Ingegneria del Software

Corso di laurea Magistrale in Ingegneria Informatica

Travel Easy: seconda 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.

UC3: Prenotazione pacchetto vacanza

Attore Primario: Cliente

Scenario Principale:

1. Il cliente inserisce nome, cognome, data nascita e documento per ogni viaggiatore
2. Il sistema calcola il totale applicando eventuali sconti fedeltà
3. Il cliente conferma i dati e procede al pagamento
4. Il sistema registra la prenotazione e attiva il caso d'uso **UC13: Accumulo ore**
5. Il sistema mostra un messaggio di conferma della prenotazione

Scenari Alternativi:

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

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

3a. Pagamento fallito: il sistema annulla la prenotazione

3b. saldo del portafoglio insufficiente:

3b.a il cliente ricarica il portafoglio digitale:

1. il sistema attiva il caso d'uso **UC16: Ricarica Portafoglio**

3b.b il cliente decide di non effettuare la prenotazione:

1. il caso d'uso termina

4a. Registrazione fallita:

1. Il sistema effettua un rimborso
2. il sistema rimanda al passo 3

Regole di dominio

Il cliente inserirà il documento selezionandolo tra carta d'identità, patente di guida o passaporto. Il documento verrà riconosciuto tramite il suo codice identificativo. Il pagamento del pacchetto avverrà tramite portafoglio virtuale.

UC9: Elimina account

Attore Primario: Cliente

Scenario Principale:

1. Il Cliente seleziona "Elimina Account"
2. Il Sistema chiede un'ulteriore conferma per procedere
3. Il Cliente conferma e inserisce la propria password
4. Il Sistema elimina l'account e tutte le informazioni relative al Cliente

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

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

3a. Il Cliente decide di non procedere con l'eliminazione dell'account cliccando "Annulla":

1. Il caso d'uso termina

3b. La password non è corretta:

1. Il Sistema mostra il messaggio "password errata"
2. Il Sistema torna al passo 2

4b. L'eliminazione dell'account non va a buon fine:

1. Il Sistema mostra il messaggio "errore nell'eliminazione dell'account"

2. Il Sistema torna al passo 2

UC10: Visualizzazione prenotazioni

Attore Primario: Cliente

Scenario Principale:

1. Il sistema recupera tutte le prenotazioni associate all'account del cliente
2. Il sistema visualizza l'elenco delle prenotazioni ordinate per data (dalla più recente alla più vecchia)
3. Per ogni prenotazione il sistema mostra le informazioni del pacchetto viaggio che è stato prenotato e la data della prenotazione

Scenari Alternativi:

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

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

1a. Nessuna prenotazione presente :

1. Il sistema verifica che non esistono prenotazioni per il cliente
2. Il sistema mostra un messaggio "Non hai ancora effettuato prenotazioni"
3. Il caso d’uso termina

UC11: Visualizza prenotazioni dell’agenzia

Attore Primario: Operatore

Scenario Principale:

1. Il sistema recupera tutte le prenotazioni
2. Il sistema visualizza un elenco completo delle prenotazioni ordinato in modo decrescente per data di partenza
3. Per ogni prenotazione il sistema mostra:
    - Codice pacchetto
    - Data prenotazione
    - Nome, cognome e recapito
    - Viaggiatori
    - Data di partenza e di ritorno

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. Nessuna prenotazione presente:

1. Il sistema verifica che non esistono prenotazioni per l'agenzia
2. Il sistema mostra un messaggio “Nessuna prenotazione presente”
3. Il caso d’uso termina

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

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.

UC17: Visualizza dati cliente

Attore Primario: Cliente

Scenario Principale:

6. Il sistema recupera tutti i dati personali associati all'account del cliente
7. Il sistema visualizza le informazioni del cliente
    - nome
    - cognome
    - email
    - numero di telefono
    - credito del suo portafoglio virtuale
    - ore accumulate
    - sconto disponibile per il prossimo acquisto
8. Il cliente consulta le informazioni visualizzate

Scenari Alternativi:

***a. Interruzione improvvisa del sistema**

3. Il Cliente aggiorna la pagina
4. Il Cliente effettua nuovamente l’accesso

1a. Errore nel recupero dei dati:

1. Il sistema non riesce a recuperare i dati dal database
2. Il sistema mostra un messaggio di errore "Impossibile caricare i dati. Riprova più tardi"
3. Il caso d'uso termina

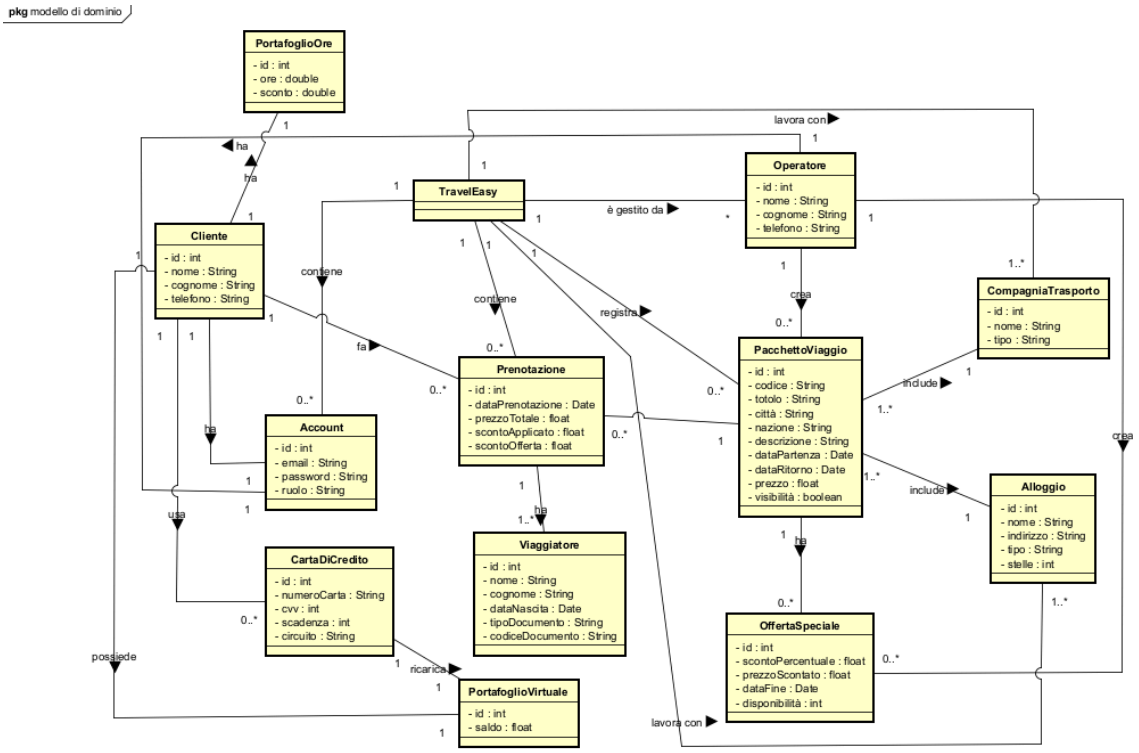
Analisi

Introduzione

Questo capitolo descrive l'analisi svolta nell'iterazione 2, mentre il capitolo successivo descrive l'attività di progettazione.

Modello di dominio

Il modello di dominio della seconda iterazione estende le fondamenta gettate nella fase precedente, introducendo le entità necessarie per gestire il ciclo di vita delle prenotazioni, la profilazione dei viaggiatori e i sistemi di fidelizzazione.



Entità Principali e Responsabilità

L'evoluzione del modello riflette l'introduzione dei casi d'uso legati alla transazionalità e alla gestione avanzata dei dati cliente:

- **Prenotazione**: Rappresenta l'istanza di acquisto di un pacchetto. Questa entità è fondamentale per gli **UC3, UC10 e UC11**. Essa non si limita a collegare il cliente al viaggio, ma storicizza dati dinamici al momento dell'acquisto come:
    - *dataPrenotazione*: per la tracciabilità temporale.
    - *prezzoTotale, scontoApplicato* e *scontoOfferta*: per garantire l'integrità finanziaria anche se i prezzi del catalogo dovessero variare in futuro.
- **Viaggiatore**: Introdotta per supportare la specifica dei partecipanti effettivi al viaggio (che possono differire dal cliente). Include attributi per la conformità legale come *tipoDocumento* e

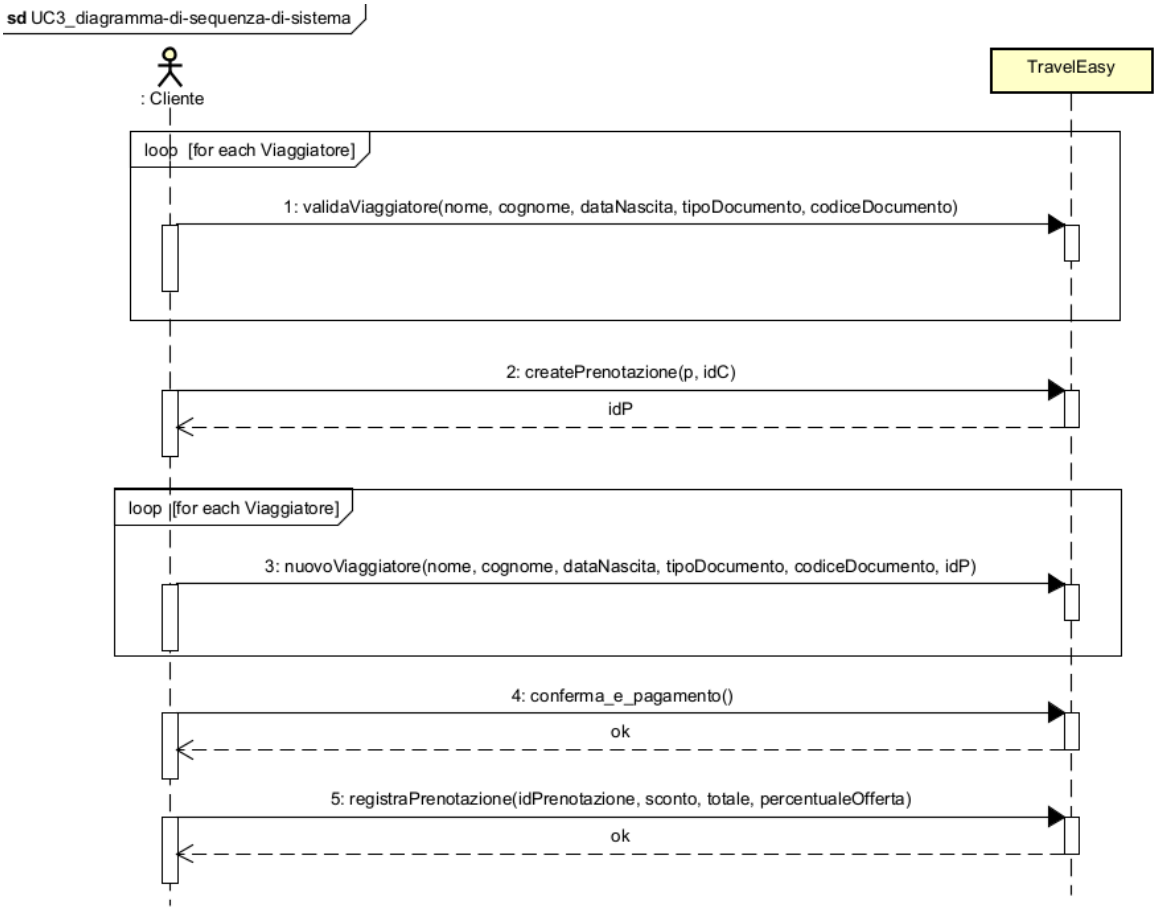
codiceDocumento. La relazione 1..* con la *Prenotazione* permette di gestire gruppi di viaggio all'interno di un'unica transazione.

