** permettono di mappare i feedback.
- **Associazioni**: Il controller *travelEasy* mantiene un riferimento a ***elencoRecensioni***, mentre ogni *Cliente* tiene traccia del proprio ***elencoRecensioniInserite***.

L'introduzione dell'interfaccia ***AssistenzaObserver*** (collegata all'operatore) evidenzia l'adozione del pattern **Observer**.

- Questa struttura permette al sistema di notificare automaticamente gli osservatori registrati quando avvengono cambiamenti critici (come l'attivazione di un'assistenza speciale in una prenotazione), garantendo che la UI e gli operatori siano sempre sincronizzati.

Per separare la logica di business dalla gestione dei dati, sono stati introdotti i **Data Access Objects (DAO)**. Classi come ***travelEasyDAO, portafoglioVirtualeDAO e PrenotazioneDAO*** incapsulano le

query al database, permettendo al sistema di essere indipendente dalla tecnologia di storage sottostante.

Revisione

Dall'analisi dei casi d'uso relativi alla presente iterazione è emersa la necessità di apportare una modifica al caso d'uso UC13: Accumulo ore, prevedendo il caso in cui esso venga chiamato in seguito all'eliminazione di una Prenotazione.

UC13: Accumulo ore

Attore Primario: Cliente

Scenario Principale:

1. Il Sistema riceve conferma del completamento del caso d'uso **UC3: Prenotazione pacchetto vacanza**
2. Il Sistema aggiorna le ore accumulate del Cliente
3. Il Sistema verifica che il totale delle ore raggiunga o superi multipli di 10
4. Il Sistema registra lo sconto da applicare alla prenotazione successiva
5. Il Sistema sottrae le ore utilizzate per generare lo sconto dal totale

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

1. Il Cliente aggiorna la pagina
2. Il Cliente effettua nuovamente l'accesso

1a. La prenotazione non va a buon fine:

1. Il Sistema mostra il messaggio "prenotazione non riuscita"
2. Il Cliente tenta nuovamente la prenotazione, ritornando al passo 1

1b. Il sistema riceve conferma del completamento del caso d'uso UC5: Elimina prenotazione

1. Il sistema converte lo sconto attuale del Cliente in ore e le somma a quelle accumulate
2. Il Sistema sottrae le ore della prenotazione eliminata dal totale
3. Il Sistema verifica che il totale delle ore raggiunga o superi multipli di 10
4. Il Sistema registra lo sconto da applicare alla prenotazione successiva
5. Il Sistema sottrae le ore utilizzate per generare lo sconto dal totale

3a. Il totale delle ore non dà diritto ad uno sconto:

1. Il Sistema aggiorna il numero di ore
2. Il caso d'uso termina

4a. La registrazione dello sconto non va a buon fine:

1. Il Sistema mostra il messaggio "errore nel calcolo dello sconto"

2. Il Sistema torna al passo 3 e ripete le operazioni

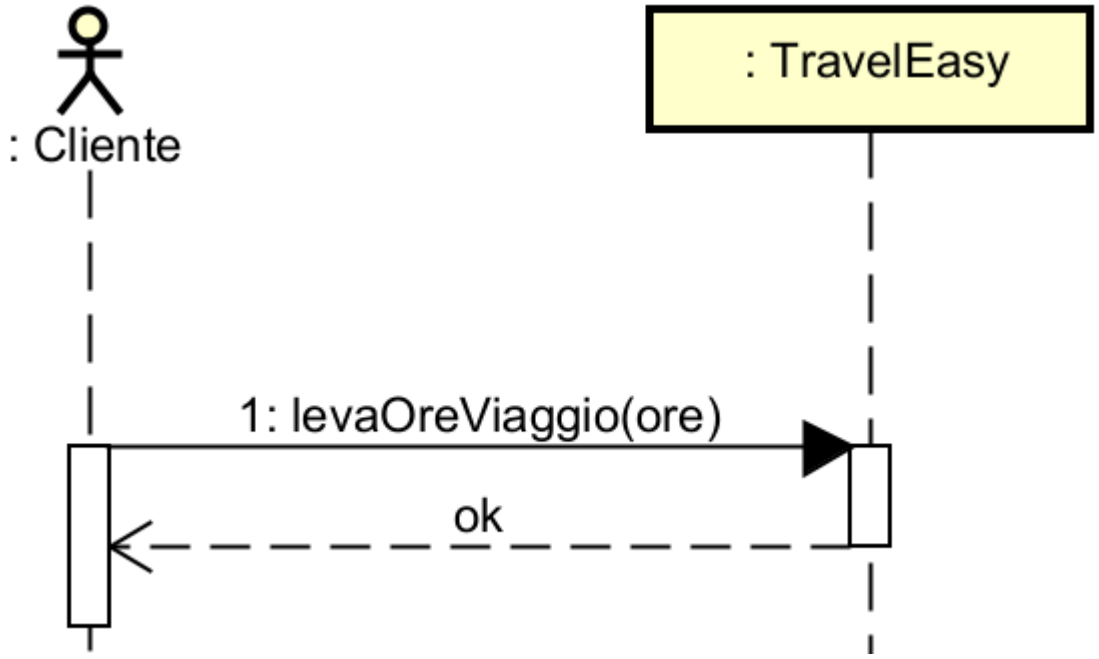
5a. La sottrazione delle ore non va a buon fine:

1. Il Sistema mostra il messaggio “errore nel calcolo delle ore”
2. Il Sistema ripete il passaggio

Regole di dominio

Ogni 10 ore di volo il cliente ha diritto al 3% di sconto sul prossimo pacchetto che acquista.

Diagramma di sequenza di sistema dello scenario alternativo 1b



Contratti delle operazioni

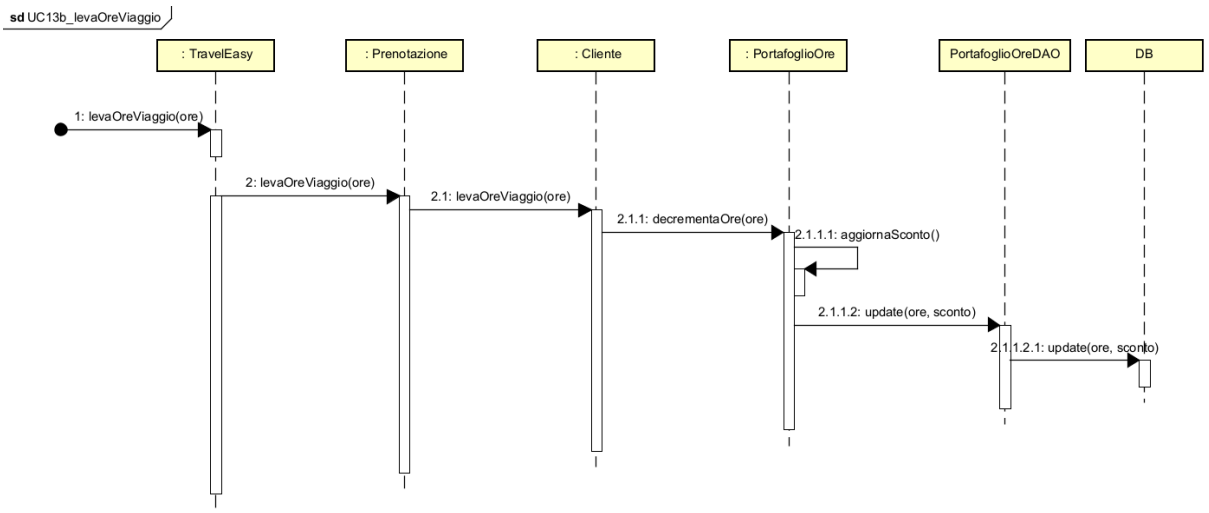
levaOreViaggio

L'operazione di sistema ***aggiornaOreViaggio*** consente al **Sistema** di aggiornare le ore di viaggio contenute nel **PortafoglioOre** del **Cliente** che ha effettuato una prenotazione.

