PawBookings: Elaborazione - Iterazione 2

Giuseppe Leocata, Alberto Provenzano, Daniele Lucifora

Introduzione

Per l'iterazione 2, sono stati scelti i seguenti requisiti:

- Implementazione del caso d'uso *UC3: Nuovo Affido.*
- Implementazione del caso d'uso *UC4: Concludi Affido.*
- Implementazione del caso d'uso *UC5: Gestisci Cliente.*
- Implementazione del caso d'uso *UC6: Gestisci Cane.*
- Implementazione del caso d'uso d'avviamento necessario per inizializzare questa iterazione.
- dati solo in memoria principale.

Aggiornamenti elaborati della fase di Ideazione

Per quanto riguarda i casi d'uso UC3 ed UC4, sono state apportate alcune correzioni ai loro flussi base.
Non si è ritenuto opportuno raffinare i restanti elaborati della fase di ideazione.

1. Analisi Orientata agli Oggetti

Al fine di descrivere il dominio da un punto di vista ad oggetti e gestire ulteriori requisiti, saranno utilizzati nuovamente gli stessi strumenti dell'iterazione precedente (Modello di Dominio, SSD Sequence System Diagram e Contratti delle operazioni).

1.1. Modello di Dominio

Dopo un'attenta analisi dello scenario principale di successo relativo al caso d'uso *UC3,* nasce l'esigenza di inserire una nuova classe concettuale:

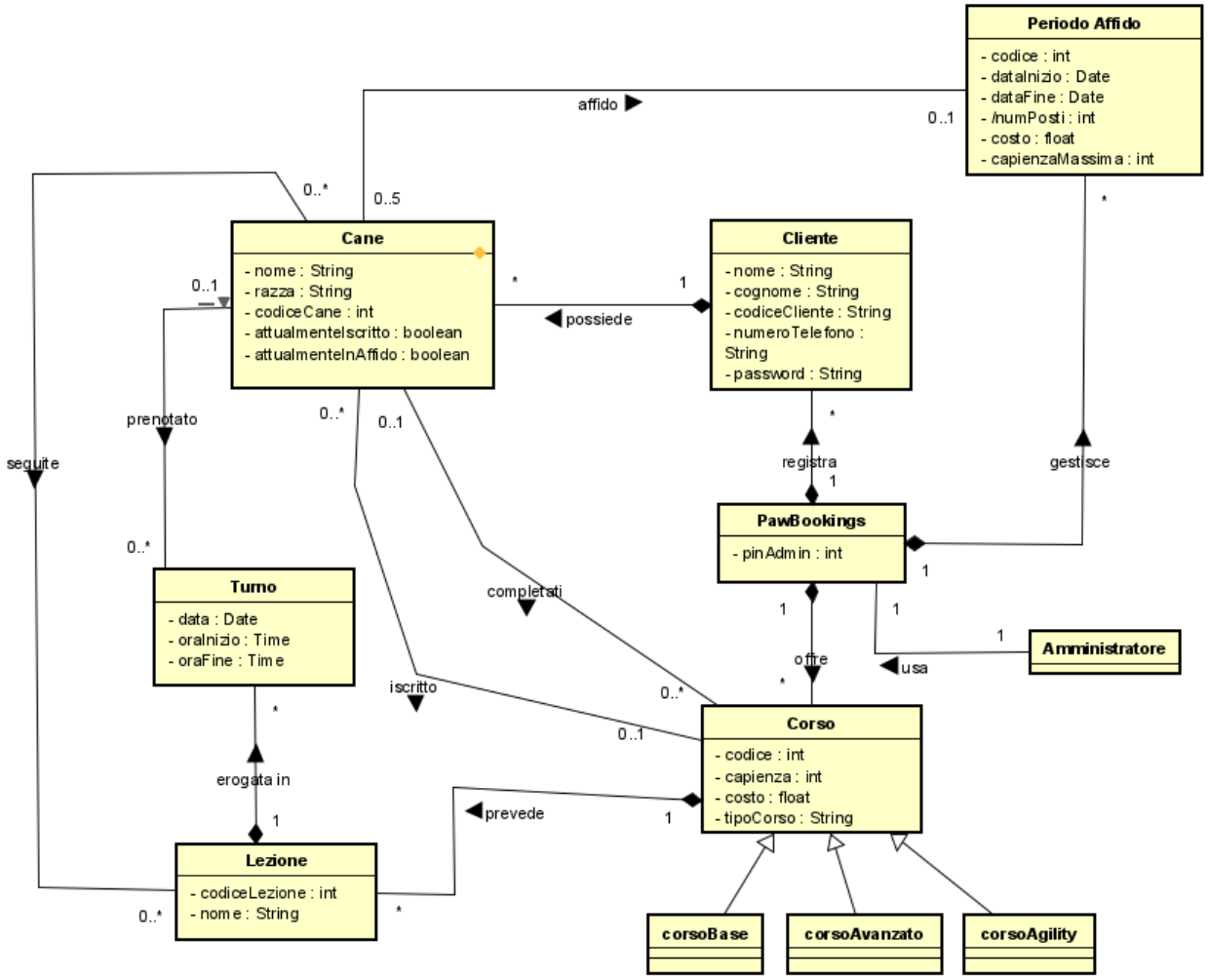
- **Periodo di Affido**

Analizzando il caso d'uso UC2, inoltre, si riscontra l'esigenza di dover aggiungere la seguente classe concettuale:

- **Amministratore**

Per quanto concerne invece l'analisi dei casi d'uso UC4 e UC5, non emerge necessità alcuna.

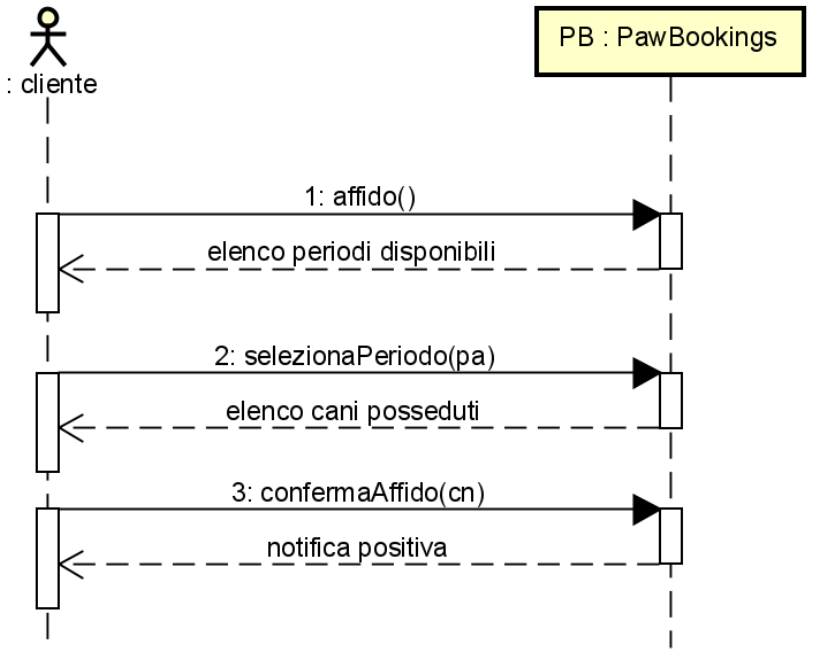
Dall'integrazione di queste nuove classi a quelle già esistenti, tenendo conto di associazioni e attributi, è stato ricavato il seguente Modello di Dominio:



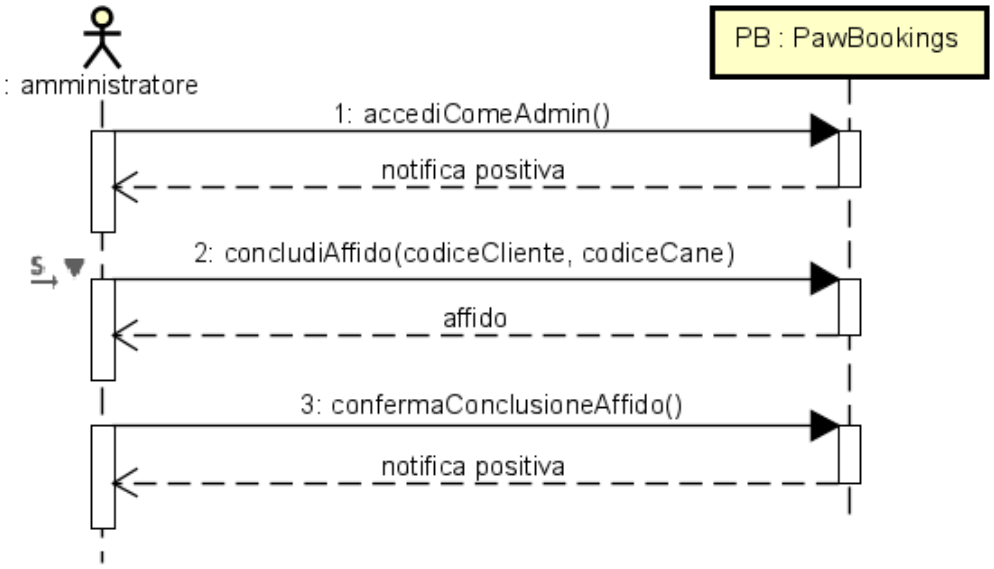
1.2. Diagrammi di Sequenza di Sistema (SSD)

Procedendo con l'OOA, il passo successivo consiste nella realizzazione dei Diagrammi di Sequenza di Sistema (SSD) relativi ai casi d'uso prescelti.

