Ingegneria del Software

Corso di Laurea Magistrale in Ingegneria Informatica

Travel Easy: prima 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 prima iterazione si considerino i seguenti casi d'uso.

UC1: Inserisci nuovo pacchetto viaggio

Attore Primario: Operatore

Scenario Principale:

1. L'operatore inserisce i dati (codice, destinazione, data di partenza e data di ritorno, compagnia di trasporto, struttura alloggio, prezzo del pacchetto, numero di persone, flag per la disponibilità di offerte speciali) nell'apposito modulo
2. L'operatore conferma i dati inseriti
3. Il sistema valida i dati inseriti
4. Il sistema registra il pacchetto
5. Il sistema conferma l'avvenuta registrazione mostrando il codice del pacchetto

Scenari Alternativi:

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

1. L'operatore aggiorna la pagina
2. L'operatore effettua nuovamente l'accesso

3a. Dati non validi o incompleti:

1. Il sistema evidenzia i campi errati o mancanti
2. Si riprende dal passo 3 dello scenario principale

3b. Il pacchetto con le stesse caratteristiche esiste già:

1. Il sistema notifica la duplicazione
2. L'operatore sceglie se modificare i dati inseriti riprendendo dal passo 3 dello scenario principale oppure annullare l'inserimento terminando il caso d'uso

UC2: Ricerca pacchetto vacanza

Attore Primario: Cliente

Scenario Principale:

1. Il cliente seleziona i filtri (destinazione, date andata/ritorno, numero persone, prezzo massimo)
2. Il sistema mostra i risultati della ricerca

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

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

2a. Non ci sono pacchetti disponibili con i filtri inseriti:

1. il sistema mostra un messaggio "non ci sono pacchetti disponibili con i filtri inseriti"

Regole di dominio

Il Sistema mostrerà i pacchetti coincidenti con la destinazione inserita e le date desiderate.

UC6: Gestione offerte speciali

Attore Primario: Operatore

Scenario Principale:

1. Il sistema visualizza l'elenco dei pacchetti disponibili
2. L'operatore seleziona il pacchetto a cui applicare l'offerta
3. Il sistema presenta il modulo per l'inserimento dell'offerta
4. L'operatore inserisce:
    - Percentuale di sconto
    - Data di fine validità dell'offerta
    - Numero di pacchetti disponibili in offerta
5. L'operatore conferma i dati inseriti
6. Il sistema valida i dati dell'offerta

7. Il sistema conferma la pubblicazione dell'offerta

Scenari Alternativi:

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

1. L'operatore aggiorna la pagina
2. L'operatore effettua nuovamente l'accesso

2a. Nessun pacchetto disponibile:

1. Il sistema notifica che non ci sono pacchetti disponibili
2. Il caso d'uso termina

2b. Il pacchetto ha già un'offerta:

1. Il sistema notifica che è già presente un'offerta
2. L'operatore seleziona un altro pacchetto ripetendo il passo 2

6a. Dati dell'offerta non validi:

1. Il sistema evidenzia gli errori (es. prezzo scontato maggiore del prezzo originale, disponibilità nulla o negativa)
2. Il sistema mostra un messaggio di errore
3. Si riprende dal passo 5 dello scenario principale

UC7: Creazione account

Attore Primario: Cliente

Scenario Principale:

1. Il Cliente inserisce:
    - Nome
    - Cognome
    - Email
    - Password
    - Numero di telefono
2. Il Cliente conferma i propri dati
3. Il Sistema convalida i dati
4. Il Sistema registra i dati nel database
5. Il Sistema consente l'accesso all'area personale

Scenari Alternativi:

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

1. Il Cliente aggiorna la pagina
2. Il Cliente inserisce nuovamente le sue informazioni, ritornando al passo 1

3a. L'email è già presente nel database:

1. Il Sistema mostra il messaggio "email già utilizzata"
2. Il Cliente inserisce nuovamente le sue informazioni, ritornando al passo 1

3b. Il campo "password" e il campo "conferma password" non combaciano:

1. Il Sistema mostra il messaggio "password e conferma password non combaciano"
2. Il Cliente inserisce nuovamente le sue informazioni, ritornando al passo 1

3c. Almeno uno dei campi non è stato compilato:

1. Il Sistema mostra il messaggio "per favore compilare tutti i campi"
2. Il Cliente inserisce nuovamente le sue informazioni, ritornando al passo 1

4a. Registrazione nel database fallita:

1. Il Sistema mostra il messaggio "registrazione nel database fallita"
2. Il Cliente inserisce nuovamente le sue informazioni, ritornando al passo 1

UC8: Login

Attore Primario: Cliente/Operatore

Scenario Principale:

1. L'Utente inserisce:
    - Email
    - Password
2. Il sistema verifica la correttezza dei dati
3. Il sistema consente l'accesso all'area personale

Scenari Alternativi :

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

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

2a. I dati inseriti sono errati:

1. Il Sistema mostra il messaggio "email o password errati"
2. L'Utente inserisce nuovamente le sue informazioni, ritornando al passo 1

Regole di dominio

Operatori e Clienti verranno distinti tramite l'attributo "ruolo" e il dominio presente nell'email: l'Operatore avrà una sua email aziendale nel seguente formato "nome.cognome@traveleasy.com". Le email degli Operatori sono già inserite nel sistema.

UC16: Ricarica portafoglio virtuale

Attore Primario: Cliente

Scenario Principale:

1. Il cliente inserisce il CVV
2. Si verifica che il CVV sia corretto
3. Il cliente inserisce il valore della ricarica
4. Il cliente conferma l'importo
5. Il sistema aggiorna il credito del cliente

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

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

1a. Carta di Credito non ancora inserita:

1. Il cliente inserisce i dati della sua carta di credito
2. Il cliente conferma i suoi dati
3. Il sistema verifica la correttezza dei dati
4. Si riprende dal passo 3

1a.3a. Data di scadenza non valida:

1. il sistema mostra il messaggio "la carta è già scaduta"

1a.3b. Formato numero carta non valido:

1. Il sistema mostra il messaggio "il numero della carta non è nel formato corretto"

1a.3c. Formato CVV non valido:

1. Il sistema mostra il messaggio "il CVV non è nel formato corretto"

1a.3d. I dati inseriti sono incompleti:

1. Viene mostrato un messaggio di errore e si ripete il passo 1a.2

2a. Il CVV è errati:

1. Viene mostrato un messaggio di errore e rimanda al passo 1

5a. il pagamento non è andato a buon fine:

1. Il credito non viene aggiornato e il sistema mostra un messaggio di errore

5b. L’importo inserito non è un numero maggiore di 0

1. Viene mostrato un messaggio di errore e rimanda al passo 3

Regole di dominio

Per l’inserimento della carta di credito verranno richiesti numero carta, data di scadenza della carta e CVV.

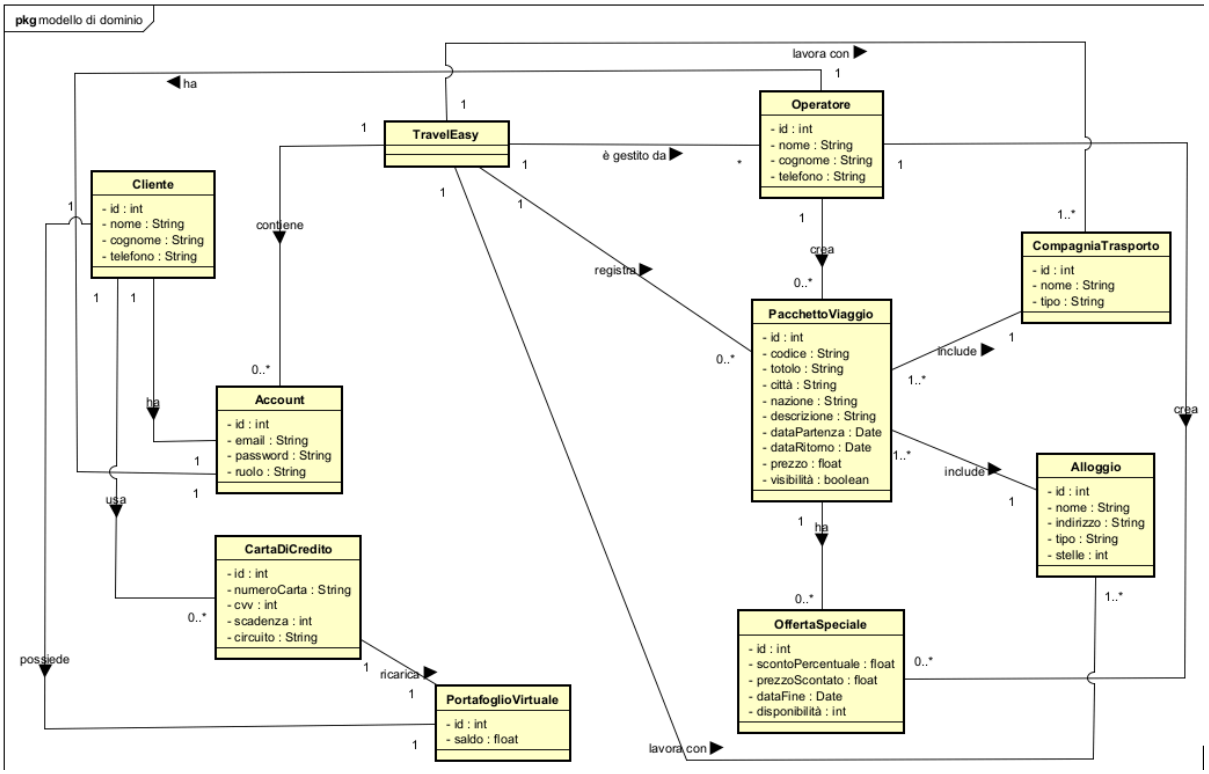
Analisi

Introduzione

Questo capitolo descrive l'analisi svolta nell'iterazione 1, mentre il capitolo successivo descrive l'attività di progettazione.

Modello di dominio

Il modello di dominio rappresenta le entità principali del sistema **TravelEasy** e le relazioni che intercorrono tra esse. Il diagramma riflette la logica di business necessaria per gestire l'intero ciclo di vita di un pacchetto viaggio, dalla creazione alla vendita, includendo la gestione profili e finanziaria.



Entità Principali e Responsabilità

Le classi possono essere raggruppate in tre aree logiche principali:

1. **Area Utenti e Autenticazione (UC7, UC8)**
    - **Account**: Rappresenta l'entità di accesso al sistema. Contiene le credenziali (*email, password*) e definisce il *ruolo* (es. Cliente o Operatore), permettendo di gestire il controllo degli accessi richiesto da **UC8 (Login)**.
    - **Cliente**: Specializzazione dell'utente che interagisce con il sistema per ricercare e acquistare viaggi. È legato univocamente a un **Account** e possiede gli strumenti di pagamento.
    - **Operatore**: Personale dell'agenzia incaricato della gestione del catalogo. Ha il compito di creare l'offerta commerciale.
2. **Area Catalogo e Offerta (UC1, UC2, UC6)**

- **PacchettoViaggio**: Il nucleo del dominio (UC1). Aggrega informazioni su destinazione, date, prezzo e visibilità. È composto da:
    - **CompagniaTrasporto**: Fornitore dei servizi di spostamento.
    - **Alloggio**: Struttura ricettiva (hotel, B&B, ecc.) con relativi dettagli qualitativi (*stelle*).
- **OffertaSpeciale**: Estensione del pacchetto viaggio per la gestione di promozioni (UC6). Introduce attributi come *scontoPercentuale* e *disponibilità* limitata.

