InViaggio: Elaborazione - Iterazione 3

G. Messina – S. Squillaci – A. Zarbo

Introduzione

Per l'iterazione 3 sono stati scelti i seguenti requisiti su cui concentrarsi:

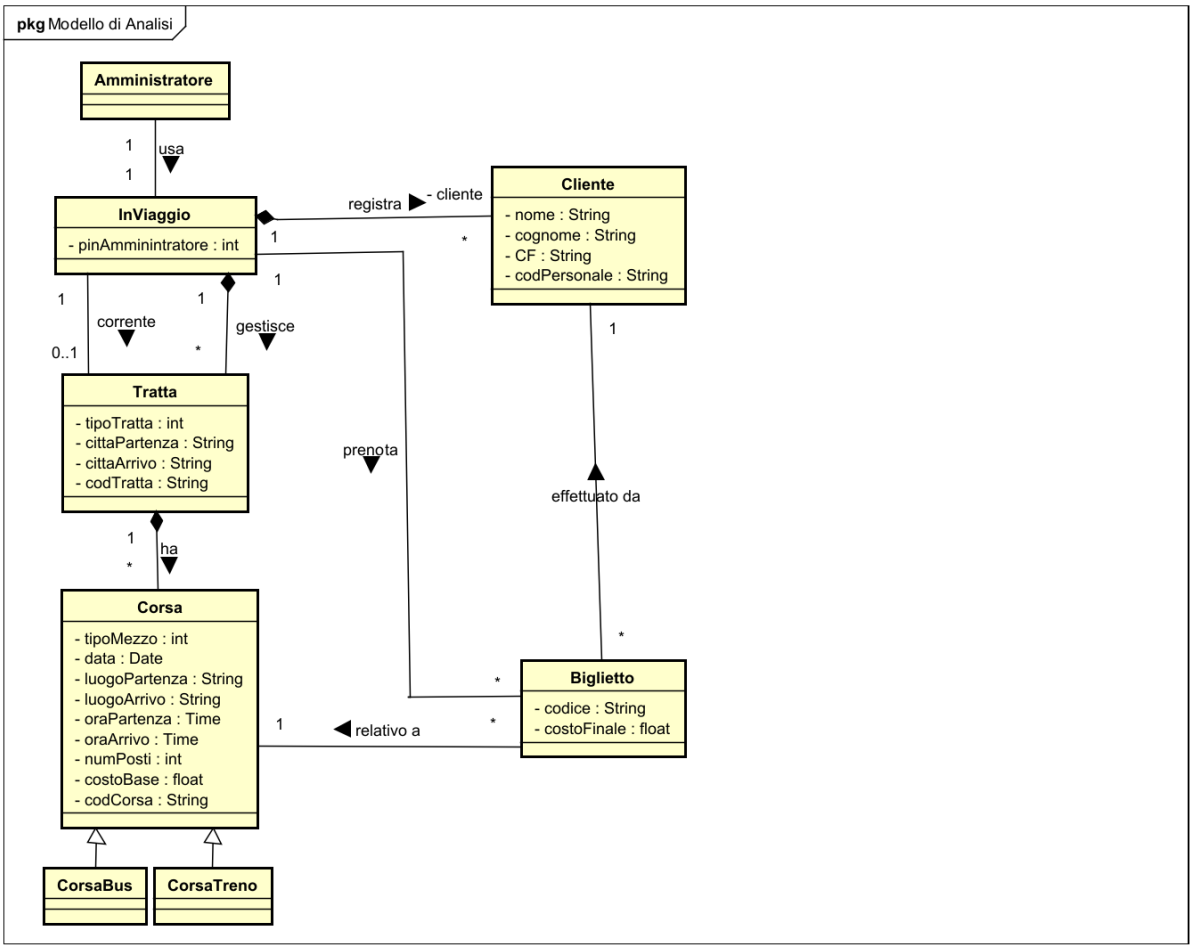
- Implementazione del caso d'uso UC1: Login Amministratore.
- Implementazione del caso d'uso UC9: Sospensione Tratta.
- Implementazione del caso d'uso UC10: Rimozione Corsa.
- Implementazione del caso d'uso di Avviamento necessario per inizializzare questa iterazione.