Operazione	aggiornaOreViaggio(ore: int)
Riferimenti	Caso d’uso: *Accumulo ore*
Pre-condizioni	- il Cliente è autenticato - è in corso l’eliminazione di una prenotazione
Post-condizioni	- è stata identificata l’istanza *po* di PortafoglioOre connessa al Cliente autenticato - l’attributo “ore” di *po* è stato aggiornato - l’attributo “sconto” di *po* è stato aggiornato

Diagrammi di interazione

levaOreViaggio



Flusso delle Operazioni e Logica Interna

1. **Recupero dell'oggetto**: Il controller ***TravelEasy*** riceve il messaggio iniziale contenente il parametro ore. Il controller procede mandando un messaggio *aggiornaOreViaggio(...)* alla ***Prenotazione*** appena eliminata dal caso d'uso **UC5**. La ***Prenotazione*** manda dunque un messaggio *levaOreViaggio(...)* al proprio ***Cliente***.
2. **Aggiornamento ore**: Il ***Cliente*** manda un messaggio al ***PortafoglioOre*** *decrementaOre(...)*. Il ***PortafoglioOre*** procede dunque ad aggiornare il proprio monte ore sottraendo le ore perse tramite la ***Prenotazione*** cancellata.
3. **Aggiornamento e applicazione dello sconto**: Una volta aggiornate le proprie ore, il ***PortafoglioOre*** procede a calcolare eventuali sconti disponibili sulla base del nuovo monte ore. Infine provvede a salvare nel database i nuovi dati.

Elementi di Design

- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di aggiornamento e archiviazione.
- **Gestione dei Dati**: L'uso di un'associazione tra ***Prenotazione*** e ***Cliente*** suggerisce una scelta di design volta a ottimizzare il recupero del cliente proprietario di quel portafoglio.
- **Persistenza dei dati**: L'utilizzo delle classi DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.

Testing

In Iterazione 3 il testing ha coperto i casi d'uso UC4, UC5, UC12, UC14, UC15 e UC18.
I test automatici sono stati eseguiti sulle classi TravelEasyModificaPrenotazioneTest, TravelEasyEliminaPrenotazioneTest, TravelEasyAssistenzaSpecialeTest, TravelEasyCheckInTest e TravelEasyRecensioniTest, con esito positivo.

UC4 - Modifica prenotazione

Per UC4, i test verificano sia la modifica del pacchetto associato alla prenotazione sia la modifica dei viaggiatori.

Nel test ***modificaPacchettoPrenotazione_aggiornaPacchettoETotale***, il metodo di testing richiama TravelEasy.getPrenotazioneById(1) e poi TravelEasy.modificaPacchettoPrenotazione(p, nuovoPacchetto) con input:

- prenotazione esistente id 1;
- nuovo pacchetto id 2.

Il metodo di progetto chiamato aggiorna DB e oggetto in memoria.

- Risultato: true, pacchetto della prenotazione aggiornato, totale ricalcolato tramite TravelEasy.getTotalePrenotazione(...), e conferma SQL su Prenotazioni (Pacchetto e PrezzoTotale aggiornati).

Nel test ***modificaViaggiatori_sostituisceInteramenteIRecordDellaPrenotazione***, vengono chiamati:

- TravelEasy.modificaViaggiatori(...) due volte (indice 0 e 1) con nuovi dati anagrafici/documento e flag assistenza (sediaRotelle, cecita);
- Prenotazione.replaceViaggiatoriDB(conn) per persistere la sostituzione completa.

Risultato: entrambi i modificaViaggiatori ritornano 0, in DB restano 2 viaggiatori per la prenotazione e i campi risultano aggiornati (nomi/cognomi e flag assistenza).

Nel test ***annullaPrenotazioneBozza_rimuovePrenotazioneEViaggiatori,*** il flusso usa:

- TravelEasy.createPrenotazione(...);
- TravelEasy.createViaggiatore(...);
- TravelEasy.annullaPrenotazioneBozza(idBozza).

Risultato: annullamento true, prenotazione rimossa da mappa (getPrenotazioneById(...) = null) e rimozione su DB sia dalla tabella Prenotazioni sia da Viaggiatore.

UC5 - Elimina prenotazione

Per UC5 i test verificano il calcolo rimborso e l'eliminazione condizionata ai giorni mancanti.