3. **Area Finanziaria (UC16)**
    - **PortafoglioVirtuale**: Un deposito fondi associato al **Cliente**. Permette di gestire le transazioni interne al sistema.
    - **CartaDiCredito**: Strumento esterno utilizzato dal cliente per l'operazione di **UC16 (Ricarica portafoglio)**.

Relazioni

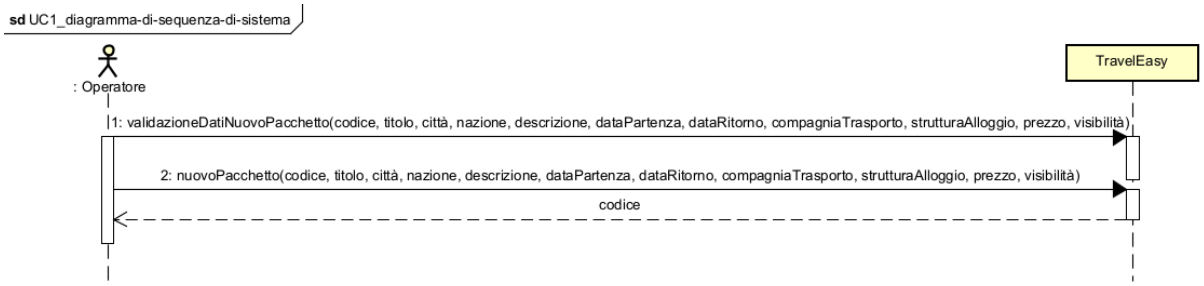
Il diagramma evidenzia legami cruciali per il funzionamento dei casi d'uso:

- **Gestione Pacchetti (UC1):** L'**Operatore** ha una relazione di creazione con **PacchettoViaggio** (0…*). Ogni pacchetto deve necessariamente includere una **CompagniaTrasporto** e un **Alloggio**, garantendo l'integrità dei dati minimi per la vendita.
- **Ricerca e Visibilità (UC2):** Il sistema **TravelEasy** registra i pacchetti. L'attributo *visibilità: boolean* in *PacchettoViaggio* permette di filtrare i risultati durante la ricerca da parte del Cliente.
- **Promozioni (UC6):** La relazione 1 a 0..* tra *PacchettoViaggio* e *OffertaSpeciale* indica che un pacchetto può essere oggetto di più offerte nel tempo.
- **Flusso Finanziario (UC16):** Il **Cliente** "possiede" un **PortafoglioVirtuale**. La ricarica avviene tramite l'entità **CartaDiCredito**, che interagisce con il portafoglio tramite l'associazione "ricarica".

Caso d’uso UC1 - Inserisci Nuovo Pacchetto Viaggio

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema di **UC1** descrive lo scenario principale in cui l'**Operatore** interagisce con il sistema **TravelEasy** per popolare il catalogo dell'agenzia con una nuova proposta di viaggio.



L'interazione si sviluppa attraverso un singolo scambio di messaggi principale che sintetizza l'operazione di inserimento:

1. **Validazione Preliminare (*validazioneDatiNuovoPacchetto(...)*)**: L'Operatore avvia il processo inviando un set completo di parametri che definiscono il pacchetto. Questi includono dati anagrafici (*codice, titolo*), geografici (*città, nazione*), logistici (*dataPartenza, dataRitorno*), riferimenti ai fornitori *(compagniaTrasporto, strutturaAlloggio*) e dettagli commerciali *(prezzo, visibilità*). Il sistema analizza questi input per verificare la coerenza temporale delle date e la validità dei riferimenti ai fornitori esterni.
2. **Creazione e Conferma *(nuovoPacchetto(...)*)**: Una volta superata la fase di controllo, l'attore conferma l'inserimento attraverso il secondo messaggio di sistema, che trasporta nuovamente i metadati del pacchetto per la finalizzazione. Il sistema elabora la richiesta, crea l'istanza nel database e risponde restituendo il ***codice*** identificativo del pacchetto come conferma dell'avvenuta operazione.

Contratti delle operazioni

validazioneDatiNuovoPacchetto

L'operazione ***validazioneDatiNuovoPacchetto(...)*** rappresenta il primo passaggio della realizzazione interna dell'UC1. In questa fase, il sistema agisce come filtro di sicurezza per garantire che i parametri forniti dall'Operatore siano formalmente corretti e coerenti con le regole di business prima di procedere alla creazione dell'oggetto nel dominio.

Operazione	validazioneDatiNuovoPacchetto(codice: String, titolo: String, città: String, nazione: String, descrizione: String, dataPartenza: Date, dataRitorno: Date, compagniaTrasporto: String, strutturaAlloggio: String, prezzo: float, visibilità: boolean)
Riferimenti	UC1: Inserisci nuovo pacchetto viaggio
Precondizioni	- L'operatore è autenticato nel sistema - L'operatore ha avviato il processo di inserimento di un nuovo pacchetto
Postcondizioni	- Il sistema ha verificato che tutti i campi obbligatori siano compilati - Il sistema ha verificato la coerenza dei dati - Il sistema ha verificato che non esista già un pacchetto con le stesse caratteristiche -Il sistema ha verificato che non esista un codice uguale a quello inserito

nuovoPacchetto

L'operazione ***nuovoPacchetto(...)*** rappresenta la fase conclusiva e di persistenza del caso d'uso, in cui il sistema traduce i dati validati in un'entità concreta del dominio, rendendola disponibile nel catalogo dell'agenzia.

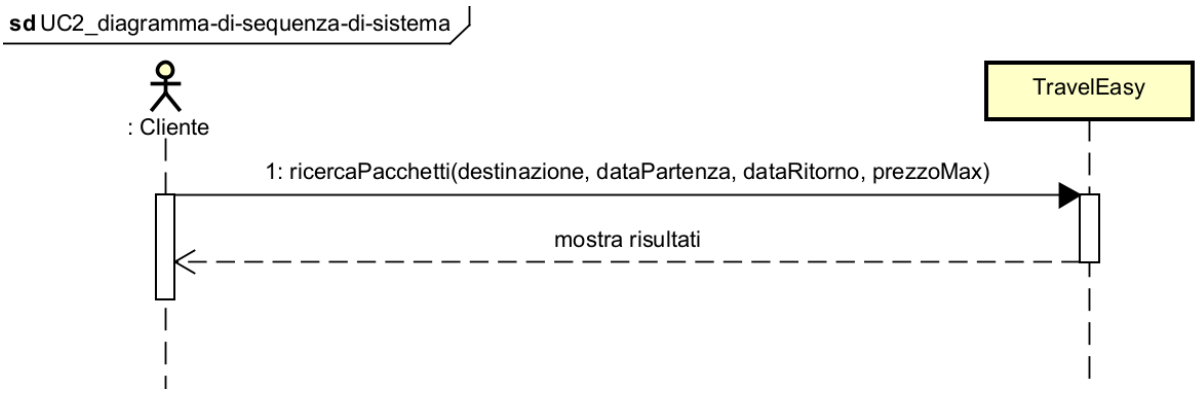
Operazione	nuovoPacchetto(codice: String, titolo: String, città: String, nazione: String, descrizione: String, dataPartenza: Date, dataRitorno: Date, compagniaTrasporto: String, strutturaAlloggio: String, prezzo: float, visibilità: boolean)

Riferimenti	UC1: Inserisci nuovo pacchetto viaggio
Precondizioni	- I dati del pacchetto sono stati validati con successo
Postcondizioni	- È stata creata una nuova istanza di Pacchetto con tutti gli attributi forniti - È stata aggiornata la mappa contenente l'elenco dei pacchetti - p è stato associato a traveleasy tramite l'associazione "registra"

Caso d'uso UC2 - Ricerca Pacchetti

Diagramma di sequenza di sistema

Il **Diagramma di Sequenza di Sistema** per il caso d'uso **UC2** illustra l'interazione tra il **Cliente** e il sistema per filtrare l'offerta del catalogo in base a criteri specifici.



L'operazione di sistema ***ricercaPacchetti*** consente al **Cliente** di avviare la ricerca di un pacchetto vacanza all'interno del sistema **TravelEasy**. Questo diagramma rappresenta l'interfaccia di comunicazione tra l'utente finale e il sistema durante la fase di consultazione del catalogo.

L'interazione è definita da un unico scambio di messaggi che sintetizza la richiesta di filtraggio dei dati:

1. **Chiamata di Sistema (*ricercaPacchetti*)**: Il Cliente invoca l'operazione inserendo i criteri di ricerca che riflettono le sue esigenze di viaggio. I parametri passati sono:
    - ***destinazione***: Filtro geografico.
    - ***dataPartenza*** e ***dataRitorno***: Vincoli temporali per la disponibilità del pacchetto.
    - ***prezzoMax***: Limite economico per filtrare i pacchetti in base all'attributo *prezzo*.
2. **Risposta del Sistema (*mostra risultati*)**: Il sistema **TravelEasy** elabora i criteri e restituisce al Cliente l'elenco dei pacchetti che soddisfano simultaneamente tutti i parametri inseriti.

Contratti delle operazioni

ricercaPacchetti

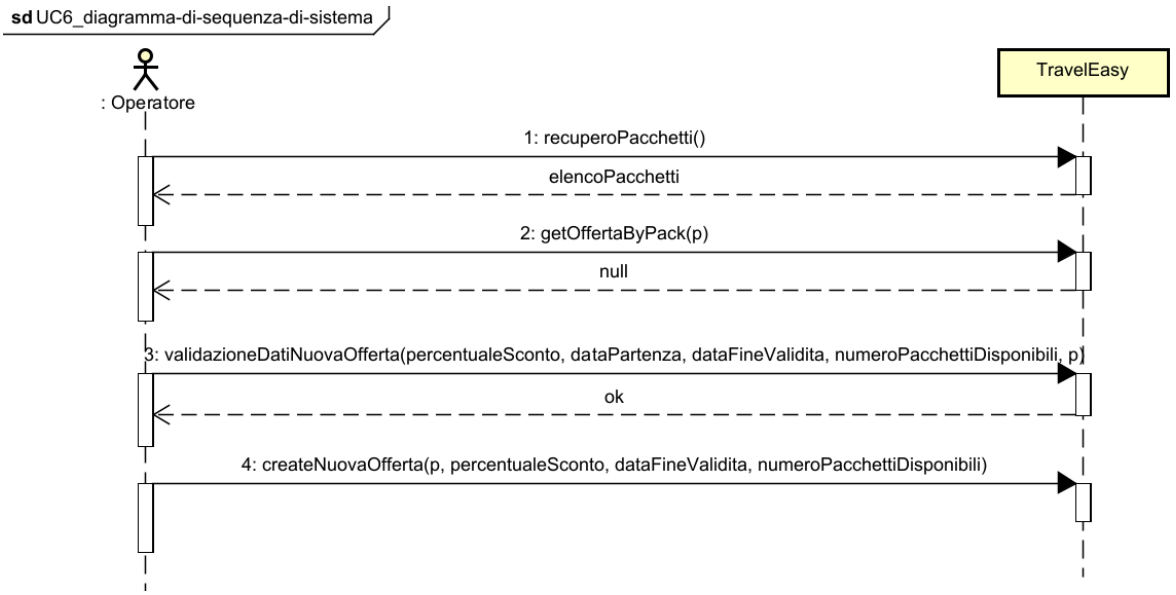
L'operazione di sistema ***ricercaPacchetti*** consente al **Cliente** di avviare la ricerca di un **pacchetto vacanza** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce i parametri fondamentali come la destinazione, la data di partenza, la data di ritorno e il prezzo massimo, ricevendo dal sistema la visualizzazione dei viaggi disponibili.