1. Analisi Orientata agli Oggetti

Di seguito, come nelle precedenti iterazioni, vengono riportati: Modello di Dominio, Diagrammi di Sequenza di Sistema e contratti delle operazioni

1.1. Modello di Dominio

Per l'implementazione del caso d'uso UC1 è stato necessario aggiungere l'attributo pinAmministratore alla classe "inViaggio".

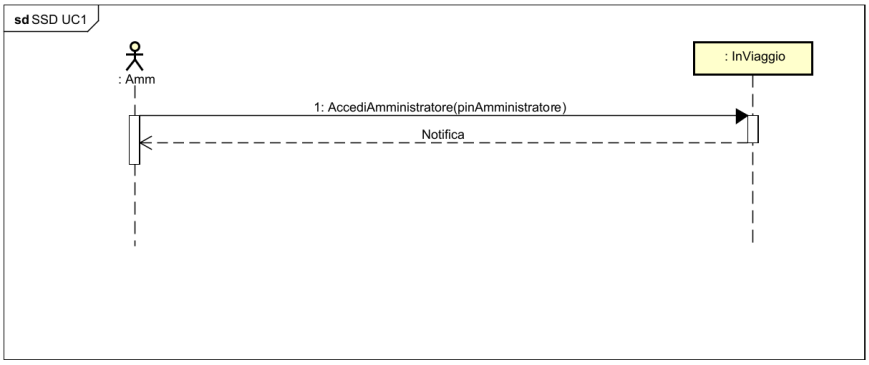


1.2. Diagrammi di Sequenza di Sistema (SSD)

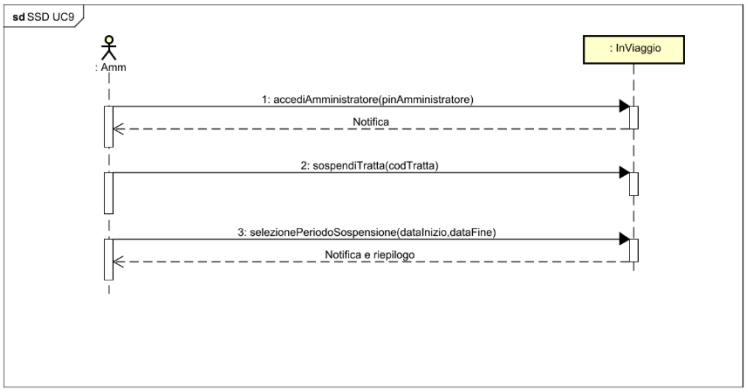
Di seguito vengono riportati i diagrammi di sequenza di Sistema relativi ai casi d'uso presi in esame in questa iterazione.

È stato modificato il nome dell'operazione prenotaBiglietto in visualizzaElencoTratte per poter essere riutilizzato oltre che nel UC6 anche nel UC10