- **PortafoglioOre**: Questa entità abilita il caso d'uso **UC13 (Accumulo ore)**. Si tratta di un sistema di fidelizzazione dove il cliente accumula *ore* basate sui viaggi effettuati, che si traducono in uno *sconto* percentuale o monetario sui futuri acquisti.

Caso d'uso UC3 - Prenotazione Pacchetto Vacanza

Diagramma di sequenza di sistema

Il diagramma di sistema per l'**UC3** descrive l'interazione tra il **Cliente** e il sistema **TravelEasy** per la formalizzazione di una prenotazione.



Il processo è articolato in cinque fasi principali, incluse strutture iterative per la gestione dei dati anagrafici:

1. **Validazione Preliminare (Loop)**: All'interno di un ciclo *for each Viaggiatore*, il Cliente invia il messaggio ***validaViaggiatore(...)***. Il sistema verifica preventivamente la correttezza dei dati anagrafici e dei documenti di ogni partecipante prima di procedere alla creazione della pratica, garantendo la qualità del dato in ingresso.
2. **Apertura della Pratica**: Una volta validati i soggetti, il Cliente invoca ***createPrenotazione(p, idC)***, fornendo i riferimenti del pacchetto ***(p***) e del cliente (***idC***). Il sistema risponde restituendo l'identificativo univoco della prenotazione ***(idP***), che servirà da riferimento per le operazioni successive.

3. **Associazione Viaggiatori (Loop)**: Attraverso un secondo ciclo iterativo, viene invocato il messaggio ***nuovoViaggiatore(..., idP)*** per ogni partecipante. Questa operazione lega formalmente ogni viaggiatore precedentemente validato alla specifica prenotazione identificata da *idP*.
4. **Transazione Finanziaria:** Il Cliente procede alla chiusura economica con il messaggio ***conferma_e_pagamento()***. Il sistema, dopo aver processato il pagamento tramite i moduli competenti, restituisce un feedback di "ok".
5. **Finalizzazione e Persistenza:** L'ultimo step prevede l'invocazione di ***registraPrenotazione(idPrenotazione, sconto, totale, percentualeOfferta)***. Con questo messaggio, il sistema consolida tutti i dati economici e promozionali, rendendo la prenotazione definitiva e persistente nel database.

Contratti delle operazioni

validaViaggiatore

L'operazione di sistema ***validaViaggiatore*** consente al **Cliente** di avviare la verifica preventiva dei dati anagrafici e documentali di ogni partecipante al viaggio all'interno del sistema **TravelEasy**.

Operazione	validaViaggiatore(nome: String, cognome: String, dataNascita:Date, tipoDocumento: String, codiceDocumento: String)
Riferimenti	UC3: Prenotazione pacchetto vacanza
Precondizioni	- Il cliente deve essere autenticato nel sistema - Deve essere stato selezionato un pacchetto viaggio
Postcondizioni	- I dati del viaggiatore sono stati validati

createPrenotazione

L'operazione di sistema ***createPrenotazione*** consente al **Cliente** di formalizzare l'apertura di una nuova pratica di viaggio all'interno del sistema **TravelEasy**.

Operazione	createPrenotazione(p: PacchettoViaggio, idC: int)
Riferimenti	UC3: Prenotazione pacchetto vacanza
Precondizioni	- Tutti i viaggiatori sono stati validati con successo
Postcondizioni	- È stata creata una nuova istanza di Prenotazione. - La prenotazione è stata associata al Cliente con identificativo idC. - Il parametro p è stato associato alla nuova prenotazione.

nuovoViaggiatore

L'operazione di sistema ***nuovoViaggiatore*** consente al **Cliente** di associare formalmente i partecipanti alla pratica di viaggio precedentemente aperta all'interno del sistema **TravelEasy**.
Attraverso questo messaggio, inserito in un blocco iterativo (**loop**), l'attore fornisce i parametri necessari per ogni singolo partecipante

Operazione	nuovoViaggiatore(nome: String, cognome: String, dataNascita:Date,

	tipoDocumento: String, codiceDocumento: String, idP: int)
Riferimenti	UC3: Prenotazione pacchetto vacanza
Precondizioni	- La prenotazione con identificativo idP esiste nel sistema
Postcondizioni	- È stata creata una nuova istanza di Viaggiatore con gli attributi forniti - L'istanza Viaggiatore è stata associata alla Prenotazione identificata da idP

conferma_e_pagamento

L'operazione di sistema ***conferma_e_pagamento(totale)*** consente al **Cliente** di finalizzare la transazione finanziaria e confermare l'acquisto del pacchetto all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore autorizza il prelievo dell'importo precedentemente calcolato dal proprio **PortafoglioVirtuale**.

Operazione	conferma_e_pagamento(totale: float)
Riferimenti	UC3: Prenotazione pacchetto vacanza
Precondizioni	- Almeno un Viaggiatore è associato alla prenotazione. - Il cliente ha confermato i dati inseriti.
Postcondizioni	- Il saldo del portafoglio virtuale è stato aggiornato

registraPrenotazione

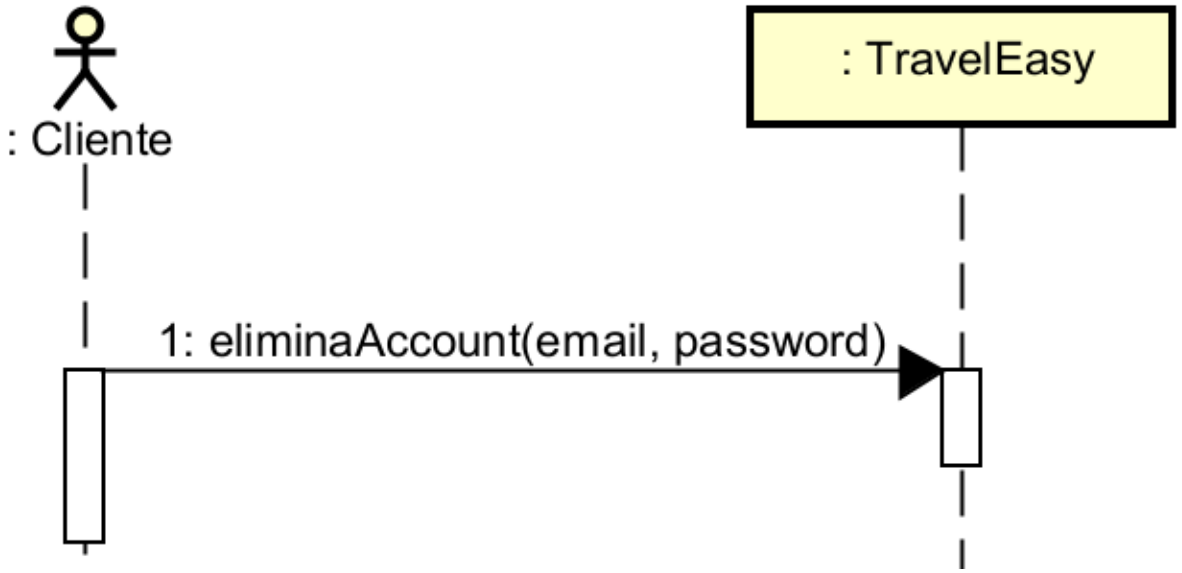
L'operazione di sistema ***registraPrenotazione()*** consente al sistema **TravelEasy** di consolidare e rendere persistente l'acquisto nel database dell'agenzia.

Operazione	registraPrenotazione(idPrenotazione: int, sconto: float, totale: float, percentualeOfferta: float)
Riferimenti	UC3: Prenotazione pacchetto vacanza
Precondizioni	- Il pagamento deve essere stato completato con successo
Postcondizioni	- Gli attributi sconto, totale e percentualeOfferta sono stati registrati sulla Prenotazione

Caso d’uso UC9 - Elimina 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 eliminare il proprio account.



Contratti delle operazioni

eliminaAccount

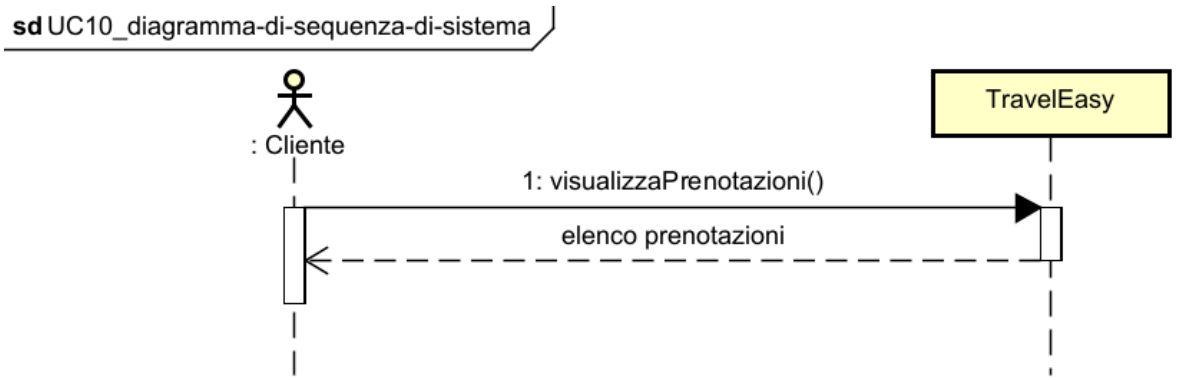
L'operazione di sistema ***eliminaAccount*** consente al **Cliente** di avviare l'eliminazione del proprio **Account** dal sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce la propria password per confermare la propria identità.

Operazione	eliminaAccount(email: String, password: String)
Riferimenti	Caso d'uso: *Eliminazione Account*
Pre-condizioni	- il Cliente è autenticato - la richiesta di eliminazione è stata confermata
Post-condizioni	- è stata identificata l'istanza *a* di Account corrispondente al cliente autenticato - è stata identificata l'istanza *c* di Cliente corrispondente ad *a* tramite l'associazione "possiede" - è stata identificata l'istanza *cc* di CartaCredito corrispondente a *c* tramite l'associazione "possiede" - è stata identificata l'istanza *pv* di PortafoglioVirtuale corrispondente a *c* tramite l'associazione "possiede" - è stata identificata l'istanza *po* di PortafoglioOre corrispondente a *c* tramite l'associazione "possiede" - è stata eliminata l'istanza *po* e tutte le sue associazioni - è stata eliminata l'istanza *cc* e tutte le sue associazioni - è stata eliminata l'istanza *pv* e tutte le sue associazioni - è stata eliminata l'istanza *c* e tutte le sue associazioni - è stata eliminata l'istanza *a* e tutte le sue associazioni

Caso d’uso UC10 - Visualizzazione Prenotazioni

Diagramma di sequenza di sistema

Il diagramma di sistema per l'**UC10** descrive l'interazione mediante la quale un **Cliente** registrato interroga il sistema per consultare lo storico dei pacchetti viaggio acquistati.