Operazione	ricercaPacchetti(destinazione:String, dataAndata:Date, dataRitorno:Date, prezzoMassimo:float)
Riferimenti	UC2: Ricerca PacchettoVacanza
Precondizioni	- Il Cliente è autenticato
Postcondizioni	- il sistema ha prodotto una collezione di risultati.

Caso d’uso UC6 - Gestione Offerte Speciali

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema per l'UC6 illustra l'interazione tra l'**Operatore** e il sistema **TravelEasy** per l'applicazione di sconti e promozioni a pacchetti viaggio esistenti. Il processo è progettato per garantire che un'offerta venga creata solo su pacchetti idonei e con parametri validi.



Il flusso si articola nel seguente modo:

1. **Recupero Catalogo *(recuperoPacchetti()*)**: L'operazione inizia con la richiesta da parte dell'Operatore dell'elenco completo dei pacchetti disponibili nel sistema. Il sistema risponde restituendo ***l'elencoPacchetti***, permettendo all'utente di selezionare il pacchetto specifico (**p**) su cui applicare la promozione.
2. **Verifica Offerte Esistenti (*getOffertaByPack(p)*)**: Prima di procedere, l'Operatore interroga il sistema per verificare se il pacchetto selezionato abbia già un'offerta associata. Il sistema restituisce ***null*** se il pacchetto è libero da altre promozioni, condizione necessaria per evitare sovrapposizioni di sconti.
3. **Validazione Parametri Offerta (*validazioneDatiNuovaOfferta(...)*)**: L'Operatore invia i dettagli della nuova promozione: *percentualeSconto, dataPartenza, dataFineValidità, e il numeroPacchettiDisponibili* per l'offerta. Il sistema esegue un controllo di coerenza (es. la

data di fine validità non deve superare la data di partenza del viaggio) e risponde con un messaggio di ***ok***.

4. **Creazione e Associazione (*createNuovaOfferta(...)*)**: L'ultima operazione di sistema finalizza il processo. Il sistema crea l'istanza dell'offerta e la lega al pacchetto **p**, rendendo lo sconto attivo e visibile per i clienti durante la fase di ricerca.

Contratti delle operazioni

recuperoPacchetti

L'operazione di sistema ***recuperoPacchetti*** consente all'**Operatore** di avviare la definizione di una nuova **OffertaSpeciale** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore richiede l'accesso al catalogo dei viaggi esistenti, ricevendo dal sistema l'**elenco dei pacchetti** disponibili per poter successivamente selezionare quello da associare alla promozione.

Operazione	recuperoPacchetti()
Riferimenti	UC6: Gestione offerte speciali
Precondizioni	L'operatore è autenticato nel sistema
Postcondizioni	- Il sistema ha recuperato l'elenco dei pacchetti - L'elenco è stato visualizzato all'operatore

getOggertaByPack

L'operazione di sistema ***getOffertaByPack(p)*** consente all'Operatore di verificare lo stato promozionale di un determinato **PacchettoViaggio** selezionato dal catalogo. Attraverso questo messaggio, l'attore interroga il sistema fornendo come parametro il riferimento al pacchetto (**p**) scelto, al fine di accertarsi che non vi siano sconti già applicati.

Operazione	getOffertaByPach(p: PacchettoViaggio)
Riferimenti	UC6: Gestione offerte speciali
Precondizioni	- è stato selezionato un pacchetto
Postcondizioni	- è stato verificato che il pacchetto non ha offerte attive

validazioneDatiNuovaOfferta

L'operazione di sistema ***validazioneDatiNuovaOfferta*** consente all'Operatore di sottoporre al sistema **TravelEasy** i parametri di una nuova promozione per sottoporli a una verifica di coerenza e integrità. Attraverso questo messaggio, l'attore fornisce i dettagli specifici dell'offerta, quali la **percentuale di sconto**, la **data di inizio**, la **data di fine validità** e il **numero di pacchetti disponibili**, legandoli al pacchetto (**p**) selezionato in precedenza.

Operazione	validazioneDatiNuovaOfferta(percentualeSconto: float, dataPartenza: Date, dataFineValidita: Date, numeroPacchettiDisponibili: int, p: PacchettoViaggio)

Riferimenti	UC6: Gestione offerte speciali
Precondizioni	- L'operatore ha selezionato il pacchetto p - il pacchetto p non ha offerte attive
Postcondizioni	- i dati sono stati validati

createNuovaOfferta

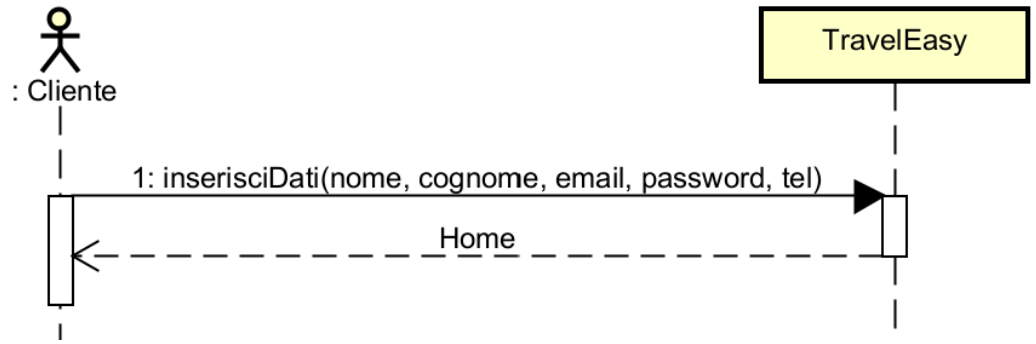
L'operazione di sistema ***createNuovaOfferta*** consente all'Operatore di finalizzare la creazione di una promozione e di renderla effettiva all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore conferma l'associazione definitiva tra il pacchetto selezionato (**p**) e i parametri della promozione (percentuale di sconto, data di fine validità e numero di pacchetti disponibili), innescando la persistenza dei dati nel catalogo.

Operazione	inserimentoDatiOfferta(p: PacchettoViaggio, percentualeSconto: float, dataFineValidita: Date, numeroPacchettiDisponibili: int)
Riferimenti	UC6: Gestione offerte speciali
Precondizioni	- L'operatore ha selezionato il pacchetto p - i dati sono stati validati
Postcondizioni	- È stata creata una nuova istanza o di OffertaSpeciale - Gli attributi di o sono stati inizializzati - il pacchetto p è stato associato all'offerta o - o è stata associata a traveleasy tramite l'associazione "registra"

Caso d'uso UC7 - Creazione Account

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema di **UC7** descrive lo scenario principale in cui il Cliente interagisce con il sistema **TravelEasy** per creare il proprio account.



Contratti delle operazioni

inserisciDati

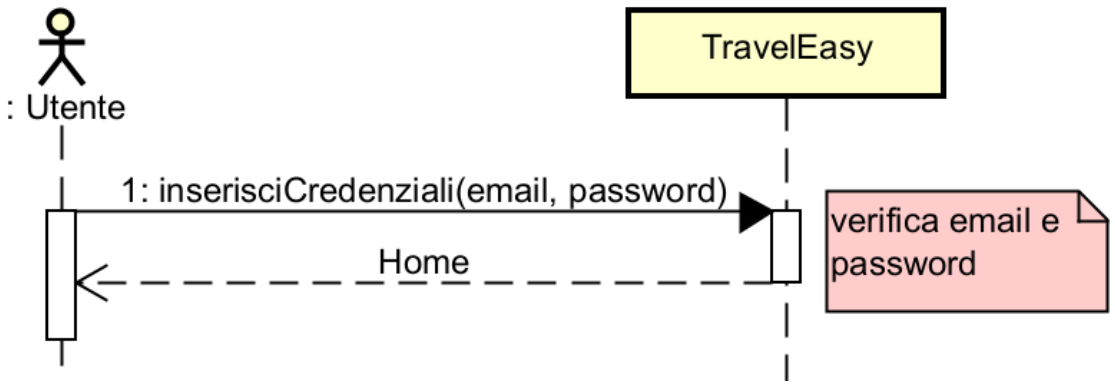
L'operazione di sistema ***inserisciDati*** consente al **Cliente** di avviare la definizione di un nuovo **Account** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce i parametri fondamentali come il nome, il cognome, l'email, la password e il numero di telefono, ricevendo dal sistema la conferma dell'avvenuta creazione tramite il reindirizzamento alla propria area personale.

Operazione	inserisciDati(nome: String, cognome: String, email: String, password: String, tel: String)
Riferimenti	UC7: Creazione account
Precondizioni	
Postcondizioni	- È stata creata una nuova istanza *c* di Cliente - È stata creata una nuova istanza *a* di Account - Gli attributi di *c* sono stati inizializzati - Gli attributi di *a* sono stati inizializzati - *a* è stata associata a *c* tramite l'associazione "possiede" - *a* è stata associata a TravelEasy tramite l'associazione "contiene" - Il Sistema ha verificato la coerenza dei dati - È stata creata un'istanza *pv* di PortafoglioVirtuale - È stata creata un'istanza *cc* di CartaCredito - Gli attributi di *pv* sono stati inizializzati - Gli attributi di *cc* sono stati inizializzati - *pv* è stata associata a *c* tramite l'associazione "possiede" - *cc* è stata associata a *c* tramite l'associazione "usa" - *pv* è stata associata a *cc* tramite l'associazione "ricarica"

Caso d'uso UC8 - Login

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema di **UC8** descrive lo scenario principale in cui l'Utente interagisce con il sistema **TravelEasy** per accedere alla propria area personale tramite le informazioni relative al proprio account.



Contratti delle operazioni

inserisciCredenziali

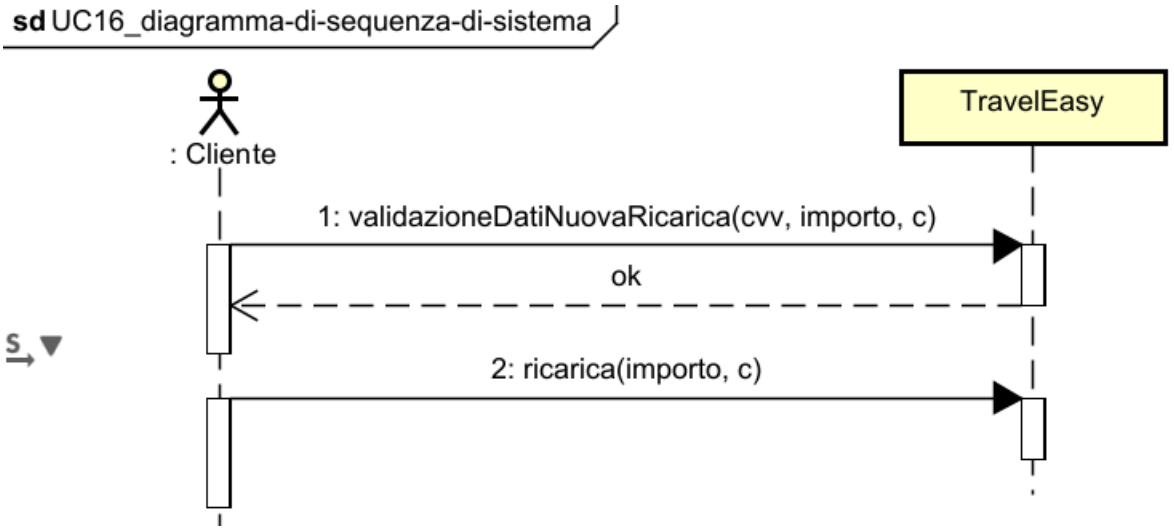
L'operazione di sistema ***inserisciCredenziali*** consente all'**Utente** di accedere al proprio **Account** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce i parametri fondamentali come l'email e la password ricevendo dal sistema la conferma della correttezza delle credenziali inserite tramite il reindirizzamento alla propria area personale.