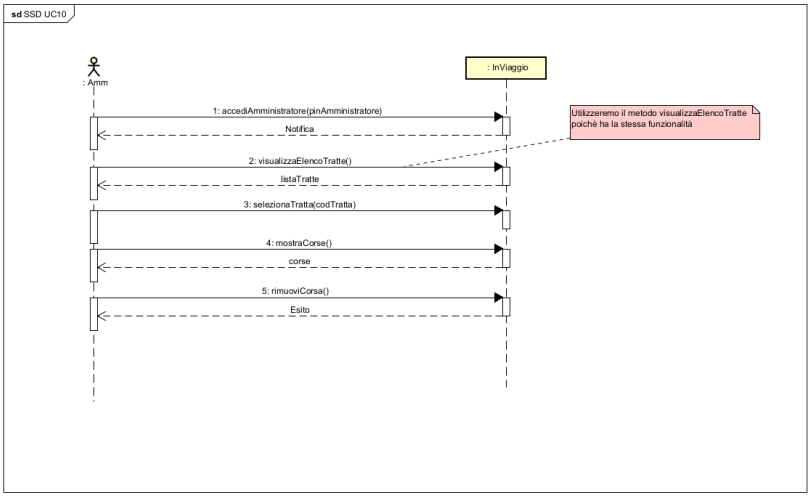
1.2.1. SSD UC1



1.2.2. SSD UC9



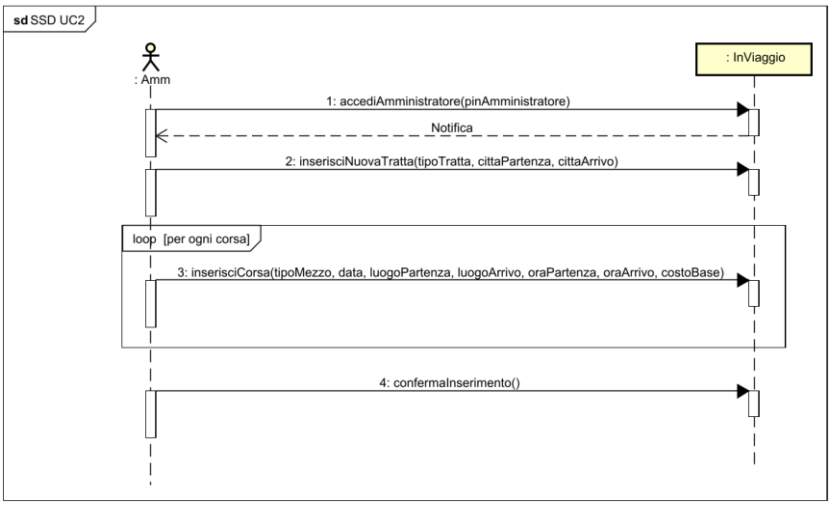
1.2.3. SSD UC10



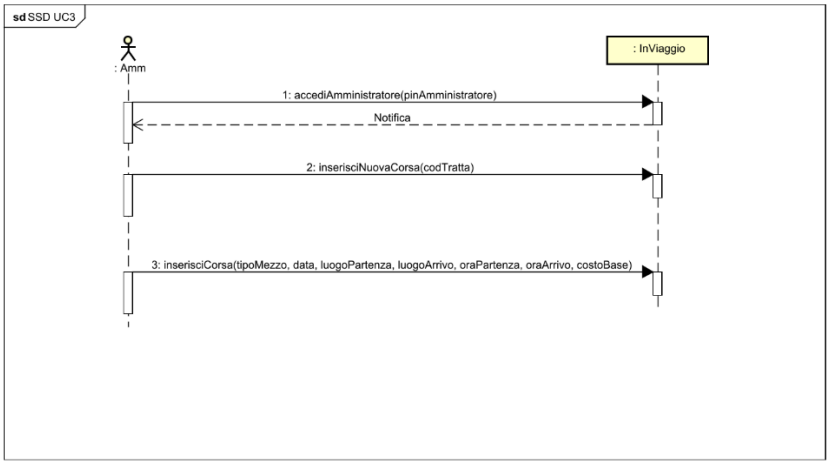
1.2.4. Aggiornamento SSD Iterazioni precedenti

In seguito all'implementazione del caso d'uso UC1 sono state apportate delle modifiche ad alcuni degli SSD realizzati nelle iterazioni precedenti, e di conseguenza sono state aggiornate le precondizioni nel modello dei casi d'uso, eliminando la condizione che l'amministratore è già loggato.

1.2.4.1 SSD UC2



1.2.4.2 SSD UC3



1.3 Contratti delle Operazioni

1.3.1 selezionaPeriodoSospensione

Operazione	selezionePeriodoSospensione(dataInizio:Date, dataFine: Date)
Riferimenti	Caso d'uso UC9: Sospensione Tratta.
Pre-condizioni	È stata recuperata l'istanza "t" di Tratta da sospendere
Post-condizioni	- Sono state recuperate le istanze di Corsa la cui data ricade nell'intervallo [dataInizio, dataFine] tramite la relazione "ha". - Sono state dissociate le istanze di Corsa recuperate dall'istanza "t" tramite l'associazione "ha".