L'operazione di sistema ***visualizzaPrenotazioni()*** consente al **Cliente** di avviare la consultazione del proprio archivio viaggi all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore richiede l'accesso ai dati relativi a tutte le transazioni concluse con successo, ricevendo dal sistema la conferma e la visualizzazione dei dati tramite il messaggio di ritorno ***elenco prenotazioni***.

Contratti delle operazioni

visualizzaPrenotazioni

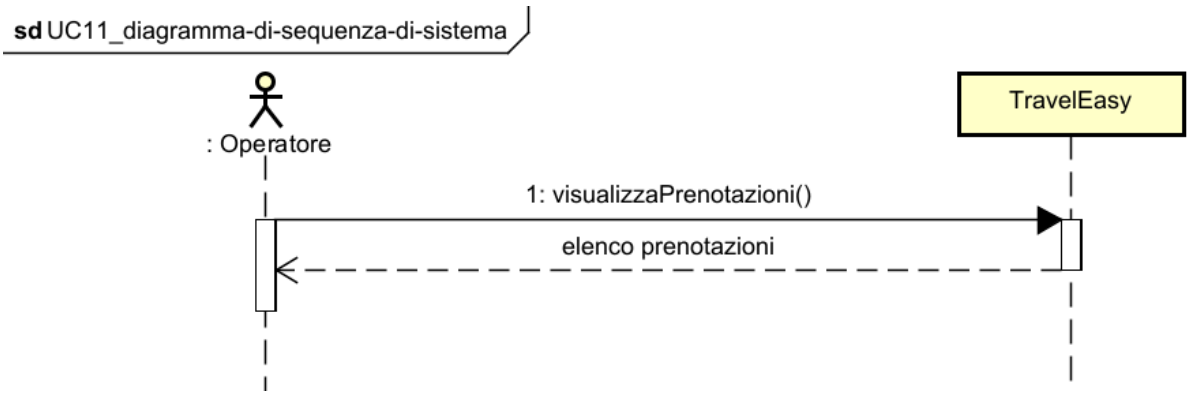
L'operazione di sistema ***visualizzaPrenotazioni()*** consente al **Cliente** di avviare la **consultazione dello storico dei propri viaggi** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore richiede l'accesso ai dati relativi ai pacchetti acquistati, ricevendo dal sistema il riepilogo delle proprie transazioni tramite la restituzione dell'**elenco prenotazioni**.

Operazione	visualizzaPrenotazioni()
Riferimenti	UC10: Visualizzazione prenotazioni
Precondizioni	- Il cliente deve essere autenticato nel sistema
Postcondizioni	- È stata creata una lista di Prenotazioni del cliente ordinata in modo decrescente per data di prenotazione

Caso d’uso UC11 - Visualizza Prenotazioni dell’Agenzia

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema per l'**UC11** descrive l'interazione mediante la quale l'**Operatore** interroga il sistema per ottenere una visione d'insieme di tutte le prenotazioni effettuate dai clienti presso l'agenzia.



L'operazione di sistema ***visualizzaPrenotazioni()*** consente all'**Operatore** di avviare la **consultazione dell'elenco globale delle prenotazioni** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore richiede l'accesso ai dati relativi a tutti i pacchetti viaggio venduti, ricevendo dal sistema il registro completo delle attività dell'agenzia tramite la restituzione dell'**elenco prenotazioni**. A differenza dell'UC10 (lato cliente), dove la ricerca è limitata ai viaggi del singolo utente, l'**Operatore** ha una relazione di lavoro con il sistema che gli permette di accedere alla collezione globale delle istanze di **Prenotazione**.

Contratti delle operazioni

visualizzaPrenotazioni

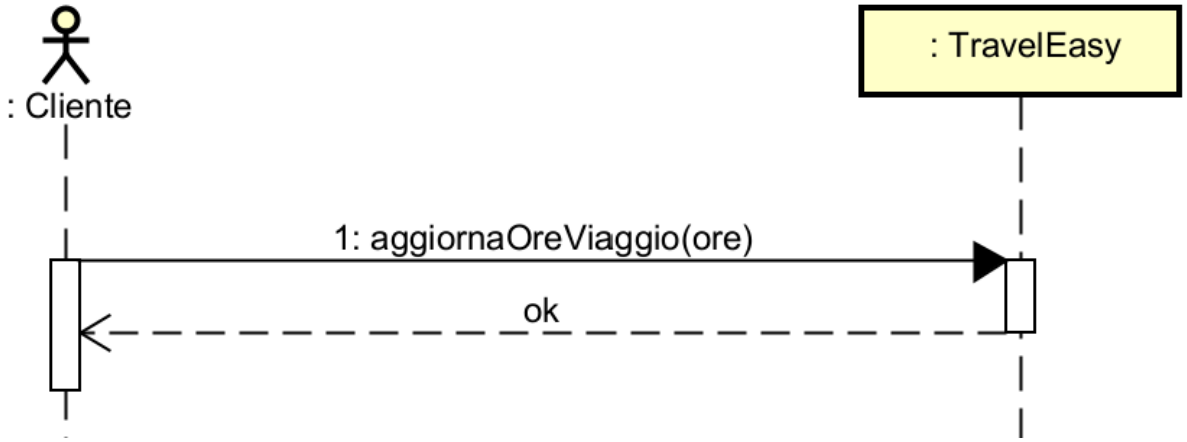
L'operazione di sistema ***visualizzaPrenotazioni()*** consente all'**Operatore** di avviare la **consultazione dell'elenco globale delle prenotazioni** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore richiede l'accesso ai dati relativi a tutti i pacchetti viaggio venduti dall'agenzia, ricevendo dal sistema il registro completo delle prenotazioni effettuate tramite la restituzione dell'**elenco prenotazioni**.

Operazione	visualizzaPrenotazioni()
Riferimenti	UC11: Visualizza prenotazioni dell’agenzia
Precondizioni	- L'operatore deve essere autenticato nel sistema
Postcondizioni	- È stata creata una collezione di istanze Prenotazione ordinata in modo decrescente per data di partenza

Caso d’uso UC13 - Accumulo Ore

Diagramma di sequenza di sistema

Il diagramma di sequenza di sistema di **UC13** descrive lo scenario principale in cui il Sistema aggiorna le ore di viaggio connesse ad un determinato Cliente.



Contratti delle operazioni

aggiornaOreViaggio

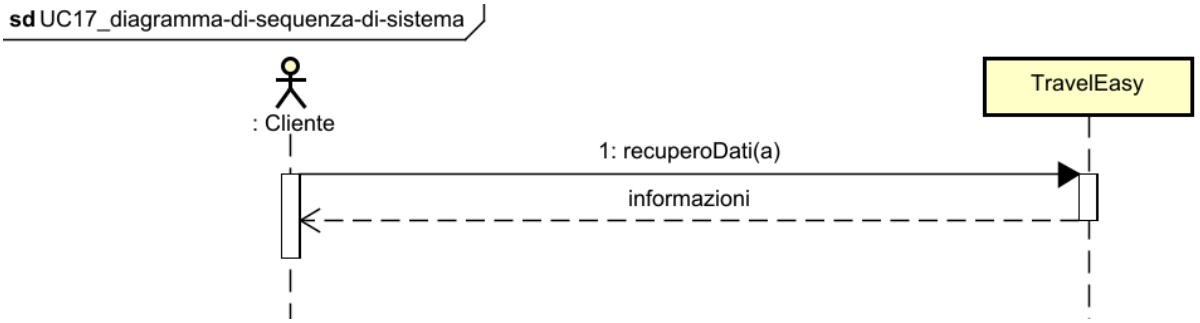
L'operazione di sistema ***aggiornaOreViaggio*** consente al **Sistema** di aggiornare le ore di viaggio contenute nel **PortafoglioOre** del **Cliente** che ha effettuato una prenotazione.

Operazione	aggiornaOreViaggio(ore: int)
Riferimenti	Caso d'uso: *Accumulo ore*
Pre-condizioni	- il Cliente è autenticato - è stata effettuata almeno 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

Caso d'uso UC17 - Visualizza Dati Cliente

Diagramma di sequenza di sistema

Il diagramma di sistema per l'**UC17** descrive l'interazione mediante la quale un **Cliente** richiede al sistema l'accesso alle informazioni personali e tecniche associate al proprio profilo utente.



L'operazione di sistema ***recuperoDati(a)*** consente al **Cliente** di avviare la **consultazione delle informazioni del proprio profilo** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce il parametro identificativo dell'account (**a**).

Contratti delle operazioni

recuperoDati

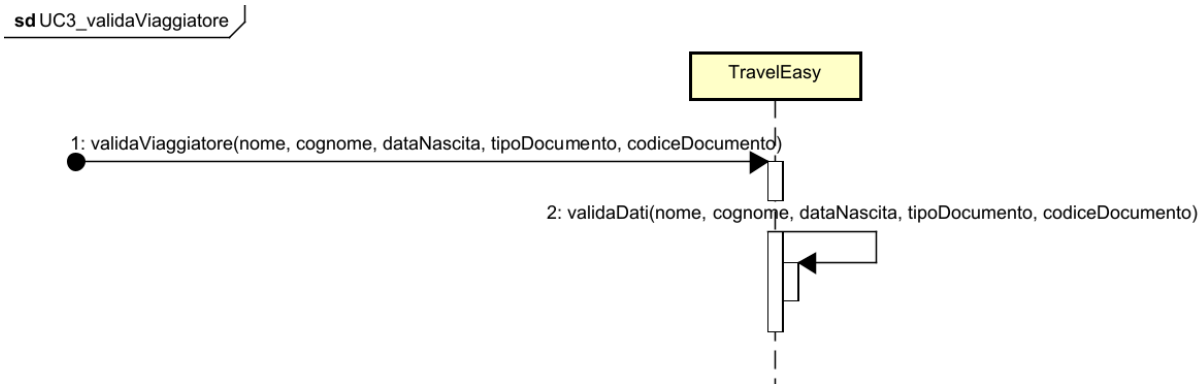
L'operazione di sistema ***recuperoDati(a)*** consente al **Cliente** di avviare la **visualizzazione dei dati relativi al proprio profilo** all'interno del sistema **TravelEasy**. Attraverso questo messaggio, l'attore fornisce il parametro identificativo dell'account (**a**), ricevendo dal sistema l'accesso ai propri dettagli anagrafici e di registrazione tramite la restituzione delle **informazioni** associate.