Operazione	inserisciCredenziali(email: String, password: String)
Riferimenti	UC8: Login
Precondizioni	-
Postcondizioni	- È stato individuato l'account *a* sulla base di email e password inseriti

Caso d'uso UC16 - Ricarica Portafoglio

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema per l'UC16 illustra le interazioni tra il **Cliente** e il sistema **TravelEasy** per l'operazione di ricarica del credito virtuale. L'operazione è progettata come un processo in due fasi per garantire la sicurezza e la conferma della transazione.



- **Inizializzazione e Validazione (*validazioneDatiNuovaRicarica(cvv, importo, c)*)**: L'attore avvia la procedura fornendo tre parametri critici: l'importo desiderato, il codice di sicurezza della carta di credito (**cvv**) e il riferimento al cliente (**c**). In questa fase, il sistema esegue i controlli preliminari sulla validità del metodo di pagamento e sulla congruenza dei dati inseriti. Il sistema risponde con un messaggio di ***ok*** per confermare che la fase di autorizzazione è andata a buon fine.

- **Conferma e Accreditamento *(ricarica(importo, c))***: Dopo aver ricevuto l'autorizzazione, il cliente invia il messaggio finale per completare l'operazione. Questa seconda chiamata agisce come conferma definitiva del trasferimento di fondi. Il sistema riceve l'importo e il riferimento del cliente per procedere all'aggiornamento fisico del saldo nel database.

Contratti delle operazioni

validazioneDatiNuovaRicarica

L'operazione di sistema ***validazioneDati*** consente a **TravelEasy** di effettuare la verifica del cvv. Attraverso questo messaggio, l'attore fornisce i parametri fondamentali come il cvv e l'importo da accreditare, ricevendo dal sistema la conferma dell'avvenuta operazione.

Operazione	validazioneDatiNuovaRicarica(cvv: int, importo: float, c: Cliente)
Riferimenti	UC16: Ricarica Portafoglio
Precondizioni	- Il cliente è autenticato - È stata avviata una ricarica
Postcondizioni	- I dati della carta (cvv) e l'importo sono stati validati formalmente

ricarica

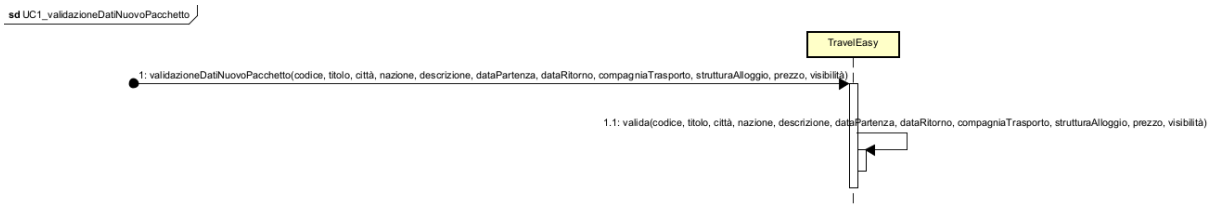
L'operazione di sistema ***ricarica*** consente al **Cliente** di avviare la procedura di **ricarica del proprio PortafoglioVirtuale** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce l'importo da accreditare.

Operazione	ricarica(importo: float, c: Cliente)
Riferimenti	UC16: Ricarica Portafoglio
Precondizioni	- I dati della carta e l'importo sono stati validati con successo
Postcondizioni	- Il saldo del portafoglio virtuale del cliente è stato incrementato del valore *importo*

Progettazione

Caso d’uso UC1 - Inserisci Nuovo Pacchetto Viaggio, diagrammi di interazione

validazioneDatiNuovoPacchetto



L'operazione ***validazioneDatiNuovoPacchetto(...)*** costituisce la fase di controllo che precede l'inserimento di qualsiasi nuova offerta nel catalogo di **TravelEasy**. Questa procedura assicura che il sistema operi solo su dati coerenti, prevenendo errori di integrità nel dominio.

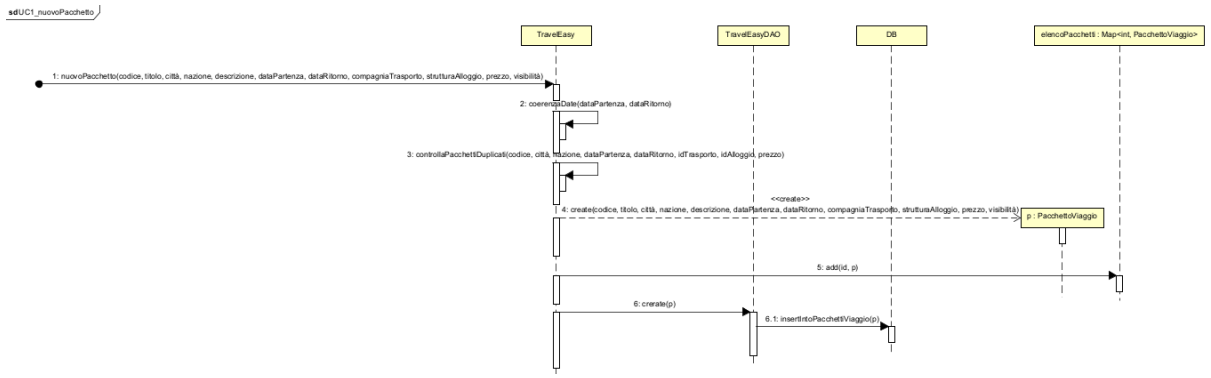
Flusso delle Operazioni e Logica Interna

1. **Ingresso dei Metadati**: Il controller ***TravelEasy*** riceve la chiamata contenente l'intero profilo del pacchetto: *codice, titolo, città, nazione, descrizione*, le date di viaggio, i riferimenti ai fornitori (*compagniaTrasporto, strutturaAlloggio*), il *prezzo* e lo stato di *visibilità*.
2. **Logica di Controllo Interna**: Il controller esegue un messaggio riflessivo ***valida(...)*** per analizzare la validità degli argomenti ricevuti. In questa fase, il sistema verifica internamente:
    - La correttezza del formato dei dati.
    - Il rispetto dei vincoli di business (es. data di ritorno posteriore alla partenza).
3. **Preparazione alla Persistenza**: Il completamento di questa operazione senza sollevare eccezioni è necessaria per la successiva fase di creazione effettiva dell'oggetto nel sistema.

Elementi di Design

- **Separazione delle Responsabilità**: Isolare la validazione in un'operazione distinta garantisce che la classe ***pacchettoViaggio*** venga istanziata solo con parametri già verificati, riducendo la logica di controllo all'interno del costruttore.
- **Robustezza**: Questa struttura protegge la mappa ***elencoPacchetti*** dall'inserimento di istanze corrotte o incomplete che potrebbero causare malfunzionamenti durante la fase di ricerca (UC2).

inserisciNuovoPacchetto



L'operazione ***nuovoPacchetto(...)*** conclude il processo di inserimento, trasformando i dati precedentemente validati in un'istanza persistente all'interno del sistema **TravelEasy**.

Flusso delle Operazioni e Logica Interna

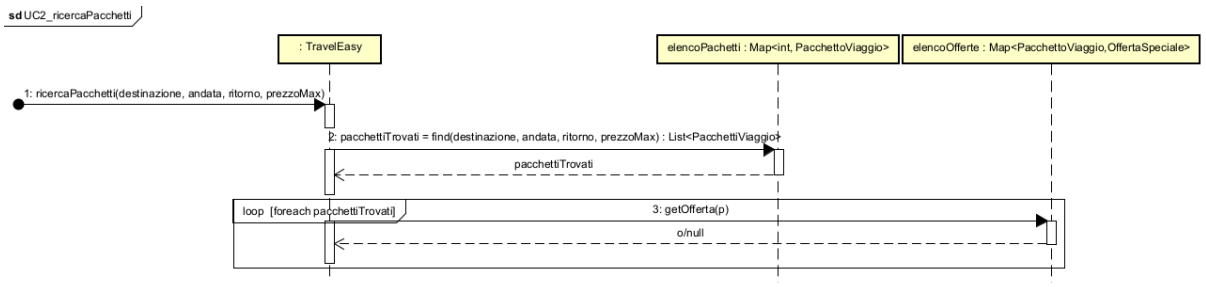
1. **Controlli di Coerenza Finale**: Prima di procedere all'istanziamento, il controller esegue due verifiche cruciali tramite messaggi riflessivi:
    - ***coerenzaDate(dataPartenza, dataRitorno)***: Garantisce che l'intervallo temporale dell'offerta sia logicamente corretto.
    - ***controllaPacchettiDuplicati(...)***: Verifica che non esistano già nel catalogo pacchetti identici per destinazione, date, fornitori e prezzo, evitando ridondanze informative.
2. **Istanziamento (Pattern Creator)**: Superati i controlli, il controller invoca lo stereotipo ***<<create>>*** sulla classe ***p: PacchettoViaggio***. L'oggetto viene creato incapsulando tutti i parametri descrittivi e i riferimenti ai fornitori di trasporto e alloggio.
3. **Memorizzazione nel Catalogo**: L'ultimo step prevede il messaggio ***add(id, p)*** verso l'oggetto ***elencoPacchetti***. Questa collezione, implementata come una ***Map<int, PacchettoViaggio>***, associa il codice identificativo univoco alla nuova istanza, rendendo il pacchetto ufficialmente disponibile per le operazioni di ricerca e vendita.
4. **Memorizzazione nel Database**: l'ultimo step prevede l'inserimento nel Database dell'oggetto *PacchettoViaggio*

Elementi di Design

- **Pattern Information Expert**: La responsabilità della creazione è affidata al controller poiché possiede la visibilità sia sulle regole di business (coerenza date) sia sulle collezioni necessarie per la verifica dei duplicati.
- **Integrità del Dominio**: L'uso di una mappa per l'archiviazione assicura che ogni pacchetto sia rintracciabile tramite un ID univoco, facilitando la gestione delle relazioni.
- **Persistenza dei dati**: L'utilizzo della classe DAO è volta a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento

Caso d’uso UC2, diagrammi di interazione

ricercaPacchetti



Il diagramma di sequenza di **UC2** descrive la logica di interrogazione del catalogo. L’operazione di sistema ***ricercaPacchetti*** non si limita a un semplice filtro sui dati del viaggio, ma integra dinamicamente le informazioni sulle promozioni attive.