	- Per ogni Cliente sono state recuperate le istanze di Biglietto relative alle corse, ottenute precedentemente, tramite la relazione “effettuato da”. - Le istanze di Biglietto recuperate vengono dissociate dal proprio Cliente tramite l’associazione “effettuato da”.

1.3.2 rimuoviCorsa

Operazione	rimuoviCorsa()
Riferimenti	Caso d’uso UC10: Rimozione Corsa.
Pre-condizioni	È stata recuperata l’istanza “t” di Tratta È stata recuperata l’istanza “c” di Corsa, contenuta in “t”, da rimuovere
Post-condizioni	- È stata dissociata l’istanza di Corsa da “t” tramite l’associazione “ha” - Per ogni Cliente sono state recuperate le istanze “b” di Biglietto relative alla corsa “c”, ottenuta precedentemente, tramite la relazione “effettuato da”. - Le istanze “b” di Biglietto recuperate vengono dissociate dal proprio Cliente tramite l’associazione “effettuato da”.

2. Progettazione Orientata agli oggetti

In questa fase si procede con la definizione degli oggetti software a partire dagli oggetti concettuali individuati nella fase precedente. Inoltre, si definiscono anche le loro responsabilità e le loro interazioni al fine di soddisfare i requisiti individuati nei passi precedenti. A seguire sono riportati i diagrammi di sequenza (SD) ed il diagramma delle classi relativi ai casi d’uso UC1, UC9 ed UC10.

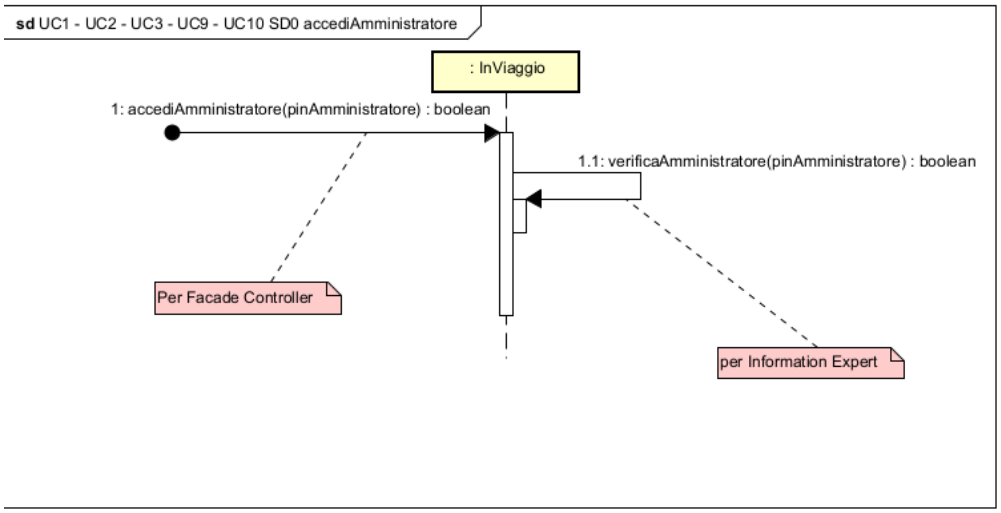
2.1 Pattern applicati

Sono stati applicati i principali pattern GRASP come Controller, Information Expert, basso accoppiamento (low Coupling) e alta coesione (High cohesion). È stato inoltre utilizzato anche il pattern GoF Observer per poter implementare il sistema di notifica agli utenti.