Operazione	recuperoDati(a: Account)
Riferimenti	UC17: Visualizza dati cliente
Precondizioni	Il cliente deve essere autenticato nel sistema
Postcondizioni	Il sistema ha mostrato i dati in un’apposita finestra

Progettazione

Caso d’uso UC3 - Prenotazione Pacchetto Vacanza, diagrammi di interazione

validaViaggiatore



L'operazione ***validaViaggiatore*** rappresenta il primo step logico della realizzazione del caso d'uso UC3. Questa fase è cruciale per garantire che ogni partecipante aggiunto alla prenotazione possieda requisiti anagrafici e documentali validi

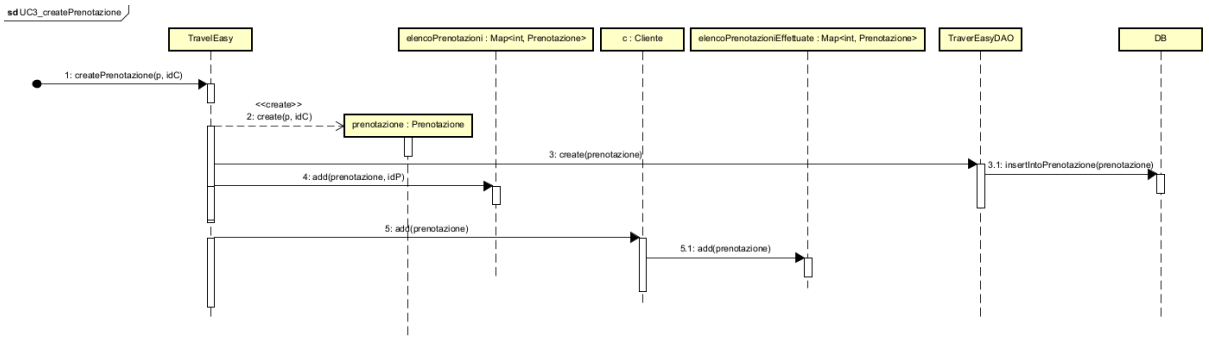
Flusso delle Operazioni e Logica Interna

1. **Ricezione dei Dati**: Il controller ***TravelEasy*** riceve la chiamata di sistema contenente il set completo di informazioni del viaggiatore: nome, cognome, data di nascita e i dettagli del documento (tipo e codice).
2. **Logica di Validazione**: Ricevuti i parametri, il controller esegue un messaggio riflessivo (***validaDati***). Questa operazione interna al controller attiva i controlli di integrità, quali:
    - La verifica della completezza dei campi obbligatori.
    - Il controllo della validità del formato dei dati inseriti.

Elementi di Design

- **Controllo Preventivo**: Seguendo il flusso stabilito nel Diagramma di Sequenza di Sistema (SSD), questa operazione viene eseguita all'interno di un ciclo iterativo per ogni passeggero. Validare i dati singolarmente e preventivamente permette di bloccare il processo di prenotazione in caso di errori formali, evitando la creazione di pratiche incomplete o errate.
- **Pattern Controller**: Il controller ***TravelEasy*** agisce qui come primo filtro di sistema, centralizzando la logica di validazione superficiale prima di delegare la creazione fisica degli oggetti viaggiatore nelle fasi successive.

createPrenotazione



L'operazione ***createPrenotazione(p, idC)*** rappresenta il momento in cui il sistema concretizza la richiesta del cliente.

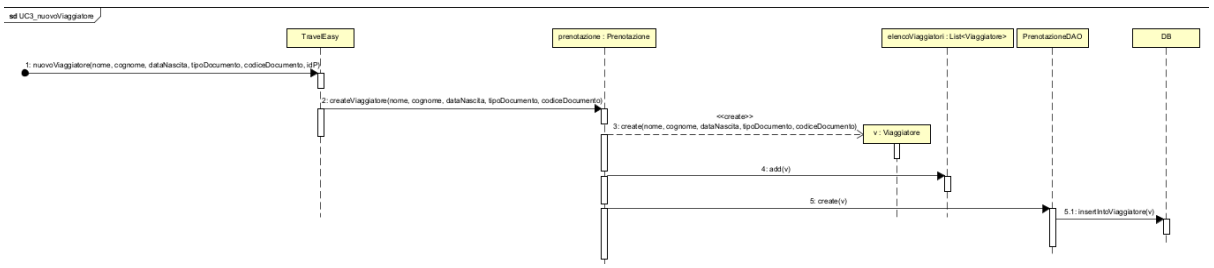
Flusso delle Operazioni e Logica Interna

1. **Istanziamento dell'Oggetto**: Il controller ***TravelEasy*** invoca il costruttore della classe ***Prenotazione***, creando l'oggetto ***prenotazione***. In questa fase vengono passati come parametri il riferimento al pacchetto selezionato (**p**) e l'identificativo del cliente (**idC**).
2. **Memorizzazione nel Database:** questo step prevede l'inserimento nel Database dell'oggetto *Prenotazione*.
3. **Registrazione Globale**: Una volta creato l'oggetto, il controller provvede immediatamente a inserirlo nella collezione globale ***elencoPrenotazioni***, che è implementata come una ***Map<int, Prenotazione>***. L'inserimento avviene tramite il messaggio ***add(prenotazione, idP)***.
4. **Aggiornamento Storico Cliente (Pattern Information Expert)**: Per garantire la navigabilità dei dati lato utente, il controller delega al cliente ***(c: Cliente***) l'aggiornamento del proprio profilo. L'oggetto *Cliente* riceve il riferimento alla nuova prenotazione e lo aggiunge alla propria mappa personale ***elencoPrenotazioniEffettuate***.

Elementi di Design

- **Pattern Creator**: Il controller ha la responsabilità di creare la *Prenotazione* in quanto possiede le informazioni necessarie (pacchetto e cliente) e aggrega l'elenco globale delle prenotazioni.
- **Persistenza e Tracciabilità**: L'uso simultaneo di una mappa globale e di una mappa specifica per il cliente garantisce che la prenotazione sia rintracciabile sia per finalità amministrative (**UC11**) che per la consultazione personale del cliente (**UC10**).
- **Persistenza dei dati**: L'utilizzo della classe DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.

nuovoViaggiatore



L'operazione ***nuovoViaggiatore*** completa il legame tra le anagrafiche dei partecipanti e la pratica di viaggio specifica. In questa fase, il sistema passa dalla semplice validazione dei dati alla creazione fisica degli oggetti nel dominio.

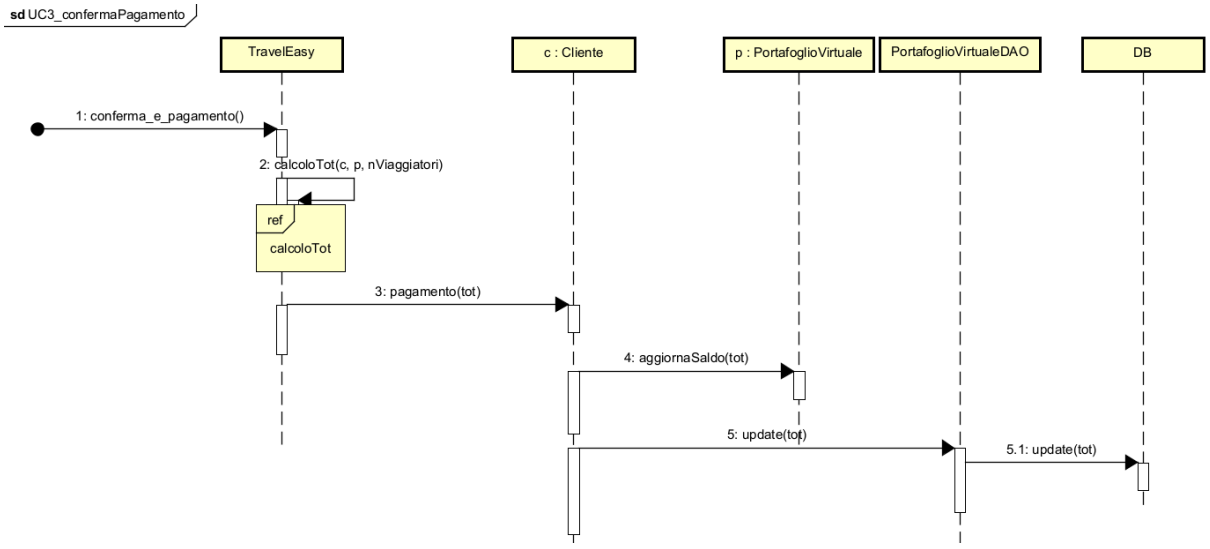
Flusso delle Operazioni e Logica Interna

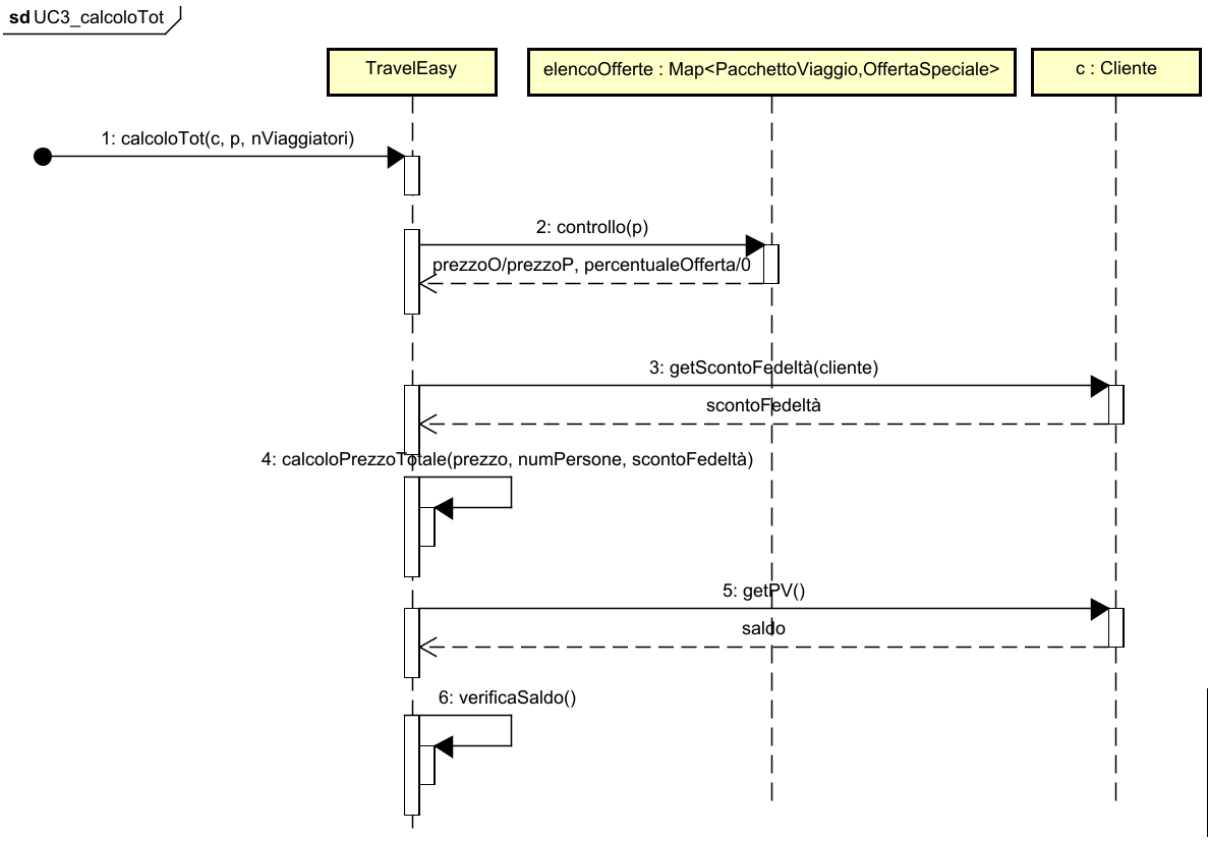
1. **Ricezione del Messaggio**: Il controller ***TravelEasy*** riceve la richiesta contenente i dati del partecipante.
2. **Delega alla Prenotazione**: Invece di creare direttamente il viaggiatore, il controller delega la responsabilità all'oggetto ***prenotazione: Prenotazione*** (identificato tramite l'idP) invocando il metodo ***createViaggiatore(...)***. Questa scelta rispetta il principio di coesione, poiché la prenotazione è l'aggregato che deve gestire i propri componenti.
3. **Istanziamento dell'Oggetto Viaggiatore**: L'oggetto *prenotazione* agisce come **Creator**, invocando il costruttore della classe ***Viaggiatore*** per creare l'istanza ***v*** con i dati anagrafici e i riferimenti ai documenti.
4. **Archiviazione nella Collezione**: Una volta creato, il viaggiatore viene aggiunto alla lista ***elencoViaggiatori : List<Viaggiatore>*** interna alla prenotazione tramite il messaggio ***add(v)***.
5. **Memorizzazione nel Database:** l'ultimo step prevede l'inserimento nel Database dell'oggetto *Viaggiatore*.