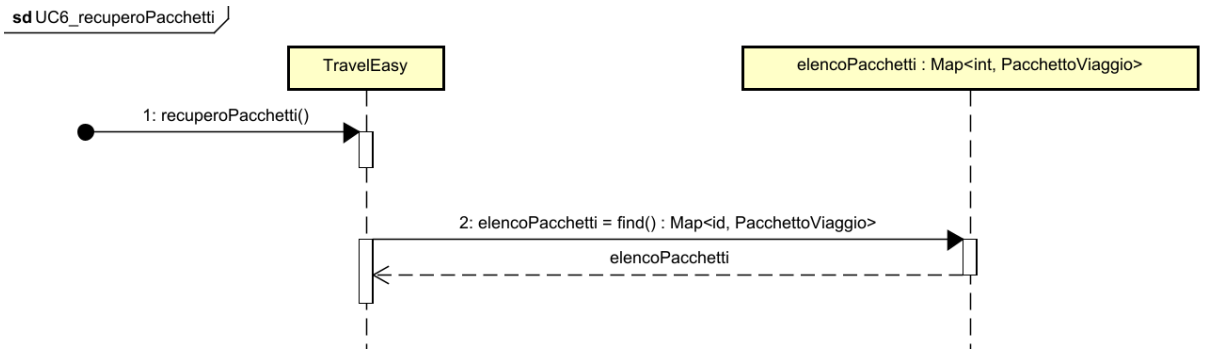
Flusso delle Operazioni e Logica Interna

Il controller ***TravelEasy*** gestisce la richiesta orchestrando l'interazione con le strutture dati globali:

1. **Invocazione del Metodo**: Il sistema riceve i criteri di ricerca (*destinazione, andata, ritorno, prezzoMax*).
2. **Delegazione alla Collezione**: Il controller interroga l'oggetto ***elencoPacchetti***, implementato come una ***Map<int, PacchettoViaggio>***.
3. **Esecuzione del Filtro**: Viene eseguito il metodo ***find(...)*** che analizza le istanze nel catalogo e restituisce una ***List<PacchettoViaggio>*** contenente esclusivamente i match che soddisfano i requisiti di budget e disponibilità temporale.
4. **Risultato Preliminare**: La lista ***pacchettiTrovati*** viene restituita al controller, che la utilizzerà come base per l'arricchimento informativo nel blocco successivo.

Caso d’uso UC6 - Gestione Offerte Speciali, diagrammi di interazione

recuperoPacchetti



Il diagramma di sequenza illustra la logica interna del sistema per la prima fase del caso d'uso **UC6**, ovvero il recupero delle informazioni necessarie dal catalogo per poter successivamente associare un'offerta a un pacchetto specifico.

Flusso delle Operazioni e Logica Interna

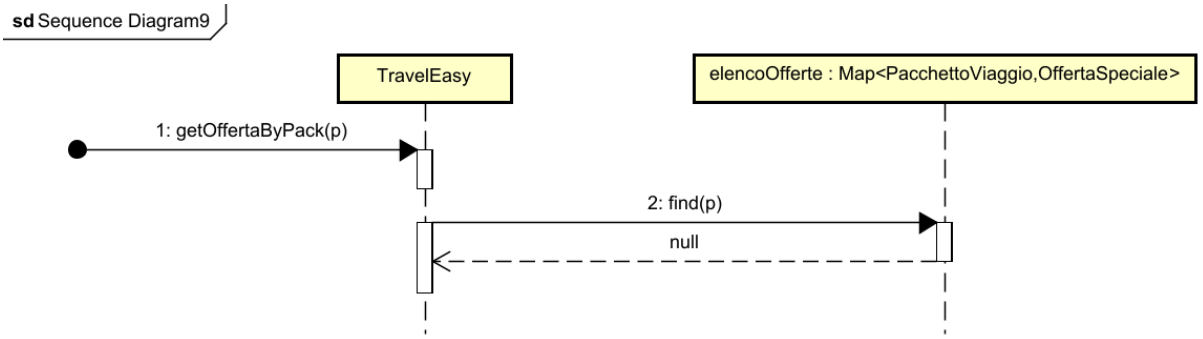
Il flusso descrive il coordinamento tra il controller e la struttura di persistenza dei dati:

1. **Innesco dell'Operazione**: L'operazione ha inizio quando il controller **TravelEasy** riceve il messaggio ***recuperoPacchetti()***. Questo messaggio attiva la logica di ricerca all'interno del sistema.
2. **Accesso alla Struttura Dati**: Per soddisfare la richiesta, il controller interagisce con l'oggetto ***elencoPacchetti***, rappresentato tecnicamente come una ***Map<int, PacchettoViaggio>***. Il controller invoca il metodo ***find()*** sulla mappa per estrarre le istanze dei pacchetti attualmente registrate nel dominio.
3. **Restituzione dei Risultati**: Una volta completata la ricerca nella collezione, i dati vengono restituiti al controller tramite un messaggio di ritorno. Questo permette al sistema di presentare all'operatore l'elenco dei pacchetti pronti per essere scontati, completando la fase di preparazione dell'offerta.

Elementi di Design

- **Disaccoppiamento**: Il diagramma evidenzia come la logica di controllo (*TravelEasy*) sia separata dalla gestione fisica dei dati (*elencoPacchetti*), seguendo i principi di design del software.
- **Efficienza**: L'utilizzo di una *Map* come struttura dati per l'entità *elencoPacchetti* è una scelta di progetto mirata a rendere il recupero dei dati rapido e indicizzato tramite gli identificativi dei pacchetti.

getOffertaByPack



L'operazione ***getOffertaByPack(p)*** costituisce il passaggio logico fondamentale per garantire l'integrità del catalogo promozionale. Il suo scopo è accertare che un pacchetto viaggio non sia già oggetto di una promozione attiva prima di consentire l'inserimento di un nuovo sconto.

Flusso delle Operazioni e Logica Interna

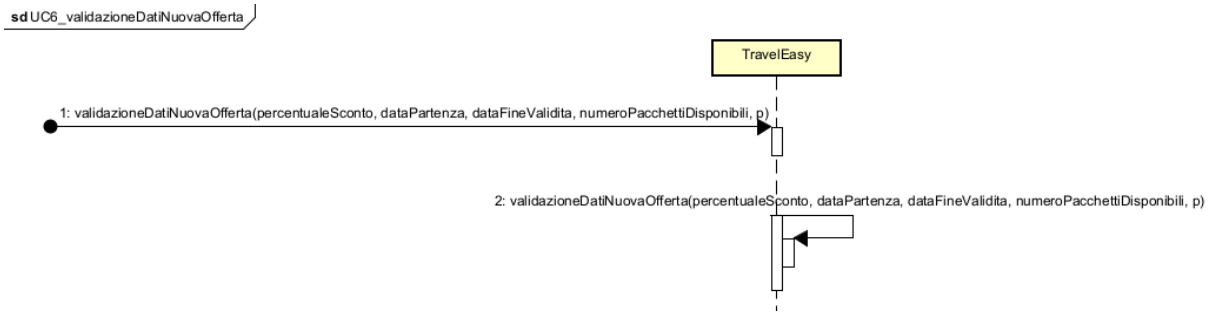
1. **Invocazione del Controller**: Il controller di sistema ***TravelEasy*** riceve la richiesta contenente il riferimento all'oggetto ***p: PacchettoViaggio*** selezionato dall'Operatore.
2. **Interrogazione della Collezione**: Per adempiere alla richiesta, il controller interroga l'oggetto ***elencoOfferte***, che è implementato come una ***Map<PacchettoViaggio, OffertaSpeciale>***.
3. **Ricerca (Pattern Information Expert)**: Viene inviato il messaggio ***find(p)*** alla mappa delle offerte. La scelta di utilizzare il pacchetto come chiave della mappa permette una ricerca immediata ed efficiente.
4. **Esito della Ricerca**:

- Se la mappa restituisce ***null***, significa che non esiste alcuna associazione tra il pacchetto e un'offerta speciale.
- Questo risultato negativo è il prerequisito necessario affinché il controller possa autorizzare l'operatore a procedere con le fasi successive di validazione e creazione della nuova offerta.

Elementi di Design

- **Integrità dei Dati**: Questa operazione implementa un vincolo di business a livello di codice, impedendo la sovrapposizione di più sconti sullo stesso pacchetto, che porterebbe a calcoli errati del *prezzoScontato*.

validazioneDatiNuovaOfferta



L'operazione ***validazioneDatiNuovaOfferta(...)*** rappresenta la fase di controllo logico in cui il sistema verifica che i termini della promozione inseriti dall'Operatore siano coerenti con le politiche commerciali e i vincoli temporali del pacchetto viaggio selezionato.

Flusso delle Operazioni e Logica Interna

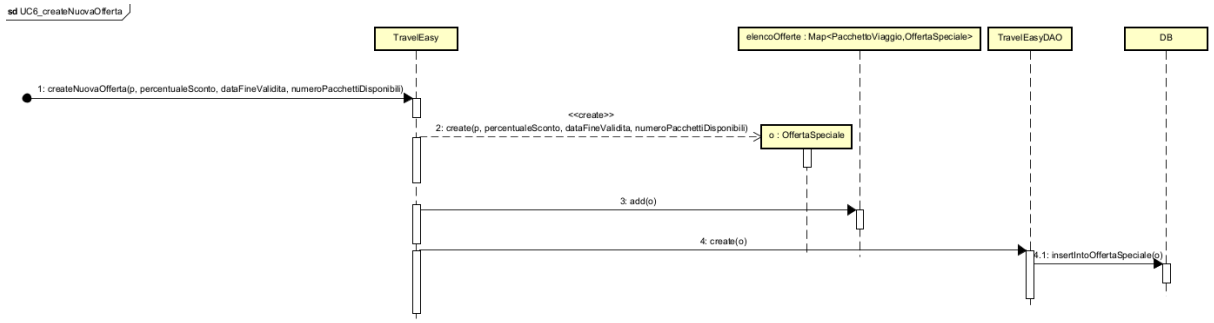
1. **Ricezione dei Parametri**: Il controller ***TravelEasy*** riceve il messaggio contenente i dati definitivi dell'offerta: la *percentualeSconto*, la *dataPartenza* (del pacchetto), la *dataFineValidita* della promozione, il *numeroPacchettiDisponibili* per lo sconto e il riferimento al pacchetto ***p***.
2. **Esecuzione del Controllo (Messaggio Riflessivo)**: Il controller invoca su se stesso il metodo ***validazioneDatiNuovaOfferta(...)***. Durante questa elaborazione interna, il sistema esegue verifiche cruciali come:
    - **Congruenza Temporale**: Accerta che la *dataFineValidita* dell'offerta non sia successiva alla *dataPartenza* del pacchetto.
    - **Integrità Quantitativa**: Verifica che il *numeroPacchettiDisponibili* per l'offerta sia un valore positivo.
    - **Validità dello Sconto**: Controlla che la *percentualeSconto* rientri nel range 1% - 100%.
3. **Conferma di Validità**: Una volta terminati i controlli con esito positivo, il controller è pronto a procedere con l'ultima operazione di sistema (*createNuovaOfferta*), avendo garantito che la promozione non contenga errori formali o logici.

Elementi di Design

- **Robustezza**: Isolare la validazione come operazione atomica impedisce al sistema di tentare la creazione di oggetti *OffertaSpeciale* inconsistenti, preservando la qualità dei dati nel catalogo.