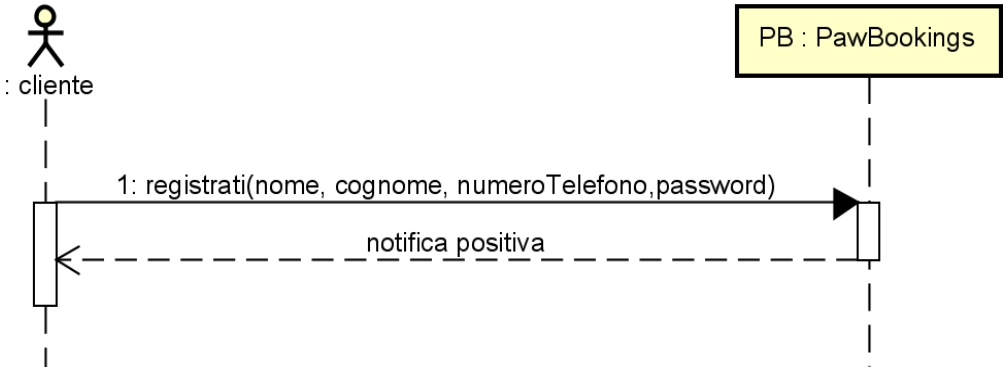
1.2.1. SSD UC3



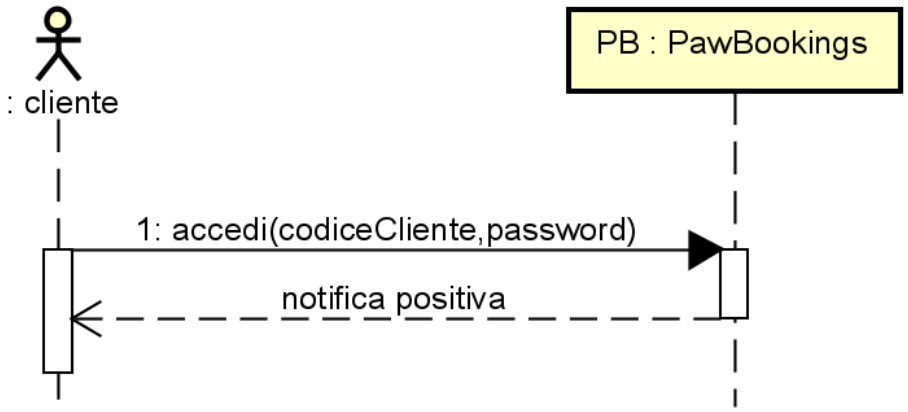
1.2.2. SSD UC4



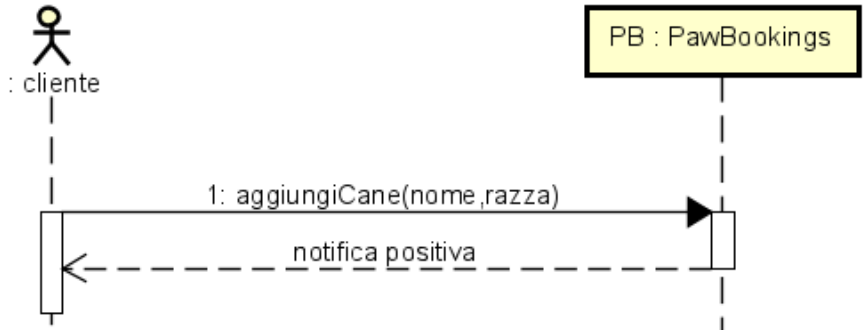
1.2.3. SSD UC5 - Scenario principale



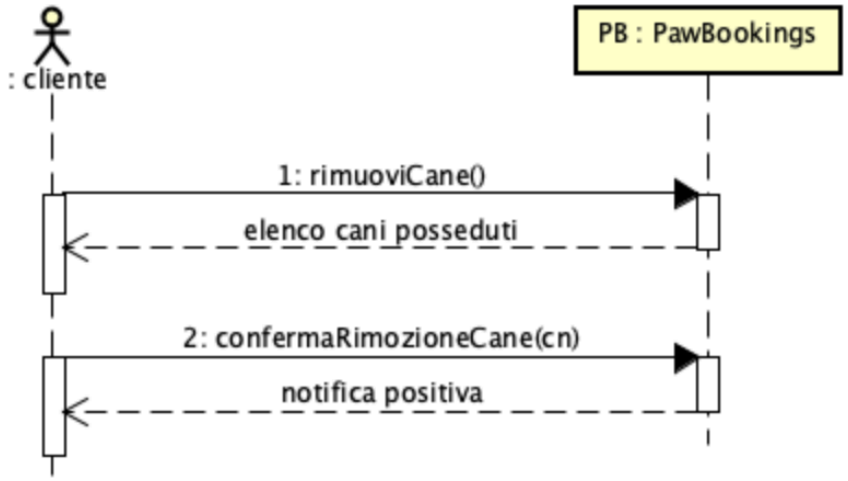
1.2.4. SSD UC5 - Estensione



1.2.5. SSD UC6 - Scenario principale



1.2.6. SSD UC6 - Estensione



1.3. Contratti delle Operazioni

1.3.1. C01: confermaAffido

operazione:	confermaAffido(cn: Cane)
riferimenti:	Caso d'uso UC3: Nuovo affido
pre-condizioni:	- è stata recuperata l'istanza *pa* della classe *PeriodoAffido* selezionata dall'utente
post-condizioni:	- cn è stata associata *pa* tramite l'associazione "affido" - *cn.attualmenteInAffido* diventa true

1.3.2. C02: confermaConclusioneAffido

operazione:	confermaConclusioneAffido()
riferimenti:	Caso d'uso UC4: Concludi affido
pre-condizioni:	- è stata recuperata l'istanza *cn* dalla classe Cane - è stata recuperata l'istanza pa della classe PeriodoAffido associata a *cn*
post-condizioni:	- *cn.attualmenteInAffido* diventa *false* - *cn* è dissociata da *pa*

1.3.3. C03: registrati

operazione:	registrati(nome: String, cognome: String, numeroTelefono: int, password: String)
riferimenti:	Caso d'uso UC5: Gestisci cliente
pre-condizioni:	- nessuna
post-condizioni:	- è stata creata un'istanza cl della classe Cliente - cl è associata a PawBookings tramite l'associazione "registra"

1.3.4. C04: aggiungiCane

operazione:	aggiungiCane(nome: String, razza: String)
riferimenti:	Caso d'uso UC6: Gestisci cane
pre-condizioni:	- è stata recuperata l'istanza *cl* della classe *Cliente*
post-condizioni:	- è stata creata un'istanza cn della classe Cane - cn è stata associata a cl tramite l'associazione "possiede"

2. Progettazione Orientata agli Oggetti

Nuovamente, l'elaborato principale di questa fase che è stato preso in considerazione è il Modello di Progetto, ovvero l'insieme dei diagrammi che descrivono la progettazione logica sia da un punto di vista dinamico (Diagrammi di Sequenza) che da un punto di vista statico (Diagramma delle Classi).