Elementi di Design

- **Pattern Creator:** La classe *Prenotazione* assume il ruolo di creatore dei viaggiatori.
- **Incapsulamento**: Il controller non ha bisogno di conoscere come i viaggiatori siano memorizzati internamente alla prenotazione; si limita a passare le informazioni necessarie all'oggetto esperto.
- **Gestione Iterativa**: Coerentemente con il Diagramma di Sequenza di Sistema, questa operazione viene eseguita ciclicamente per ogni partecipante, permettendo alla prenotazione di popolare gradualmente il proprio elenco di viaggiatori.
- **Persistenza dei dati**: L'utilizzo della classe DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.

conferma_e_pagamento





L'operazione ***conferma_e_pagamento()*** rappresenta la fase cruciale di consolidamento economico del caso d'uso. In questo step, il sistema integra le logiche di business relative a sconti, offerte e portafoglio virtuale per finalizzare la transazione.

Flusso delle Operazioni e Logica Interna

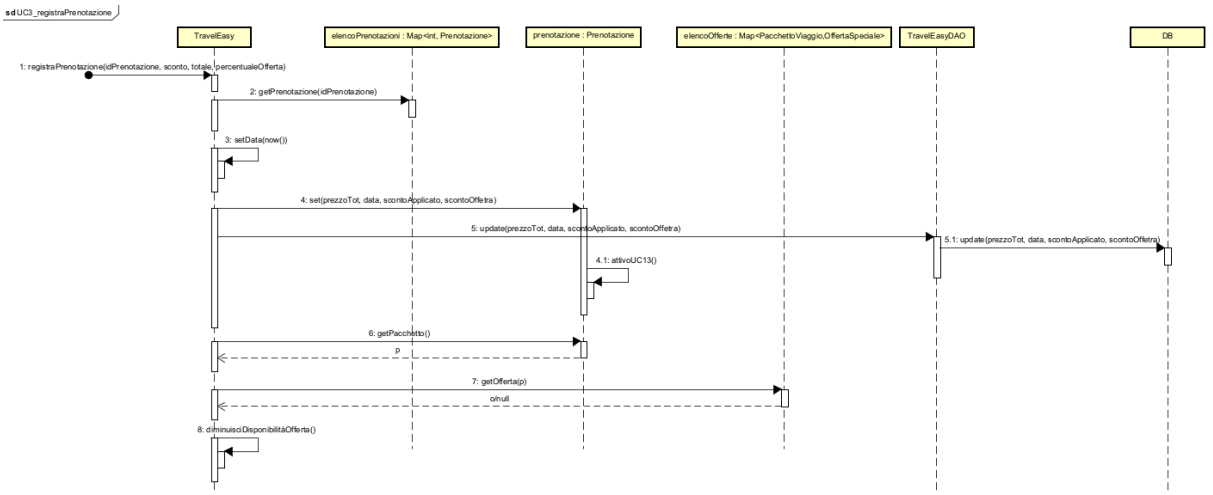
Il controller ***TravelEasy*** orchestra l'operazione attraverso una serie di deleghe mirate:

1. **Consolidamento del Prezzo (Internal Logic)**: Il controller invoca internamente il metodo ***calcoloTot(c, p, nViaggiatori)***, che attiva un flusso di recupero dati complesso:
    - **Controllo Offerte**: Viene interrogata la mappa ***elencoOfferte*** per verificare se il pacchetto ***p*** è associato a una promozione attiva, recuperando l'eventuale percentuale di sconto.
    - **Sconto Fedeltà (Pattern Information Expert)**: Il controller interroga l'oggetto ***c: Cliente*** tramite ***getScontoFedeltà()*** per ottenere eventuali riduzioni accumulate nel tempo.
    - **Verifica Copertura**: Prima di procedere, il sistema recupera il saldo dal ***PortafoglioVirtuale*** del cliente ed esegue un messaggio riflessivo ***verificaSaldo()*** per garantire che i fondi siano sufficienti a coprire il prezzo totale calcolato.
2. **Esecuzione del Pagamento**: Una volta validato il totale, il controller invia il messaggio ***pagamento(tot)*** all'oggetto ***Cliente***.
3. **Aggiornamento Saldo**: Seguendo la catena di responsabilità, il Cliente delega l'effettivo decremento monetario al proprio ***pv: PortafoglioVirtuale*** tramite il metodo ***aggiornaSaldo(tot)***.
4. **Memorizzazione nel Database**: L’ultimo step prevede l’aggiornamento nel Database dell’oggetto *PortafoglioVirtuale*.

Elementi di Design

- **Information Expert**: La responsabilità del calcolo non è centralizzata solo nel controller; il sistema interroga gli esperti delle informazioni (la mappa delle offerte per gli sconti commerciali e il cliente per quelli di fedeltà) per comporre il prezzo finale.
- **Atomicità e Sicurezza**: L'integrazione di ***verificaSaldo()*** all'interno del flusso di conferma assicura che nessuna transazione venga avviata se non è garantita la copertura economica, proteggendo l'integrità finanziaria dell'agenzia.
- **Persistenza dei dati**: L'utilizzo della classe DAO è volto a delegare le operazioni di persistenza sul Database a delle classi apposite per aumentare la coerenza e diminuire l'accoppiamento.

registraPrenotazione



L'operazione ***registraPrenotazione*** rappresenta l'atto finale del caso d'uso, in cui il sistema consolida in modo permanente i dati economici e temporali della transazione, aggiornando contemporaneamente lo stato del catalogo offerte.

Flusso delle Operazioni e Logica Interna

Il controller *TravelEasy* coordina il salvataggio definitivo attraverso i seguenti passaggi:

1. **Recupero dell'Istanza**: Il controller interroga la mappa globale ***elencoPrenotazioni*** tramite ***getPrenotazione(idPrenotazione)*** per ottenere il riferimento all'oggetto creato nelle fasi precedenti.
2. **Timestamp della Pratica**: Viene invocato il metodo riflessivo ***setData(now())*** per marcare temporalmente il momento esatto della conferma.
3. **Persistenza dei Dati Economici**: Il controller invoca il metodo ***set(...)*** sull'oggetto ***prenotazione***, passando il prezzo totale finale, la data e le percentuali di sconto (sia fedeltà che offerta) applicate.
4. **Memorizzazione nel Database**: questo step prevede l'aggiornamento nel Database dell'oggetto *Prenotazione*.
5. **Attivazione Logiche Accessorie**: L'oggetto ***prenotazione*** attiva internamente il messaggio ***attivoUC13()***, delegando l'avvio delle procedure per l'aggiornamento dei punti fedeltà o delle ore nel profilo del cliente.

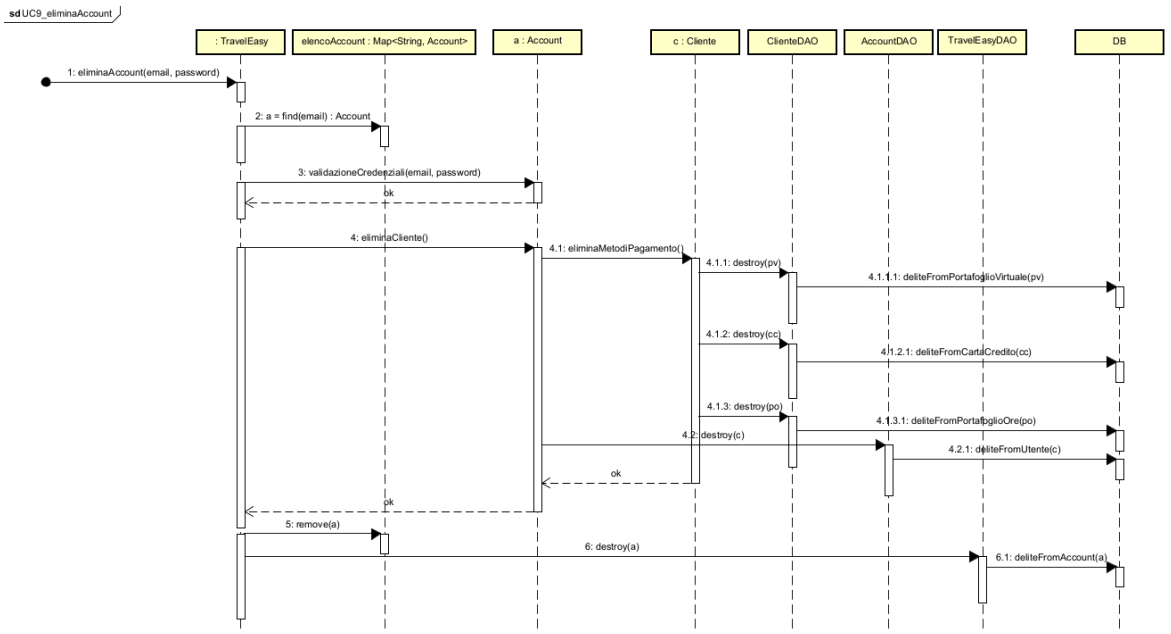
6. **Aggiornamento Disponibilità Catalogo**: Per garantire la coerenza tra vendite e offerte disponibili, il controller recupera il pacchetto associato ***(getPacchetto()***) e verifica sulla mappa ***elencoOfferte*** se era presente una promozione attiva. In caso positivo, il sistema esegue ***diminuisciDisponibilitàOfferta()***, riducendo il numero di pacchetti scontati ancora acquistabili.

Elementi di Design

- **Pattern Information Expert**: La classe *Prenotazione* è responsabile di memorizzare i propri attributi finali, mentre il controller gestisce la logica di coordinamento con le collezioni globali (*elencoOfferte*).
- **Chiusura del Ciclo**: L'aggiornamento della disponibilità dell'offerta è un esempio di come il sistema mantenga lo stato globale aggiornato in tempo reale, impattando direttamente sulla visibilità dei pacchetti per gli altri utenti.
- **Persistenza dei dati**: L'utilizzo delle classi DAO è volto a delegare le operazioni di persistenza sul Database a delle classi appositeper aumentare la coerenza e diminuire l'accoppiamento.
- Per ottimizzare la gestione del catalogo, il sistema **TravelEasy** implementa il **Pattern Observer** durante la fase finale di registrazione. Questa scelta architetturale assicura che il sistema reagisca immediatamente all'esaurimento dei pacchetti con offerte speciali.