Nel test ***eliminaPrenotazione_conPiudiSetteGiorni_restituisceSuccessoERimuoveRecord***, la prenotazione viene caricata con getPrenotazioneById(1), poi viene impostato un pacchetto con partenza tra 10 giorni.
Si chiama TravelEasy.getRimborsoEliminazionePrenotazione(p) e poi TravelEasy.eliminaPrenotazione(p, rimborso).

- Risultato: esito 0 (successo), rimozione record da Prenotazioni e Viaggiatore verificata via query SQL.

Nel test ***eliminaPrenotazione_conPartenzaEntroDueGiorni_restituisceNonEliminabile***, con partenza tra 1 giorno il rimborso calcolato è non valido e l'eliminazione ritorna -3.

- Risultato: prenotazione non eliminabile nel vincolo temporale vicino alla partenza.

UC12 - Gestione assistenza speciali

Per UC12 il test gestioneAssistenzaSpeciale_calcolaPrezzoTotaleCorretto usa una prenotazione esistente e i suoi due viaggiatori.

Il flusso di test chiama:

- TravelEasy.onAssistenzaChanged(p, primo, "sediaRotelle", true);
- TravelEasy.onAssistenzaChanged(p, secondo, "cecita", true);
- TravelEasy.confermaAssistenzaSpeciale(p).

I metodi di progetto coinvolti propagano gli aggiornamenti su prenotazione/viaggiatori e invocano Prenotazione.calcolaPrezzoAssistenzaSpeciale().

- Risultato: flag assistenza correttamente impostati e PrezzoAssistenzaSpeciale = 60.0f (35 + 25).

UC14 - Gestione check-in

Per UC14 i test verificano successo nel range consentito, fallimento fuori range e blocco doppio check-in.

Nel test ***checkIn_conPartenzaEntroDueGiorni_riesceEAggiornaDb***, viene creata una prenotazione con partenza tra 1 giorno e chiamato TravelEasy.effettuaCheckIn(p).

- Risultato: true, stato in memoria checkedIn=true e aggiornamento DB (CheckIn=1 su prenotazione id 1).

Nel test ***checkIn_conPartenzaPiuLontanaFallisce***, con partenza tra 6 giorni, effettuaCheckIn(...) restituisce false.

- Risultato: check-in non consentito.

Nel test ***checkIn_giaEffettuato_nonPermetteSecondoTentativo***, il primo tentativo ritorna true, il secondo false.

- Risultato: prevenzione doppio check-in sulla stessa prenotazione.

UC15 - Recensione

Per UC15 i test coprono inserimento recensione e validazione dati.

Nel test ***inserisciRecensione_creaRecordEAggiornaMappaRecensioni***, viene chiamato TravelEasy.inserisciRecensione(cliente, prenotazione, "Ottima esperienza", 5, "Agenzia").

Input:

- cliente cliente@example.com;
- prenotazione id 1;
- commento, stelle e riferimento.

Risultato: true, mappa recensioni in memoria con 1 elemento e tabella Recensione con 1 record.

Nel test ***validazioneNuovaRecensione_conCampiVuoti_fallisceSenzaInserireRecord***, viene chiamato TravelEasy.validaDatiNuovaRecensione("", "Trasporto ok", "Alloggio ok").

- Risultato: false, nessun inserimento in DB (Recensione invariata) e mappa recensioni invariata.

UC18 - Visualizzazione recensioni

Per UC18 il test visualizzaRecensioneLatoOperatore_clienteRecuperaTernaPerPrenotazione inserisce tre recensioni per la stessa prenotazione con riferimenti distinti: "Agenzia", "Trasporto", "Alloggio".

Il flusso chiama ripetutamente TravelEasy.inserisciRecensione(...), poi Cliente.getRecensioneByPrenotazione(p.getId()).
Risultato:

- array recensioni non nullo con 3 elementi;
- presenza di tutti i riferimenti attesi (Agenzia, Trasporto, Alloggio);
- TravelEasy.getNTotaleRecensioni() uguale a 1 (terna completa per una prenotazione).