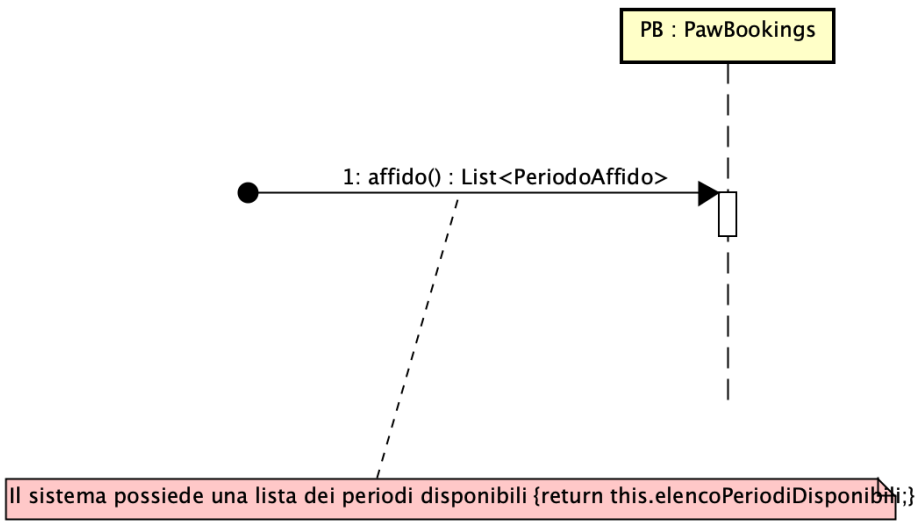
2.1. Pattern applicati

Non sono stati applicati pattern GoF in quanto non si è presentata nessuna problematica riconducibile ad uno di questi, mentre si è continuato a progettare ad oggetti. Sono stati quindi applicati altri pattern GRASP (Creator, Information Expert, Controller, Low Coupling e High Cohesion).

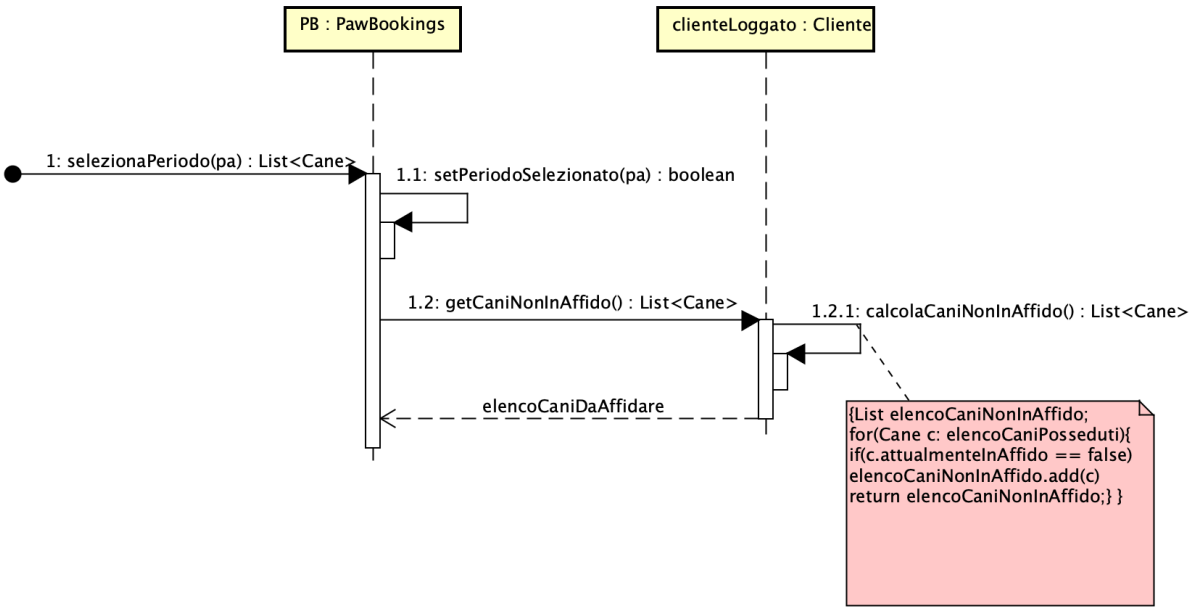
2.2. Diagrammi di Sequenza (SD)

La prima scelta di progetto da fare è sul Controller per i vari casi d'uso: si è scelto di continuare ad usare PB come facade controller.