- **Centralizzazione della Logica**: Il controller ***TravelEasy*** agisce come *Expert* della validazione, in quanto è l'unica entità che può confrontare i dati dell'offerta con lo stato attuale del pacchetto e le regole globali del sistema.

createNuovaOfferta



L'operazione ***createNuovaOfferta(p, percentualeSconto, dataFineValidita, numeroPacchettiDisponibili)*** rappresenta l'atto finale di persistenza che concretizza la promozione all'interno del sistema **TravelEasy**.

Flusso delle Operazioni e Logica Interna

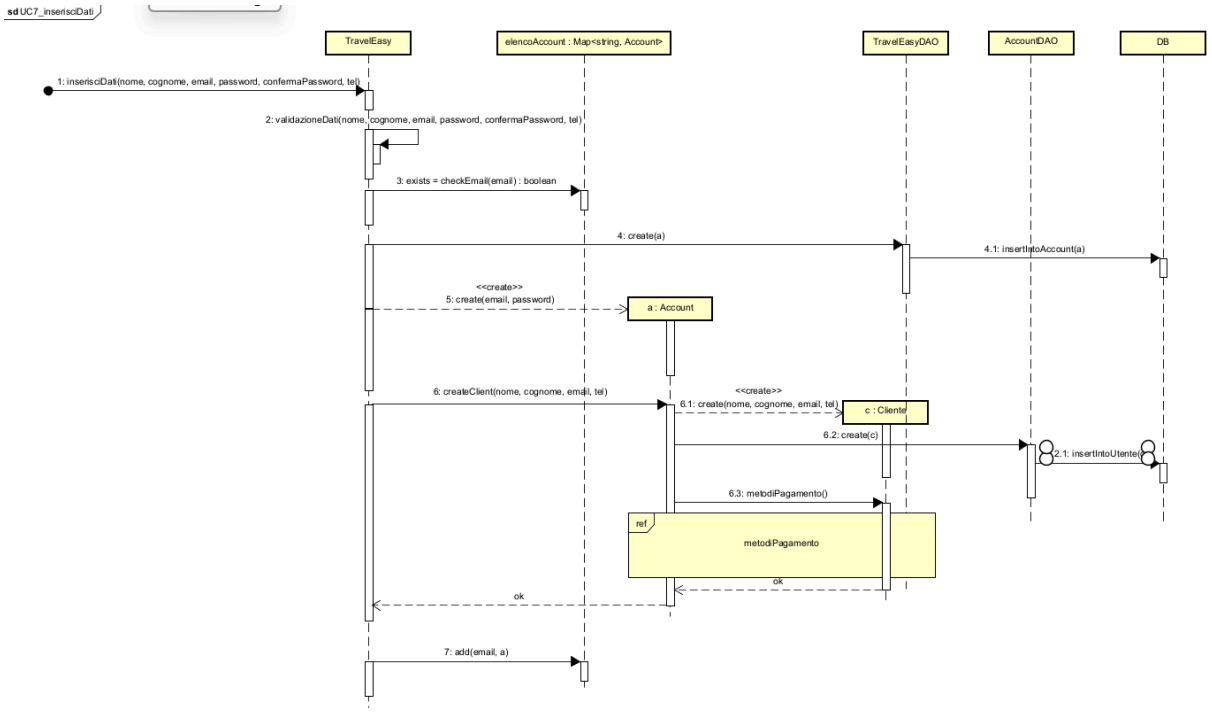
1. **Ricezione del Comando**: Il controller ***TravelEasy*** riceve la richiesta definitiva contenente il riferimento al pacchetto selezionato (**p**) e i parametri della promozione già validati nella fase precedente.
2. **Istanziamento (Pattern Creator)**: Seguendo il principio di creazione degli oggetti, il controller invoca lo stereotipo ***<<create>>*** sulla classe ***o: OffertaSpeciale***. L'oggetto viene istanziato incapsulando la percentuale di sconto, il limite temporale di validità e la quota di pacchetti riservati alla promozione.
3. **Registrazione nel Sistema**: Una volta creato l'oggetto ***o***, il controller provvede alla sua memorizzazione inviando il messaggio ***add(o)*** alla collezione ***elencoOfferte***.
4. **Mappatura e Disponibilità**: Nel modello di dominio, l'offerta viene legata univocamente al pacchetto **p** attraverso una mappa ***(Map<PacchettoViaggio, OffertaSpeciale>***), garantendo che ogni ricerca futura sul catalogo possa calcolare immediatamente il *prezzoScontato* per l'utente finale.
5. **Memorizzazione nel Database**: l'ultimo step prevede l'inserimento nel Database dell'oggetto *OffertaSpeciale*.

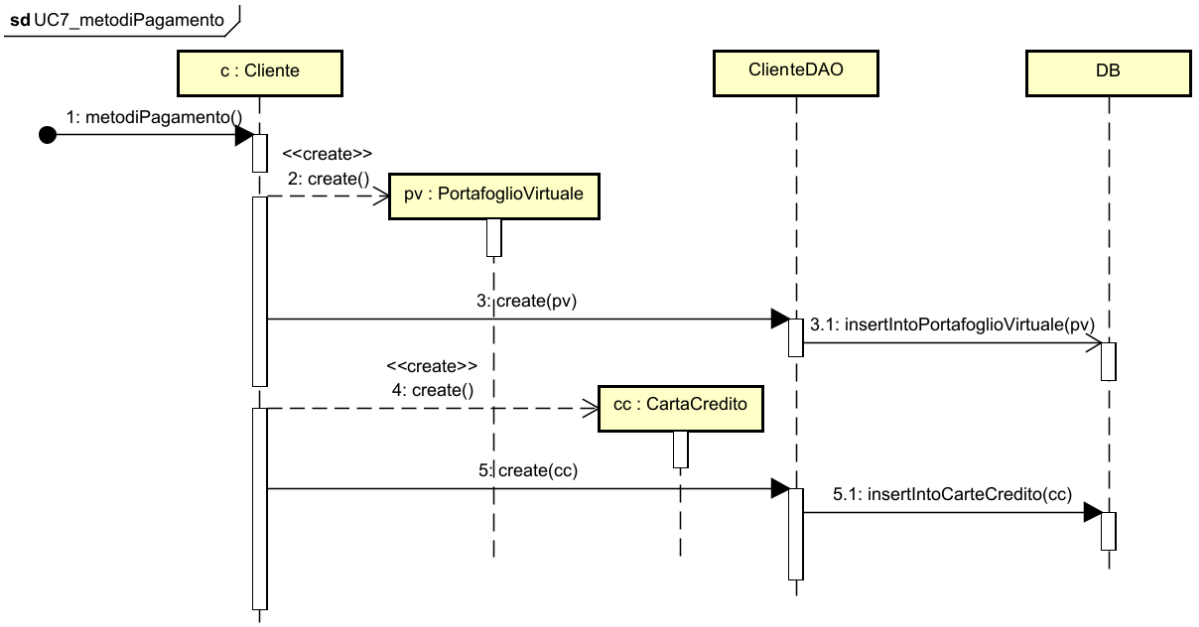
Elementi di Design

- **Information Expert**: Il controller gestisce l'inserimento ***nell'elencoOfferte*** poiché possiede la visibilità globale necessaria per coordinare l'associazione tra pacchetti e promozioni.
- Persistenza dei dati: L'utilizzo della classe DAO è volta a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.
- Per garantire che la creazione di una nuova offerta venga riflessa istantaneamente nelle visualizzazioni attive del sistema (come la ricerca del cliente), è stato implementato il design pattern **Observer**. In questo modo l'offerta diventa visibile e operativa immediatamente migliorando la percezione di efficienza della piattaforma e si evita il rischio di visualizzare prezzi obsoleti (prezzo base invece del prezzo scontato).

Caso d’uso UC7 - Creazione Account, diagrammi di interazione

inserisciDati





Il diagramma di sequenza illustra la realizzazione del caso d'uso attraverso l'interazione tra la classe di controllo e gli oggetti del dominio. L'operazione di sistema *inserisciDati* consente di iniziare la definizione di un nuovo **account** e di un nuovo **cliente** attraverso i seguenti passi logici:

Flusso delle Operazioni e Logica Interna

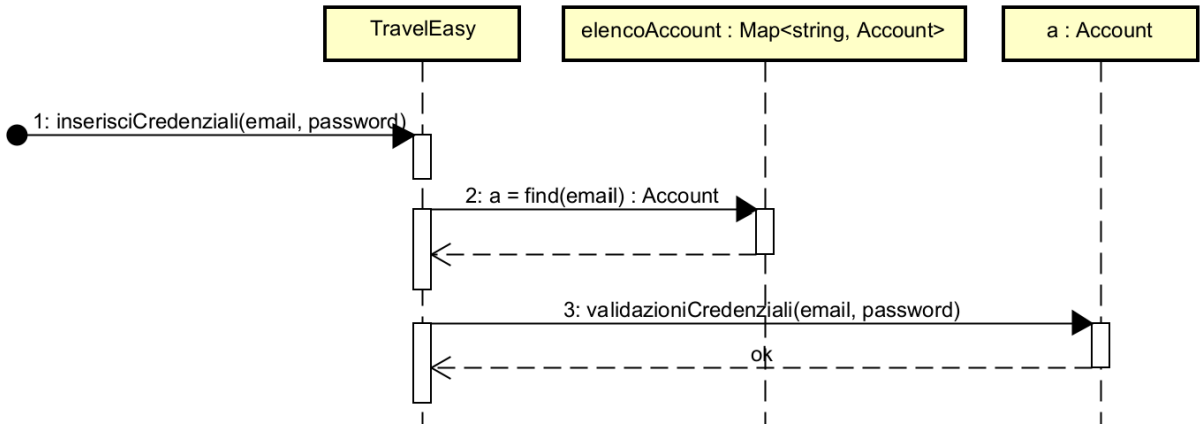
1. **Ricezione e Validazione**: Il controller ***TravelEasy*** riceve il messaggio iniziale completo di tutti i parametri (nome, cognome, email, password, conferma password e numero di telefono). Prima di procedere, il sistema esegue un'operazione interna di *validazioneDati(...)* per garantire la coerenza dei dati inseriti (es. correttezza dei formati, obbligatorietà dei campi e coerenza fra la password e la sua conferma).
2. **Controllo unicità**: Una volta validati i dati, il controller effettua un ulteriore controllo tramite il messaggio *find(...)*. In questa fase il controller sfrutta l'accesso alla propria ***mappa di Account*** per verificare che non esistano altri account associati all'email inserita dal Cliente.
3. **Istanziazione dell'Account**: Una volta validati i dati, il controller invoca il costruttore della classe ***Account*** tramite il messaggio *create(...)*. In questa fase, viene creata una nuova istanza *a* che incapsula l'email e la password fornite dal cliente.
4. **Memorizzazione nel Database**: questo step prevede l'inserimento nel Database dell'oggetto *Account.*
5. **Istanziazione del Cliente**: Il controller manda un messaggio *createClient(...)* all'oggetto *a* appena istanziato, il quale provvede a chiamare il costruttore della classe ***Cliente*** tramite il messaggio *create(...)*. Una volta creata una nuova istanza *c* che incapsula le informazioni fornite dal cliente, *a* manda un messaggio *metodiPagamento()* all'oggetto *c* appena creato, il quale provvede a chiamare i costruttori delle classi ***PortafoglioVirtuale*** e ***CartaCredito***. In questa fase, vengono create delle nuove istanze *c, pv* e *cc* che servono a costituire l'interezza dell'oggetto cliente.
6. **Memorizzazione nel Database**: questo step prevede l'inserimento nel Database degli oggetti *Cliente, PortafoglioVirtuale e CartaCredito.*
7. **Archiviazione e Persistenza**: Il controllore provvede a inserire l'oggetto *a* nella collezione ***elencoAccount*** tramite il metodo *add(email, a)*. Questa operazione garantisce che il nuovo account sia memorizzato permanentemente nel sistema e reso disponibile per le autenticazioni future.