Caso d'uso UC9 - Elimina Account, diagrammi di interazione

eliminaAccount



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 *eliminaAccount* consente di eliminare l'**account** del **cliente** autenticato e tutto ciò che è ad essi associato 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 (email e password). Prima di procedere, il controller sfrutta l'accesso alla

propria ***mappa di Account*** per individuare l'account tramite l'email (chiave della mappa). Provvede quindi a garantire la coerenza dei dati inseriti (correttezza della password per garantire autenticazione) tramite la funzione *validazioneCredenziali(...)* effettuata dall'***Account***.

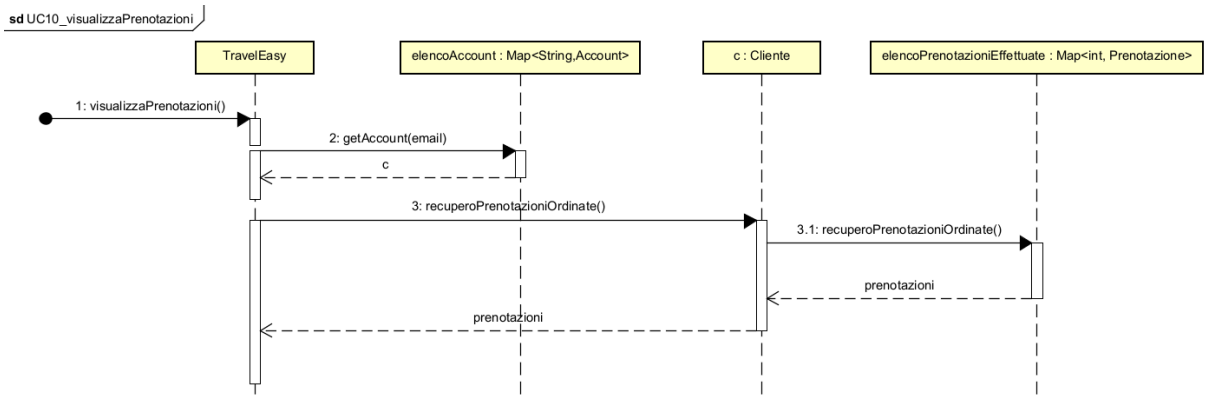
2. **Eliminazione oggetti correlati al Cliente**: Una volta validati i dati, il controller manda un messaggio all'***Account*** *eliminaCliente()*. L'***Account*** procede dunque a mandare un messaggio *eliminaMetodiPagamento()* al ***Cliente*** ad esso associato. Quest'ultimo, infine, procede a distruggere gli oggetti ***PortafoglioOre***, ***PortafoglioVirtuale*** e ***CartaCredito*** associati, eliminandoli anche dal database.
3. **Eliminazione del Cliente**: Una volta eliminati gli oggetti associati, il ***Cliente*** comunica all'***Account*** il completamento dell'operazione. L'***Account*** procede dunque a distruggere l'oggetto ***Cliente*** eliminandolo anche dal database. Procede infine a comunicare l'avvenuta eliminazione al controller.
4. **Eliminazione dell'Account**: Il controller procede a rimuovere l'***Account*** dalla propria ***mappa di Account*** e a distruggere l'oggetto eliminandolo anche dal database.

Elementi di Design

- **Controller**: La classe ***TravelEasy*** funge da punto di ingresso unico, coordinando le attività di validazione ed eliminazione.
- **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.
- **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 UC10 - Visualizzazione Prenotazioni, diagrammi di interazione

visualizzaPrenotazioni



Il diagramma di sequenza di **UC10** illustra la logica di recupero dei dati, evidenziando il passaggio dalla sessione dell'utente (identificata dall'email) alla collezione specifica delle prenotazioni archiviate nel profilo del cliente.

Flusso delle Operazioni e Logica Interna

Il processo di visualizzazione segue una catena di deleghe:

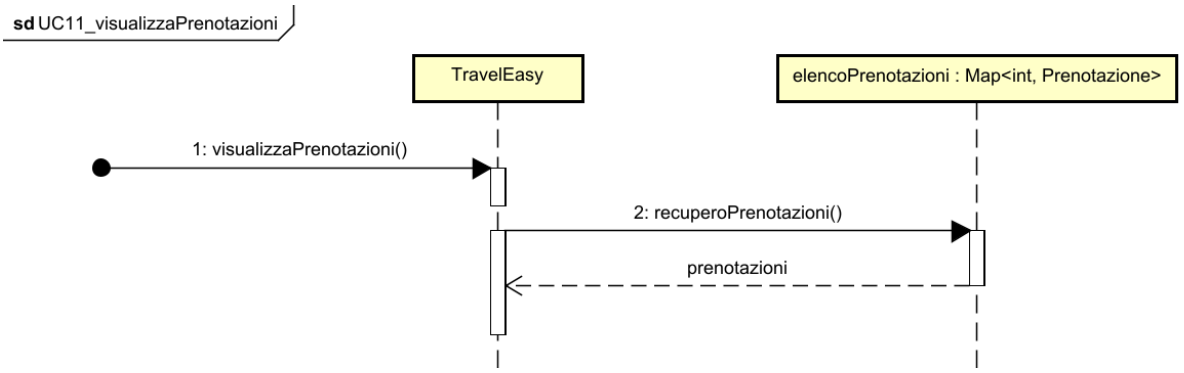
1. **Innesco dell'Operazione**: Il controller **TravelEasy** riceve il messaggio ***visualizzaPrenotazioni()***. Sebbene il messaggio di sistema sia senza parametri, il controller utilizza l'identità dell'utente loggato (email) per avviare la ricerca.
2. **Identificazione del Soggetto**: Il controller interroga l'oggetto ***elencoAccount*** (una *Map<String, Account>*) tramite il metodo ***getAccount(email)***. Questa operazione restituisce il riferimento all'oggetto ***c: Cliente*** associato a quell'account.
3. **Recupero delle Prenotazioni (Pattern Information Expert)**: Una volta ottenuto il riferimento al cliente, il controller invoca il metodo ***recuperoPrenotazioniOrdinate()*** sull'oggetto ***c: Cliente***. Con questa scelta di design è il Cliente (Expert) ad avere la responsabilità di avviare il recupero dei propri viaggi..
4. **Accesso alla Collezione**: L'oggetto ***Cliente*** delega la richiesta alla propria mappa interna ***elencoPrenotazioniEffettuate*** (una *Map<int, Prenotazione>*) invocando ***recuperoPrenotazioniOrdinate()***. La mappa restituisce l'elenco delle ***prenotazioni***, che risale la catena di chiamate fino a essere visualizzato dal controller.

Elementi di Design

- **Information Expert**: La responsabilità di fornire l'elenco non è del controller, ma della classe **Cliente**, che nel modello di dominio possiede la relazione diretta con le prenotazioni.
- **Basso Accoppiamento**: Il controller ***TravelEasy*** non accede mai direttamente ai record delle prenotazioni; interagisce esclusivamente con l'interfaccia dell'account e del cliente, mantenendo il sistema flessibile a cambiamenti nella struttura di archiviazione.
- **Efficienza del Recupero**: L'adozione di strutture dati di tipo ***Map*** sia per gli account che per le prenotazioni effettuate garantisce tempi di risposta costanti e immediati, indipendentemente dalla quantità di dati gestiti dal sistema **TravelEasy**.

Caso d'uso UC11 - Visualizza Prenotazioni dell'Agenzia, diagrammi di interazione

VisualizzaPrenotazioni



Il diagramma di sequenza dell'**UC11** illustra la logica di back-end dedicata all'amministrazione, che permette all'operatore di ottenere una visione completa, centralizzata e indicizzata di tutte le prenotazioni registrate nel sistema **TravelEasy**.

Flusso delle Operazioni e Logica Interna

L'interazione è caratterizzata da un accesso diretto e performante alla collezione globale, semplificato rispetto alla vista utente per favorire la velocità di consultazione gestionale:

1. **Chiamata di Sistema**: Il controller **TravelEasy** riceve il messaggio ***visualizzaPrenotazioni()*** inviato dall'interfaccia dell'Operatore.
2. **Accesso alla Collezione Globale**:A differenza del caso d'uso lato cliente (**UC10**), il controller non deve filtrare i dati per un singolo account. Esso interagisce direttamente con l'oggetto ***elencoPrenotazioni***, che è implementato come una ***Map<int, Prenotazione>***.
3. **Recupero Dati**: Viene invocato il metodo ***recuperoPrenotazioni()*** sulla mappa globale. Questa operazione estrae i riferimenti a tutte le istanze della classe **Prenotazione** presenti nel sistema.
4. **Restituzione dei Risultati**: L'elenco completo viene restituito al controller, che a sua volta lo inoltra all'interfaccia dell'operatore per la visualizzazione.

Elementi di Design

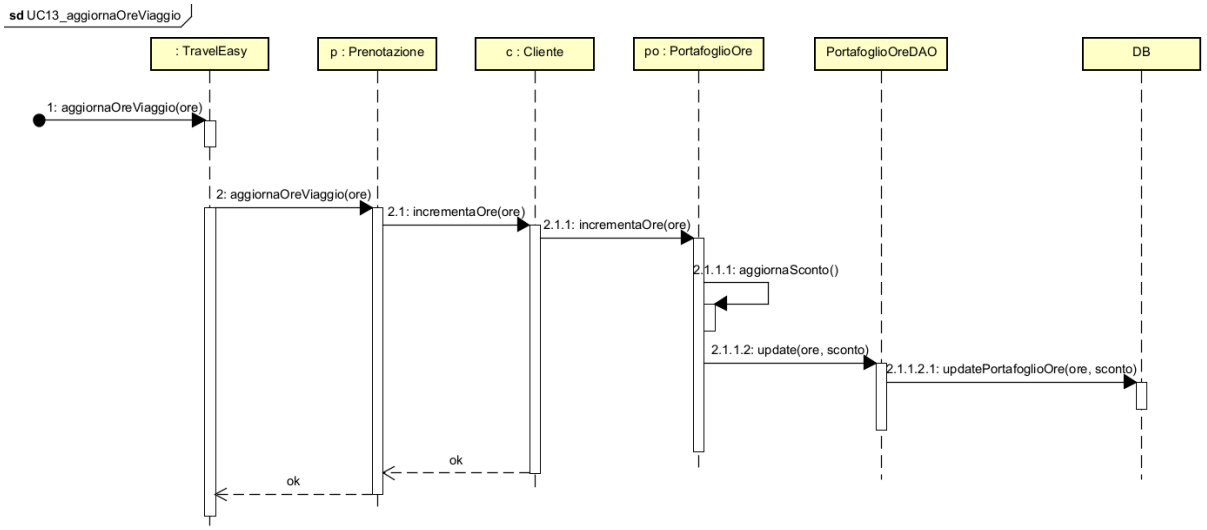
- **Pattern Controller**: La classe **TravelEasy** funge da mediatore puro, gestendo la richiesta dell'attore e delegando il reperimento dei dati alla struttura dati competente.
- **Gestione Centralizzata e Scalabile**: L'adozione di una ***Map<int, Prenotazione>*** riflette la scelta di mantenere un'unica collezione globale indicizzata, rendendo le operazioni di monitoraggio, ricerca per ID e audit dell'agenzia estremamente efficienti.