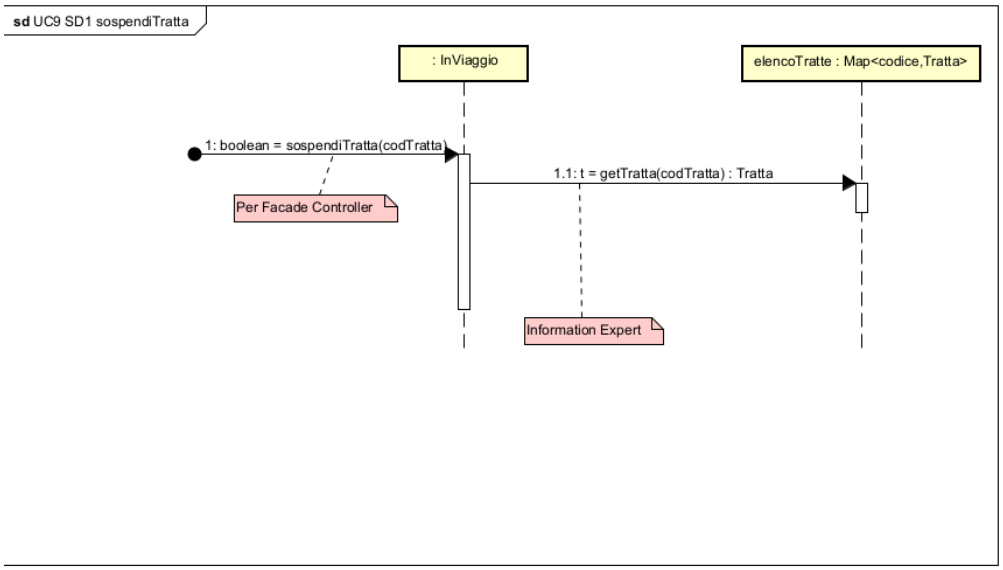
2.2 Diagrammi di Sequenza

Di seguito vengono riportati i diagrammi di sequenza relativi ai casi d’uso presi in esame:

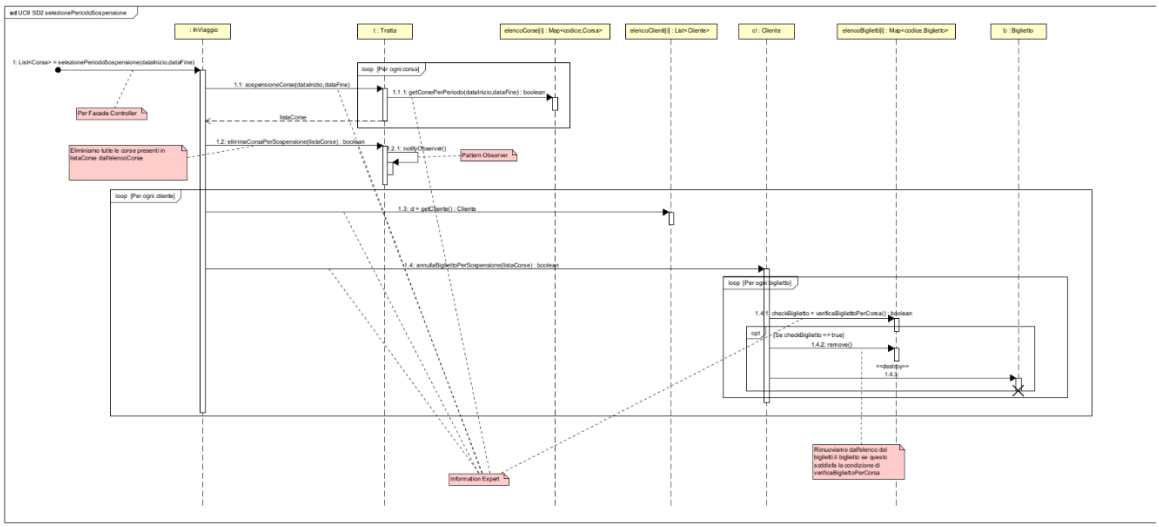
2.2.1 UC1 UC9 UC10 SD0



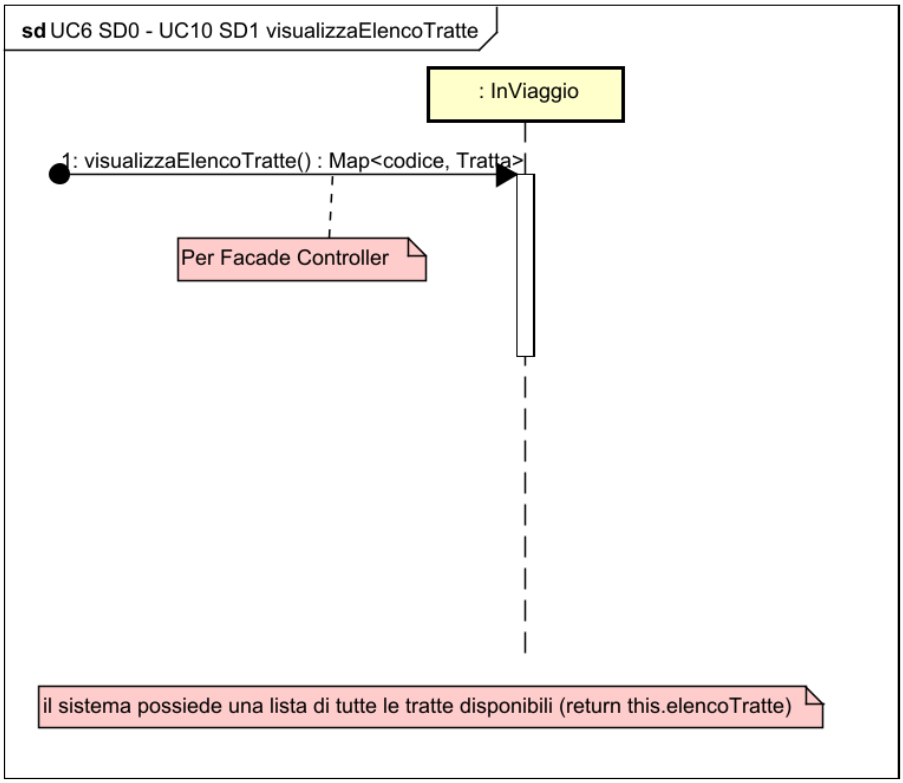
2.2.2 UC9 SD1



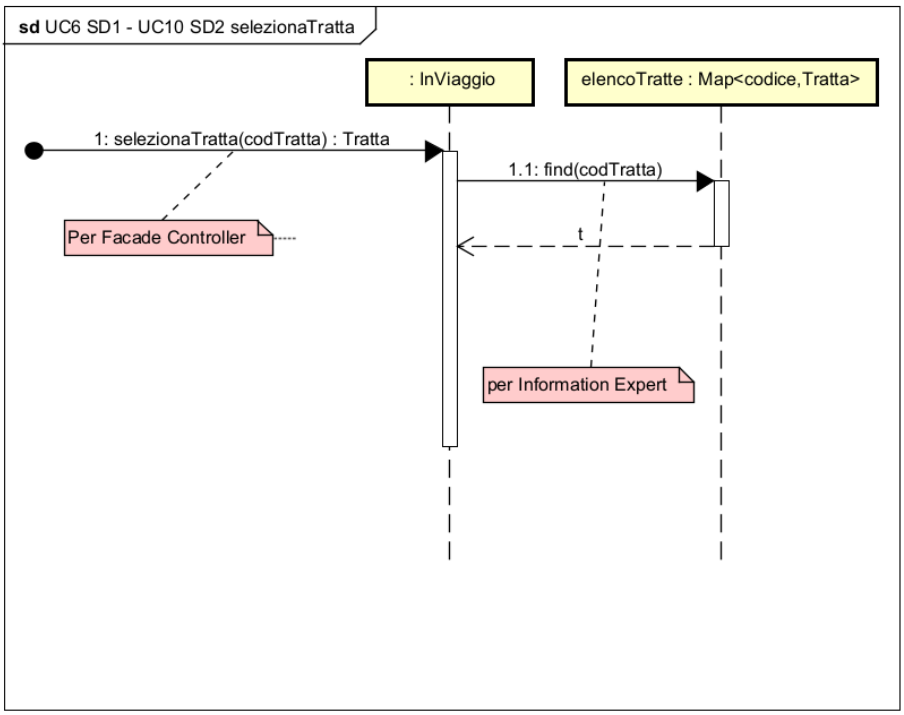
2.2.3 UC9 SD2



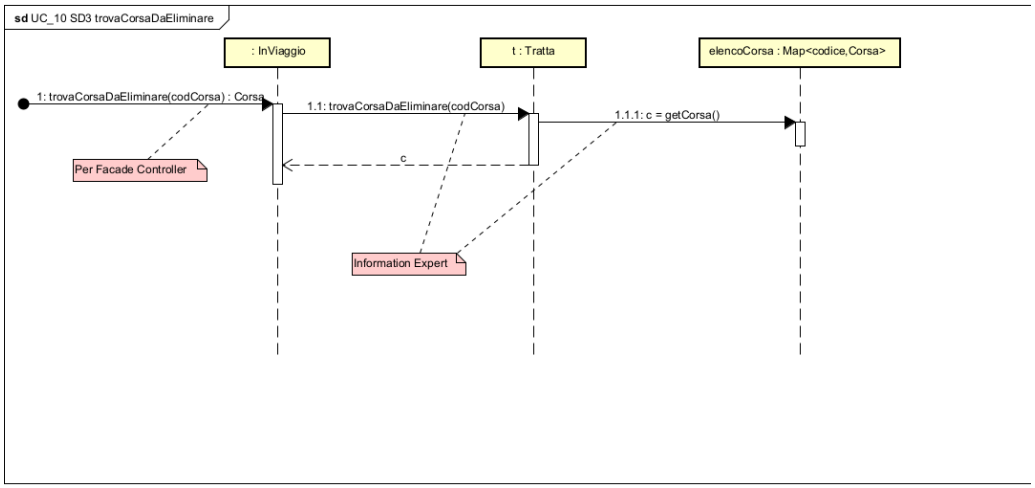
2.2.4 UC10 SD1



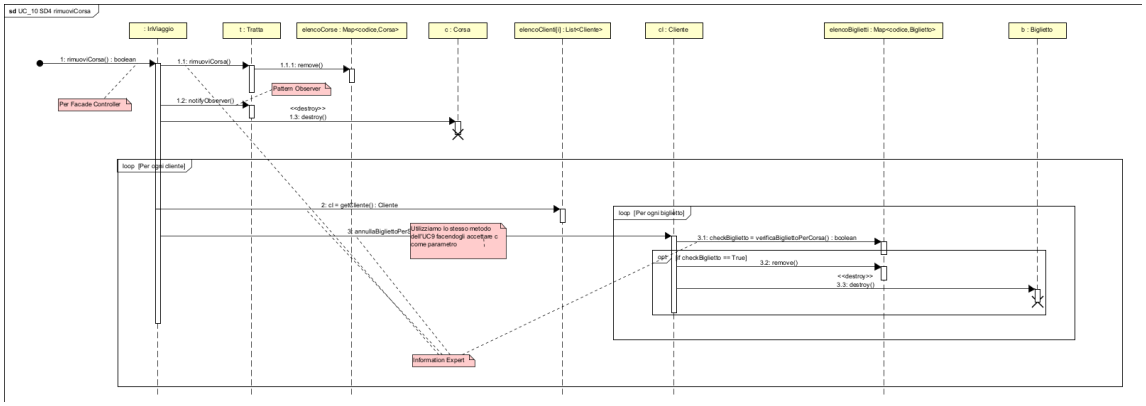
2.2.5 UC10 SD2



2.2.6 UC 10 SD3



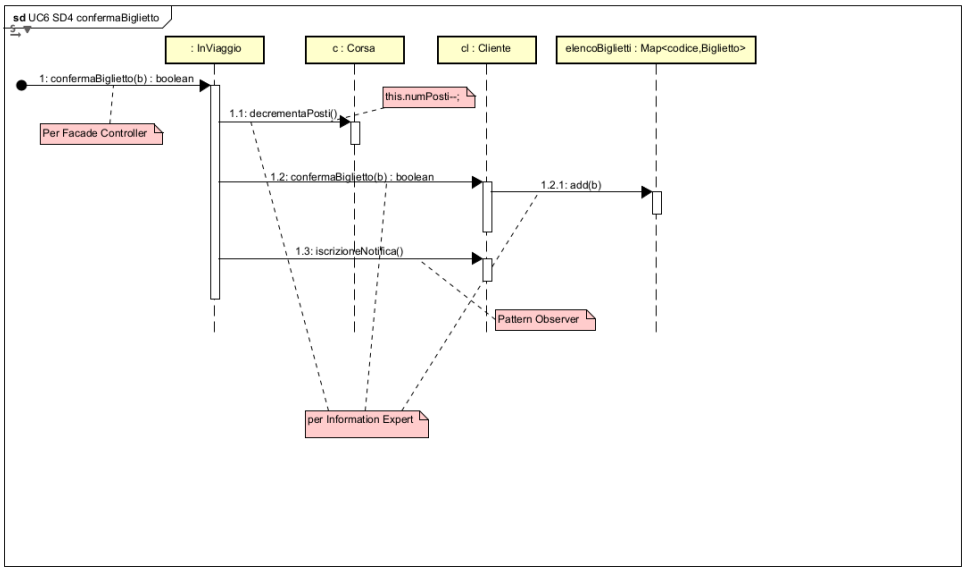
2.2.7 UC 10 SD4



2.2.8 SD iterazioni precedenti modificati

È stato necessario modificare un SD del caso d'uso UC6 realizzato nella scorsa iterazione, di seguito è riportato l'SD aggiornato:

2.2.8.1 UC6 SD4 Aggiornato



2.3 Diagramma delle Classi di progetto

Per l'implementazione della notifica (tramite il pattern Observable) è stato necessario aggiungere la classe "Observer" e la classe "Observable".
Inoltre, sono stati aggiunti gli attributi "notifica" (boolean) e "messaggio" (string) alla classe Cliente.
Per la visione del Diagramma delle classi di progetto aprire il file "iterazione3.asta".