Elementi di Design

- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di validazione, creazione e archiviazione.
- **Gestione dei Dati**: L'uso di una ***Map<email, Account>*** per *elencoAccount* suggerisce una scelta di design volta a ottimizzare il recupero degli account tramite la loro email univoca.
- **Integrità del Dominio**: La separazione tra la validazione e la creazione assicura che nel sistema vengano istanziati solo oggetti ***Account*** e ***Cliente*** validi e completi.
- **Persistenza dei dati**: L'utilizzo della classe DAO è volta a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.

Caso d’uso UC8 - Login, diagrammi di interazione

inserisciCredenziali



Il diagramma di sequenza illustra la realizzazione del caso d'uso attraverso l'interazione tra la classe di controllo e gli oggetti del dominio. L'operazione di sistema *inserisciCredenziali* consente di verificare l’autenticazione di un utente attraverso i seguenti passi logici:

Flusso delle Operazioni e Logica Interna

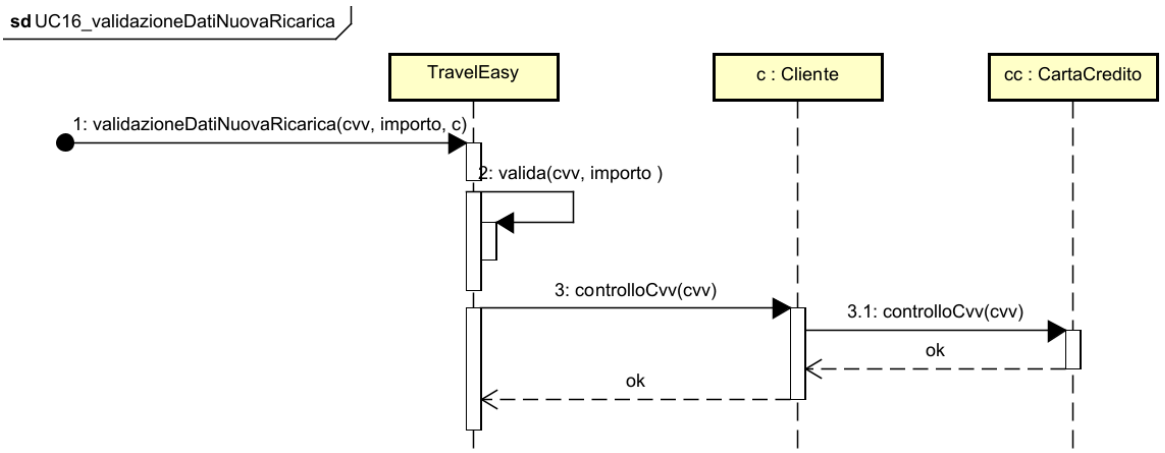
1. **Ricezione e Identificazione**: Il controller ***TravelEasy*** riceve il messaggio iniziale completo di tutti i parametri (email e password). Innanzitutto il sistema identifica l’oggetto ***Account*** *a* corrispondente all’email inserita tramite la mappa ***elencoAccount***, effettuando già un primo controllo dell’esistenza di un ***Account*** associato a quella email.
2. **Validazione**: Il controller procede a mandare un messaggio *validazioneCredenziali(...)* all’oggetto *a* appena identificato, il quale provvede a validare le informazioni inserite dall’utente confrontandole con le proprie. Questa operazione garantisce che l’utente abbia inserito correttamente tutte le informazioni.

Elementi di Design

- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di validazione, creazione e archiviazione.
- **Gestione dei Dati**: L'uso di una ***Map<email, Account>*** per *elencoAccount* suggerisce una scelta di design volta a ottimizzare il recupero degli account tramite la loro email univoca.

Caso d’uso UC16 - Ricarica Portafoglio, diagramma di interazione

validazioneDatiNuovaRicarica



L'operazione ***validazioneDatiNuovaRicarica(cvv, importo, c)*** rappresenta la fase di controllo preliminare necessaria per garantire la sicurezza della transazione finanziaria prima che avvenga l'effettivo addebito.

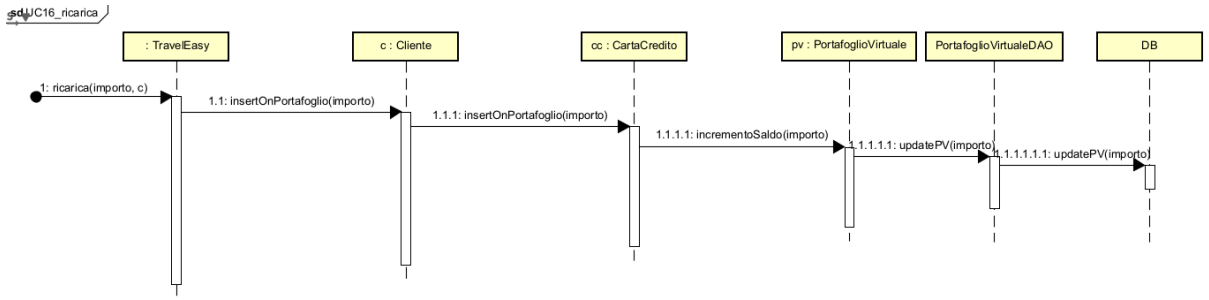
Flusso delle Operazioni e Logica Interna

- **Ricezione della Richiesta**: Il controller ***TravelEasy*** riceve i parametri di input: l'importo da ricaricare, il codice di sicurezza della carta (**cvv**) e il riferimento all'oggetto ***c: Cliente*** che sta effettuando l'operazione.
- **Validazione Interna**: Il controller esegue un messaggio riflessivo ***valida(cvv, importo)*** per verificare la congruenza dei dati (che l'importo sia positivo e il formato del CVV sia corretto).
- **Verifica di Sicurezza (Delegazione)**: Per verificare l'autenticità della carta, il sistema non interroga direttamente la base dati, ma segue la gerarchia del dominio:
    - Il controller invoca ***controlloCvv(cvv)*** sull'oggetto ***c: Cliente***.
    - Il cliente, agendo come tramite, delega la verifica finale alla propria ***cc: CartaCredito*** tramite il messaggio ***controlloCvv(cvv)***.
- **Esito della Validazione**: Una volta che la classe *CartaCredito* conferma la validità del codice, il segnale di ***ok*** risale la catena di chiamata fino al controller, che può così rispondere positivamente all'attore.

Elementi di Design

- **Pattern Information Expert**: La responsabilità di validare il CVV è affidata alla classe ***CartaCredito***, in quanto è l'unica entità a possedere l'informazione originale per il confronto.
- **Basso Accoppiamento**: Il controller comunica solo con l'oggetto *Cliente* e non deve conoscere i dettagli implementativi delle carte di credito associate ad esso, rispettando l'incapsulamento.
- **Preparazione alla Transazione**: Il successo di questa operazione è il requisito fondamentale per l'esecuzione della seconda fase, ovvero l'accreditamento monetario nel portafoglio virtuale.

ricarica



L'operazione ***ricarica(importo, c)*** rappresenta la fase esecutiva del caso d'uso, in cui il sistema procede all'effettivo trasferimento di valore monetario nel portafoglio del cliente, una volta superata con successo la fase di validazione preliminare.

Flusso delle Operazioni e Logica Interna

Il controller ***TravelEasy*** coordina l'aggiornamento del saldo attraverso una catena di deleghe che rispetta la gerarchia delle classi definita nel dominio:

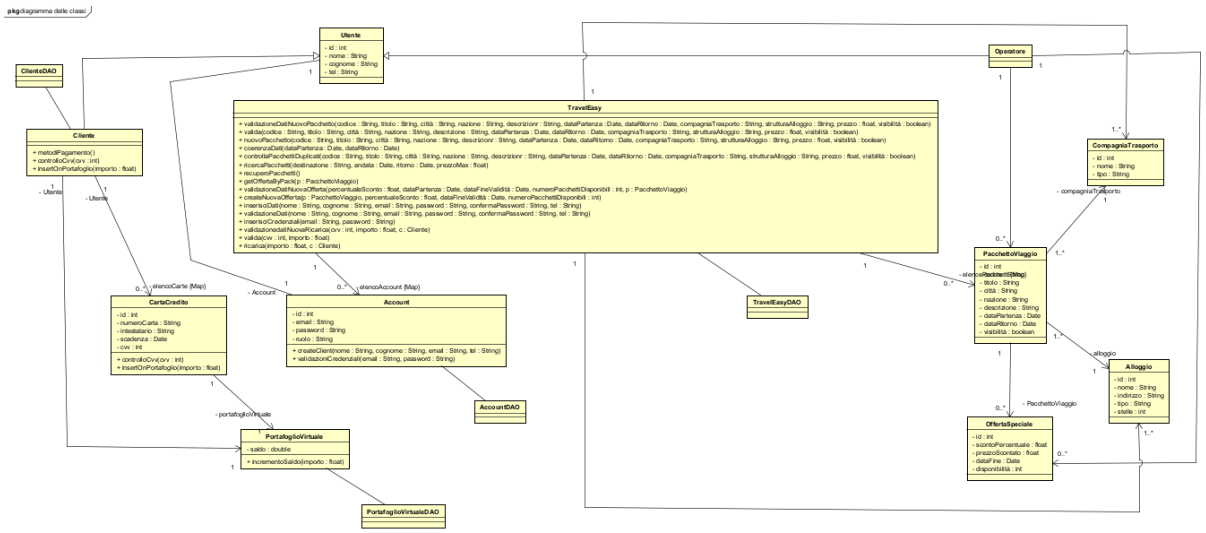
1. **Invocazione del Controller**: Il sistema riceve il messaggio di conferma contenente ***l'importo*** e il riferimento all'oggetto ***c: Cliente***.
2. **Delegazione al Cliente**: Il controller invoca il metodo ***insertOnPortafoglio(importo)*** sull'istanza del cliente. Questa scelta di design mantiene il controller isolato dai dettagli dei metodi di pagamento associati all'utente.
3. **Coinvolgimento della Carta di Credito**: L'oggetto *Cliente* delega ulteriormente l'operazione alla propria ***cc: CartaCredito*** tramite il messaggio ***insertOnPortafoglio(importo***). In questa fase, la carta agisce come l'entità responsabile della transazione finanziaria in uscita.
4. **Aggiornamento Finale del Saldo**: La classe *CartaCredito* invoca infine il metodo ***incrementoSaldo(importo)*** sull'oggetto ***pv: PortafoglioVirtuale***. Questo messaggio modifica permanentemente l'attributo *saldo* all'interno del sistema.
5. **Memorizzazione nel Database**: l'ultimo step prevede l'aggiornamento nel Database dell'oggetto *PortafoglioVirtuale*.