Differenziazione dei Flussi (UC10 vs UC11):

- **UC10 (Cliente)**: L'accesso è mediato dall'oggetto ***Cliente*** e dalla sua mappa privata per garantire la privacy e mostrare solo i dati pertinenti all'utente loggato.
- **UC11 (Operatore)**: L'accesso è diretto alla mappa globale per finalità amministrative, garantendo una visione d'insieme su tutto il business dell'agenzia attraverso un'unica operazione.

Caso d’uso UC13 - Accumulo Ore, diagrammi di interazione

aggiornaOreViaggio



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 *aggiornaOreViaggio* consente di aggiornare le ore nel **portafoglio ore** del **cliente** autenticato attraverso i seguenti passi logici:

Flusso delle Operazioni e Logica Interna

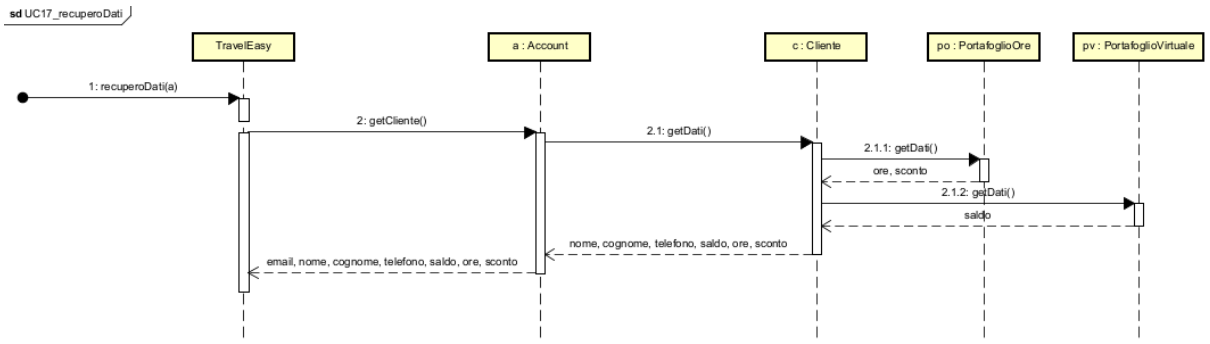
1. **Recupero dell'oggetto**: Il controller ***Travel Easy*** riceve il messaggio iniziale contenente il parametro ore. Il controller procede mandando un messaggio *aggiornaOreViaggio* alla ***Prenotazione*** appena creata dal caso d'uso **UC3**. La ***Prenotazione*** manda dunque un messaggio *incrementaOre(...)* al proprio ***Cliente***.
2. **Aggiornamento ore**: Il ***Cliente*** manda un messaggio al ***PortafoglioOre*** *incrementaOre(...)*. Il ***PortafoglioOre*** procede dunque ad aggiornare il proprio monte ore aggiungendo le ore ottenute tramite la nuova ***Prenotazione*** effettuata.
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.

Caso d'uso UC17 - Visualizza Dati Cliente, diagramma di interazione

recuperoDati



La fase della realizzazione dell'**UC17** descrive come il controller avvia l'estrazione delle informazioni del profilo.

Flusso delle Operazioni e Logica Interna

L'operazione si sviluppa attraverso un meccanismo di deleghe a cascata che garantisce l'incapsulamento dei dati:

1. **Chiamata di Sistema**: Il controller ***TravelEasy*** riceve il messaggio ***recuperoDati(a)***, dove il parametro ***a*** rappresenta l'istanza della classe ***Account*** attualmente in sessione.
2. **Identificazione del Soggetto**: Il controller non interroga direttamente l'anagrafica, ma invoca il metodo ***getClient()*** sull'oggetto ***a : Account***. Questa operazione serve a risalire all'istanza specifica di **Cliente** associata a quelle credenziali.

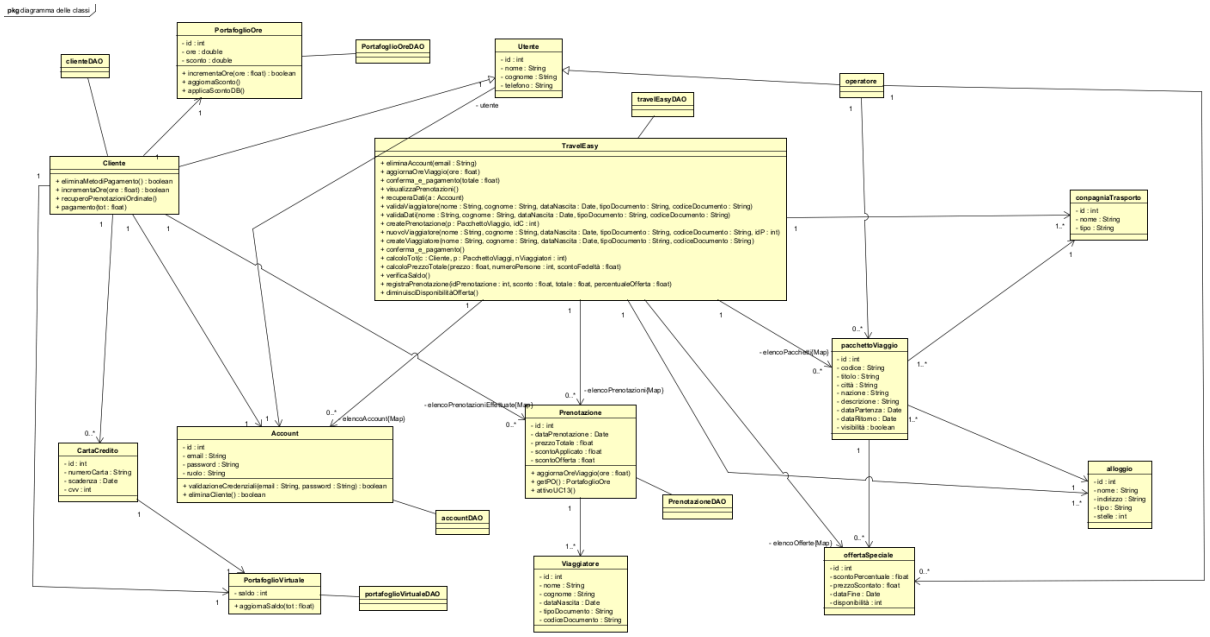
3. **Innesco del Recupero (getData)**: Una volta ottenuto il riferimento al cliente, l'oggetto **Account** invoca il metodo ***getDati()*** sull'oggetto ***c : Cliente***. Questo messaggio serve per la raccolta di tutte le informazioni di dettaglio che verranno successivamente aggregate.

Elementi di Design

- **Pattern Information Expert**: La responsabilità di avviare il recupero è affidata ad **Account** poiché è l'oggetto che possiede il riferimento diretto all'entità Cliente.
- **Sicurezza e Privacy**: Il fatto che il controller debba passare attraverso l'oggetto Account assicura che il sistema recuperi solo i dati dell'utente effettivamente loggato, prevenendo accessi non autorizzati ad altri profili.
- **Separazione delle Responsabilità**: Il controller rimane "leggero" (Pure Fabrication), limitandosi a coordinare la richiesta iniziale senza conoscere la struttura interna dei dati del cliente.

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*** funge da controller principale e da "punto di ingresso" per le operazioni di business. Essa aggrega le collezioni fondamentali del sistema attraverso strutture dati di tipo **Map**, che permettono un accesso efficiente ai dati:

- ***elencoAccount***: Mappa gli account tramite email.
- ***elencoPacchetti***: Gestisce il catalogo dei viaggi.
- ***elencoPrenotazioni***: Centralizza tutte le transazioni attive.
- ***elencoOfferte***: Associa i pacchetti alle eventuali promozioni attive.

Il sistema adotta una struttura gerarchica basata sulla classe ***Utente***, che contiene attributi come `nome`, *cognome* e *telefono*:

- ***Cliente***: Specializzazione dell'utente che può possedere più ***CartaCredito*** e gestire i propri fondi tramite il ***PortafoglioVirtuale***.
- ***Operatore***: Responsabile della gestione del catalogo e dei pacchetti.
- ***Account***: Entità separata che gestisce le credenziali (*email*, *password*) e il ruolo, garantendo la separazione tra dati anagrafici e dati di accesso.

Il cuore del sistema è rappresentato dalla relazione tra la ***Prenotazione*** e i suoi componenti:

- ***Prenotazione***: Memorizza i dettagli della transazione, inclusi la data, il prezzo totale e gli sconti applicati (fedeltà o offerta).
- ***Viaggiatore***: Ogni prenotazione aggrega uno o più viaggiatori (relazione 1..*), di cui vengono memorizzati i dati del documento e le informazioni anagrafiche. La classe include il metodo *validaDati()* per il controllo di integrità.

L'offerta turistica è modellata attraverso la classe ***pacchettoViaggio***, che aggrega informazioni su ***alloggio*** e ***compagniaTrasporto***.

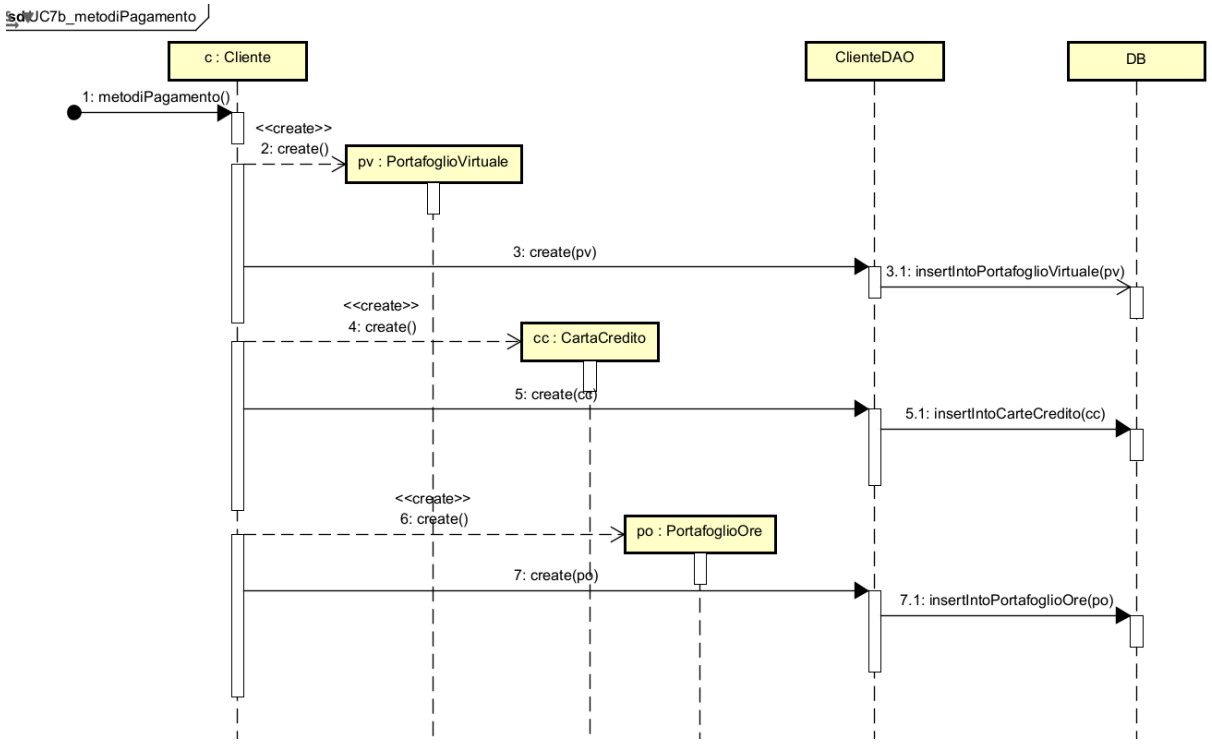
- ***offertaSpeciale***: Permette di applicare logiche di sconto percentuale a un pacchetto, gestendo la disponibilità residua e le date di validità.