3. Testing

Per la seconda iterazione sono stati fatti i test per verificare il corretto funzionamento dei casi principali UC1, UC9, UC10.

Anche in questa iterazione si è scelto il metodo bottom-up per effettuare i test. Di seguito sono riportati i test dei metodi delle relative classi:

- **Corsa:**
  - getCorsePerPeriodo: Viene verificato che il metodo ritorni “true” quando la data relativa alla corsa rientra nel periodo fornito, e false altrimenti.
- **Biglietto:**
  - verificaBigliettoPerCorsa: Viene verificato che il metodo ritorni “true” se la corsa passata come parametro sia uguale alla corsa presente nel biglietto, falso altrimenti.
- **Cliente:**
  - annullaBigliettoPerSospensione: Viene verificato che:
    - il metodo chiamato passandogli la lista delle corse che devono essere sospese ritorni “true”;
    - l’elenco dei biglietti dell’utente sia vuoto (caso in cui l’utente abbia solo biglietti relativi alle corse che vengono sospese).
- **Tratta:**
  - sospensioneCorsa: Viene verificato che dopo aver chiamato il metodo la lista ritornata contenga le corse relative al periodo passato in ingresso al metodo.
  - eliminaCorsePerSospensione: Viene verificato che:
    - Il metodo chiamato ritorni “true”;
    - La lista delle corse sia diminuita di dimensione e che sia della dimensione attesa.
- **InViaggio:**
  - sospendiTratta: Viene verificato che il metodo chiamato ritorni “true” quando passiamo il codice di una tratta esistente, false altrimenti.
  - selezionaPeriodoSospensione: Viene verificato che il metodo chiamato ritorni la corretta lista delle corse da annullare svolte nel periodo specificato.
  - rimuoviCorsa: Viene verificato che:
    - il metodo ritorni “true” quando passiamo un codice relativo ad una corsa esistente;
    - La dimensione della mappa delle corse è diminuita di 1.
    - Il metodo ritorni “false” quando viene passato un codice di una corsa inesistente.