Elementi di Design

- **Pattern Information Expert**: La responsabilità di incrementare il saldo è affidata esclusivamente alla classe ***PortafoglioVirtuale***, che è l'esperta delle informazioni relative ai fondi disponibili dell'utente.
- **Integrità e Tracciabilità**: La sequenza di chiamate assicura che ogni ricarica sia strettamente legata a una carta di credito valida e a un cliente specifico, minimizzando il rischio di errori di puntamento o aggiornamenti di saldi non autorizzati.
- **Persistenza dei dati**: L'utilizzo della classe DAO è volta a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.
- Per migliorare l'esperienza utente ed evitare ricaricamenti manuali della pagina dopo una transazione finanziaria, il sistema adotta il design pattern **Observer**. Questo meccanismo assicura che qualsiasi modifica al credito dell'utente venga propagata istantaneamente a tutte le componenti dell'interfaccia interessate.

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.



La classe **TravelEasy** agisce come controller centrale e "Information Expert" del sistema.

- **Gestione Collezioni**: Mantiene i riferimenti globali a tutte le istanze tramite mappe specializzate: *elencoAccount (Map), elencoPacchetti (Map) ed elencoOfferte (Map)*. Questa struttura garantisce un accesso efficiente ai dati tramite chiavi univoche (come l'ID o l'email).
- **Logica di Business**: Implementa i metodi di coordinamento per la validazione e creazione di pacchetti (*validaDatiNuovoPacchetto, nuovoPacchetto*), la gestione delle offerte (*createNuovaOfferta*) e le operazioni finanziarie (*ricarica*).

Il diagramma evidenzia scelte mirate alla manutenibilità e all'efficienza:

- **Utilizzo di Collezioni Indicizzate**: Per garantire performance ottimali nelle ricerche (UC2) e nei recuperi (UC6), il sistema utilizza strutture dati di tipo **Map** gestite dal controller. In particolare:
    - *elencoPacchetti (Map)*: permette il recupero istantaneo dei viaggi per ID.
    - *elencoCarte (Map)* e *elencoAccount (Map)*: associano in modo univoco gli strumenti di pagamento e le credenziali ai rispettivi utenti.
- **Gerarchia delle Figure**: L'utilizzo della generalizzazione tra **Utente** (classe base) e le specializzazioni **Cliente** e **Operatore** permette di gestire in modo centralizzato i dati anagrafici, differenziando al contempo i privilegi di accesso alle operazioni di sistema.
- **Modularità dei Servizi**: La separazione tra **CompagniaTrasporto** e **Alloggio**, collegate al pacchetto tramite associazioni specifiche, riflette la struttura reale dei fornitori di servizi turistici.
- **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.

Il design della prima iterazione si ispira ai principi **GRASP**. In particolare, l'architettura si poggia su un **Controller** centralizzato (*TravelEasy*) che coordina le operazioni di sistema. Sebbene il controller mantenga un ruolo centrale nella validazione, il sistema dimostra l'applicazione del pattern **Information Expert** delegando logiche specifiche alle entità competenti, come la verifica di sicurezza alla classe *CartaCredito* e la gestione dei saldi al *PortafoglioVirtuale*. L'accoppiamento è mantenuto basso grazie all'impiego di strutture dati a mappa che mediano l'accesso alle istanze di dominio, garantendo al contempo un'ottima scalabilità per l'integrazione di futuri casi d'uso.

Testing

In questa iterazione il testing è stato costruito per verificare i casi d'uso UC1, UC2, UC6, UC7, UC8 e UC16, oltre alle relative regole di dominio. I test sono stati implementati come test automatici JUnit e hanno verificato sia il comportamento dei metodi di dominio (TravelEasy e classi collegate), sia gli effetti sul database (quando previsti).

UC1 - Inserisci nuovo pacchetto viaggio

Per UC1 sono stati implementati test nella classe TravelEasyNuovoPacchettoTest.

Il metodo di test ***validazioneNuovoPacchetto_conDatiValidi_restituisciePrezzo*** richiama il metodo di progetto TravelEasy.validazioneDatiNuovoPacchetto(...) passando in input titolo, città, nazione, descrizione, prezzo in stringa, compagnia, alloggio e intervallo date valido ("20-03-2026" / "25-03-2026"). Il risultato atteso e ottenuto è il prezzo convertito (999.99f), quindi validazione positiva.

Il metodo di test ***validazioneNuovoPacchetto_conCampiVuoti_restituisceMenoUno*** richiama lo stesso metodo di progetto con il titolo vuoto ("") e verifica il codice errore -1.0f, confermando il controllo sui campi obbligatori.

Il test ***coerenzaDate_conPartenzaDopoArrivo_restituisceFalse*** richiama TravelEasy.coerenzaDate(...) con partenza successiva al ritorno ("25-03-2026" > "20-03-2026"). Il risultato atteso e ottenuto è false.

Infine, ***nuovoPacchetto_conDatiValidi_inserisceNuovaRiga*** richiama TravelEasy.nuovoPacchetto(...) con un pacchetto completo e coerente (codice PKG9001, città Londra, prezzo 799.0f, date future). Il metodo ritorna 0 (esito positivo), il numero di record in PacchettiViaggio aumenta di 1 e la riga inserita è verificata con query SQL sul codice pacchetto.

UC2 - Ricerca pacchetto vacanza

Per UC2 i test sono in TravelEasyRicercaPacchettiTest.

Il test ***ricercaPacchetti_conFiltriValidi_restituisciePacchettoAtteso*** richiama TravelEasy.ricercaPacchetti(...) con input "Amsterdam", date "17-04-2026" / "20-04-2026" e prezzo massimo 1000.0f. Il risultato ottenuto è una lista con un solo pacchetto (PKG1001), quindi ricerca corretta.

Il test ***ricercaPacchetti_conPrezzoMassimoTroppoBasso_nonRestituisceRisultati*** usa gli stessi filtri con prezzo massimo 800.0f: il metodo restituisce lista vuota, quindi il filtro prezzo funziona.

Il test ***ricercaPacchetti_nonMostraPacchettiNonVisibili*** richiama lo stesso metodo cercando Tokyo con prezzo massimo alto (3000.0f), ma ottiene comunque lista vuota perché il pacchetto seed Tokyo non è visibile.

Il test ***coerenzaDate_conDateCorrette_restituisceTrue*** richiama TravelEasy.coerenzaDate(...) con due date coerenti ("20-03-2026", "21-03-2026") e ottiene true.

Regole di dominio (UC1-UC2)

Le regole di dominio coperte in questa parte sono:

- coerenza temporale delle date (coerenzaDate);
- obbligatorietà dei campi in inserimento pacchetto (validazioneDatiNuovoPacchetto);
- filtri funzionali della ricerca (ricercaPacchetti) su prezzo e visibilità.

I risultati ottenuti confermano che le regole vengono applicate correttamente nei casi validi e invalidi testati.

UC6 - Gestione offerte speciali

Per UC6 i test sono in TravelEasyNuovaOffertaSpecialeTest.

I test di validazione richiamano TravelEasy.validazioneDatiNuovaOfferta(...) con diversi input:

- percentuale valida ("15") -> risultato 15.0f;
- percentuale non numerica ("abc") -> -3.0f;
- percentuale vuota ("") -> -1.0f;
- data fine oltre la partenza -> -5.0f;
- numero massimo pacchetti non numerico ("abc") -> -8.0f.

Il test ***nuovaOfferta_conDatiValidi_inserisceRigaInOffertaSpeciale*** richiama TravelEasy.createNuovaOfferta(...) con sconto 20.0f, data fine valida e disponibilità 10. Il risultato è true, la tabella OffertaSpeciale incrementa di una riga e il recupero dell'offerta tramite TravelEasy.getOffertaByPack(...) conferma la percentuale sconto attesa.

Il test ***getOffertaByPack_conPacchettoConOfferta_restituisceOfferta*** verifica che, dato un pacchetto con offerta nel seed, il metodo restituisca un oggetto non nullo.

UC7 - Creazione account

Per UC7 i test sono in TravelEasyRegistrazioneTest.

Il test ***registrazione_conDatiValidi_creaAccountEUtente*** richiama TravelEasy.registrazione(...) con dati anagrafici validi ed email nuova (nuovo.utente@example.com). Il metodo ritorna l'email, Account aumenta di una unità e l'account è recuperabile con getAccountToHomeView(...).

Il test ***registrazione_conEmailDuplicata_restituisceErrore*** passa email già presente (cliente@example.com) e ottiene "errore" senza variazioni nel numero di account.

Il test ***registrazione_conEmailNonValida_restituisceErrore*** passa formato email non valido e ottiene "errore".

Il test ***registrazione_conPasswordDiverse_restituisceErrore*** passa password e conferma diverse e ottiene "errore".

Il test ***registrazione_conDatiValidi_creaPortafoglioVirtuale*** verifica che la registrazione crei anche il metodo di pagamento di base: dopo registrazione valida (wallet.test@example.com), viene recuperato l'id utente e verificata la presenza di una riga in PortafoglioVirtuale.

Il test ***registrazione_conDatiValidi_creaMetodiPagamentoCompleti*** estende il controllo: dopo registrazione valida (metodi.pagamento@example.com) verifica in DB la creazione di CartaCredito inizializzata vuota e PortafoglioOre con Ore=0 e Sconto=0.

UC8 - Login

Per UC8 i test sono in TravelEasyLoginTest e richiamano TravelEasy.login(conn, email, password).

Con credenziali cliente valide (cliente@example.com, pwd123) il risultato è ["cliente@example.com", "Cliente"].

Con credenziali operatore valide (operatore@example.com, admin123) il risultato è ["operatore@example.com", "Operatore"].

Con email inesistente, password errata o campi vuoti, il primo elemento del risultato è "errore".

Questo conferma la corretta distinzione ruolo/identità in caso positivo e la gestione uniforme dei casi negativi.

Regole di dominio (UC7-UC8)

Le regole coperte sono:

- validità formale email;
- unicità email;
- coerenza password/conferma in registrazione;
- autenticazione stretta in login (solo credenziali corrette).

I test dimostrano che i vincoli vengono rispettati, senza effetti collaterali sul DB nei casi di errore.

UC16 - Ricarica portafoglio virtuale

Per UC16 i test sono in TravelEasyRicaricaPortafoglioTest.

Il metodo di test ***validazioneRicarica_conDatiValidi_restituisceImporto*** richiama TravelEasy.validazioneDatiNuovaRicarica(...) con input carta valida, scadenza 12/29, cvv 123, importo 150.50 e cliente esistente: risultato 150.50f.

Sono poi testati casi invalidi sullo stesso metodo:

- campi vuoti -> -1.0f;
- cvv non numerico ("abc") -> -3.0f;
- scadenza non valida ("2029-12") -> -4.0f;
- cvv numerico ma non corrispondente alla carta utente ("999") -> -7.0f.

Il test ***insertCartaCredito_conDatiValidi_aggiornaTabellaCartaCredito*** richiama TravelEasy.insertCartaCredito(...) e verifica con query SQL che i campi carta siano aggiornati in DB.

Il test ***insertCartaCredito_conCartaGiaPresente_aggiornaSenzaDuplicareRecord*** richiama due volte lo stesso metodo su utente già dotato di carta: il numero di record in CartaCredito resta invariato (nessuna duplicazione) e i valori finali corrispondono all'ultimo aggiornamento effettuato.