Per incentivare la fidelizzazione, ogni cliente è associato a un ***PortafoglioOre***:

- Questa classe gestisce il saldo delle ore accumulate e calcola lo sconto applicabile tramite i metodi *aggiornaSconto()* e *applicaScontoDB()*.
- Il ***PortafoglioVirtuale*** gestisce invece il saldo monetario effettivo, permettendo pagamenti immediati all'interno della piattaforma.

Revisione

Dall'analisi dei casi d'uso relativi alla presente iterazione è emersa la necessità di apportare una modifica al caso d'uso UC7: Creazione Account, prevedendo l'istanziazione della classe PortafoglioOre e la sua associazione al Cliente che ne risulta proprietario. L'unica modifica apportata riguarda il diagramma di interazione dell'operazione *metodiPagamento*, che è stato aggiornato prevedendo l'istanziazione di un oggetto PortafoglioOre.



Testing

UC3 - Prenotazione pacchetto vacanza

Per UC3, i test verificano la creazione bozza, l'inserimento viaggiatori, la registrazione finale e gli effetti collaterali su disponibilità offerta, data e portafoglio ore.

Nel test ***registrazionePrenotazione_conDatiValidi_inseriscePrenotazioneEViaggiatori***, il metodo di testing usa il supporto creaBozzaConViaggiatori(...), che chiama i metodi di progetto TravelEasy.createPrenotazione(...) e TravelEasy.createViaggiatore(...) con input:

- cliente: cliente@example.com;
- pacchetto: id 2;
- viaggiatori: Paolo Neri e Marta Blu con dati documento completi.

Dopo la bozza, viene chiamato TravelEasy.registrazionePrenotazione(newId, 3.0f, 930.0f, 0.0f).

- Risultato: true, incremento di 1 prenotazione, incremento di 2 viaggiatori, aggiornamento sia della lista prenotazioni cliente sia della mappa prenotazioni generale operatore.

Nel test ***registrazionePrenotazione_conScontoFedelta_aggiornaPortafoglioOre***, con un solo viaggiatore e pacchetto con OreViaggio = 1.5, viene chiamato registrazionePrenotazione(newId, 3.0f, 465.0f, 0.0f).
Il metodo di progetto invoca internamente la logica prenotazione/portafoglio ore (Prenotazione.aggiornaOreViaggio(...) -> Cliente.aggiornaOreViaggio(...) -> PortafoglioOre.incrementaOre(...)).

- Risultato verificato su DB: Ore = 6.5, Sconto = 3.0.

Nel test ***registrazionePrenotazione_conOffertaAttiva_diminuisceDisponibilita***, la registrazione usa pacchetto con offerta attiva e input offertaApplicata = 25.0f (registrazionePrenotazione(newId, 0.0f, 900.0f, 25.0f)).

- Risultato: disponibilità in OffertaSpeciale diminuita di 1.

Nel test ***registrazionePrenotazione_impostaDataPrenotazioneOdierna***, dopo registrazione valida (registrazionePrenotazione(newId, 0.0f, 480.0f, 0.0f)), viene verificata la chiamata/effetto su Prenotazione.setDataPrenotazione(...) interno: data impostata alla data corrente formattata dd-MM-uuuu.

Nel test r***egistrazionePrenotazione_nonVaABuonFine_bozzaAnnullabileERimossa***, viene forzito un fallimento impostando bozza.setPacchetto(null) prima della registrazione.
registrazionePrenotazione(...) restituisce false; poi viene chiamato TravelEasy.annullaPrenotazioneBozza(id).

- Risultato: annullamento true, prenotazione rimossa dalla mappa e rollback logico dei conteggi (Prenotazioni e Viaggiatore tornano ai valori iniziali).

UC9 - Elimina account

Per UC9 i test validano eliminazione corretta, blocco in caso credenziali errate e rimozione dei dati collegati.

Nel test ***eliminaAccount_conCredenzialiValide_rimuoveAccountEUtente*** viene chiamato TravelEasy.eliminaAccount(conn, "cliente@example.com", "pwd123").

- Risultato: true, tabella Account decrementata di 1, tabella Utenti decrementata di 1, e getAccountToHomeView("cliente@example.com") restituisce null.

Nel test ***eliminaAccount_conPasswordErrata_nonRimuoveNulla***, input password "wrong".

- Risultato: false, nessuna variazione su Account e Utenti.

Nel test ***eliminaAccount_rimuoveMetodiPagamentoAssociati***, dopo eliminazione valida vengono verificate query su:

    - PortafoglioVirtuale WHERE Utente=1;
    - CartaCredito WHERE Utente=1.

Risultato: entrambe a 0 record.

Nel test ***eliminaAccount_rimuovePortafoglioOreAssociato***, viene verificata anche la tabella PortafoglioOre con filtro su proprietario/utente.

- Risultato: 0 record.

UC10 - Visualizzazione prenotazioni (cliente)

Per UC10 i test sono nella classe TravelEasyVisualizzaPrenotazioniClienteTest e usano principalmente:

- TravelEasy.getAccountToHomeView(...);
- Cliente.getElencoPrenotazioniEffettuate();

metodi di prenotazione già citati per simulare nuove prenotazioni.

Nel test ***elencoPrenotazioniCliente_allAvvio_contienePrenotazioniEffettuate***, input cliente seed cliente@example.com.

- Risultato: la collezione iniziale contiene 1 prenotazione.

Nel test ***elencoPrenotazioniCliente_contienePrenotazioneConDettagliValorizzati***, viene verificato che la prima prenotazione abbia:

- codice pacchetto PKG1001;
- data prenotazione 16-02-2026;
- totale 900.0f.

Nel test ***elencoPrenotazioniCliente_dopoNuovaPrenotazione_vieneAggiornato***, dopo chiamate a createPrenotazione, createViaggiatore e registrazionePrenotazione(...), il numero prenotazioni cliente aumenta di 1.

Nel test ***elencoPrenotazioniCliente_ogniPrenotazioneHaListaViaggiatori***, viene verificato che l'elenco viaggiatori della prenotazione cliente sia caricato e di dimensione 2.

UC11 - Visualizza prenotazioni dell'agenzia

Per UC11, i test in TravelEasyVisualizzaPrenotazioniOperatoreTest verificano il punto di vista operatore usando:

- TravelEasy.getPrenotazioni();
- TravelEasy.getPrenotazioneById(...).

Nel test ***getPrenotazioni_allAvvio_restituiscePrenotazioniDaDatabase***, la mappa iniziale contiene 1 prenotazione e la data è 16-02-2026.

Nel test ***getPrenotazioni_contieneDettagliClienteEPacchetto***, viene verificata la valorizzazione oggetti correlati:

- cliente non nullo, nome Mario;
- pacchetto non nullo, codice PKG1001.

Nel test ***getPrenotazioni_contieneElencoViaggiatoriCaricato***, la prenotazione operatore contiene lista viaggiatori non nulla e dimensione 2.

Nel test ***getPrenotazioni_dopoNuovaPrenotazione_contieneNuovoElemento***, dopo nuova prenotazione registrata (create + registrazione), la mappa operatore passa da 1 a 2 elementi e la nuova prenotazione ha data odierna.

UC13 - Accumulo ore

Per UC13 i test sono in TravelEasyGestionePortafoglioOreTest e verificano la logica della classe PortafoglioOre e i suoi effetti su DB.

Nel test ***incrementaOre_superaSogliaMultiplaAggiornaSconto***, chiamata a PortafoglioOre.incrementaOre(conn, 5.0f) partendo dal seed (Ore=15, Sconto=3).
Il metodo usa internamente aggiornaSconto() e applicaScontoDB(conn).

- Risultato: true, nuovo stato Ore=0, Sconto=6, confermato anche su DB.

Nel test ***incrementaOre_sottoNuovaSogliaMantieneScontoCorrente***, chiamata incrementaOre(conn, 2.0f) con stesso seed.

- Risultato: Ore=7, Sconto=3.

Nel test ***aggiornaSconto_conMenoDiDieciOre_nonModificaSconto***, su oggetto standalone con Ore=8.5, chiamata aggiornaSconto().

- Risultato: Sconto resta 0.

Nel test ***applicaScontoDb_conValoriEspliciti_persistaCorrettaRiga***, con Ore=31, Sconto=9, chiamata applicaScontoDB(conn).

- Risultato: persistenza corretta dei valori su tabella PortafoglioOre.

Nel test ***decrementaOre_conEliminazionePrenotazione_aggiornaPortafoglioOre***, viene chiamato TravelEasy.eliminaPrenotazione(prenotazione, 50.0f) su una prenotazione con pacchetto impostato a OreViaggio=3.5.

Il metodo di progetto attiva internamente Prenotazione.levaOreViaggio(...) -> Cliente.levaOreViaggio(...) -> PortafoglioOre.decrementaOre(...).

- Risultato: esito 0 (successo), portafoglio ore aggiornato a Ore=1.5, Sconto=6, con conferma su DB.

UC17 - Visualizza dati cliente

Per UC17 i test in TravelEasyProfiloUtenteTest verificano recupero account e dati profilo.

Nel test ***getAccountToHomeView_conEmailCliente_restituisceAccountCliente***, chiamata TravelEasy.getAccountToHomeView("cliente@example.com").

- Risultato: account non nullo, ruolo Cliente, oggetto cliente valorizzato.

Nel test ***profiloCliente_contieneDatiAnagraficiAttesi***, usando account.getCliente(), vengono letti i campi:

- nome Mario,
- cognome Rossi,
- telefono 333111222.

Risultato coerente con seed DB.

Nel test ***profiloCliente_contieneSaldoPortafoglioAtteso***, viene letto cliente.getPv().getSaldo().

- Risultato: 120.0.

Nel test ***profiloCliente_contienePortafoglioOreAtteso***, vengono letti cliente.getPo().getOre() e cliente.getPo().getSconto().

- Risultato: 15.0 ore e 3.0 sconto.

Regole di dominio (Iterazione 2)

Le regole di dominio verificate in questa iterazione includono:

- aggiornamento consistente degli stati prenotazione (bozza/finale), inclusi viaggiatori e data prenotazione;
- propagazione effetti di prenotazione su disponibilità offerte e su accumulo ore;
- reversibilità tramite annullamento bozza in caso di flusso non completato;
- integrità eliminazione account con rimozione dei metodi di pagamento e portafoglio ore;
- corretto caricamento aggregato dei dati prenotazione lato cliente/operatore;
- corretta esposizione dei dati profilo cliente (anagrafica e portafogli).