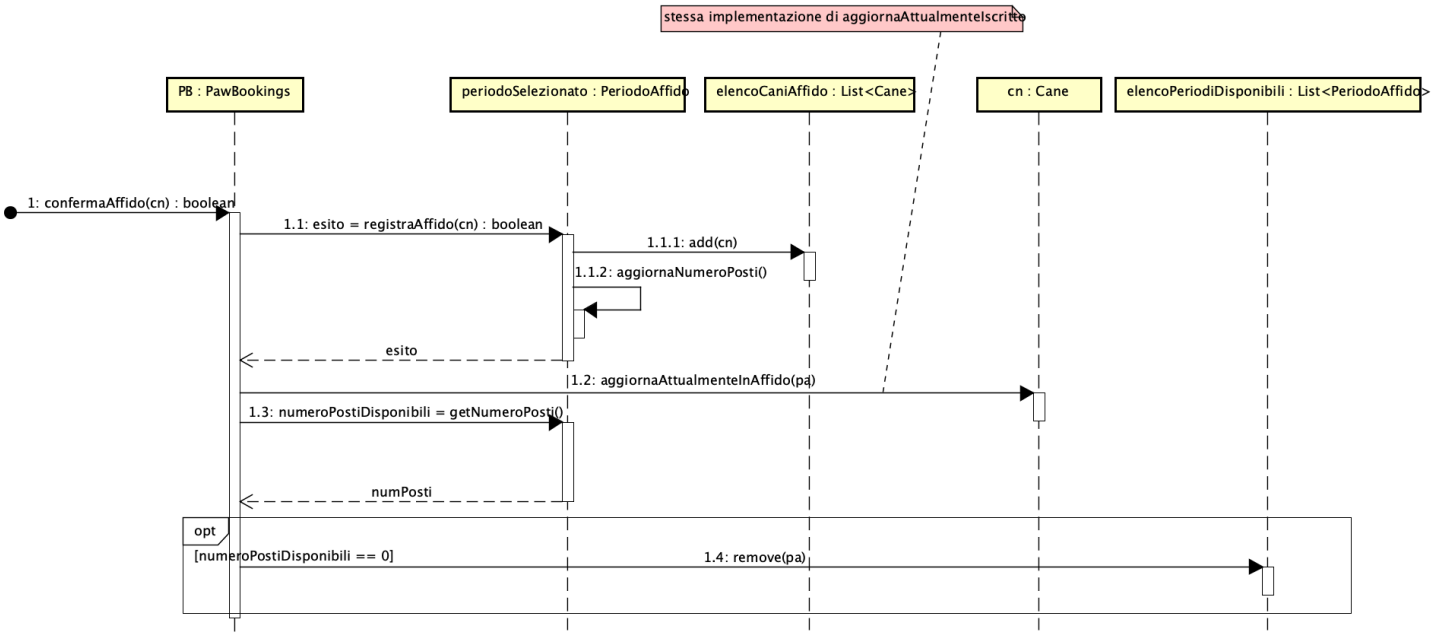
2.2.1. UC3 SD1:



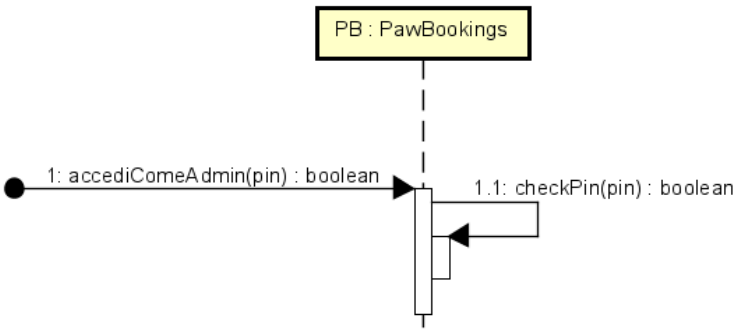
2.2.2. UC3 SD2



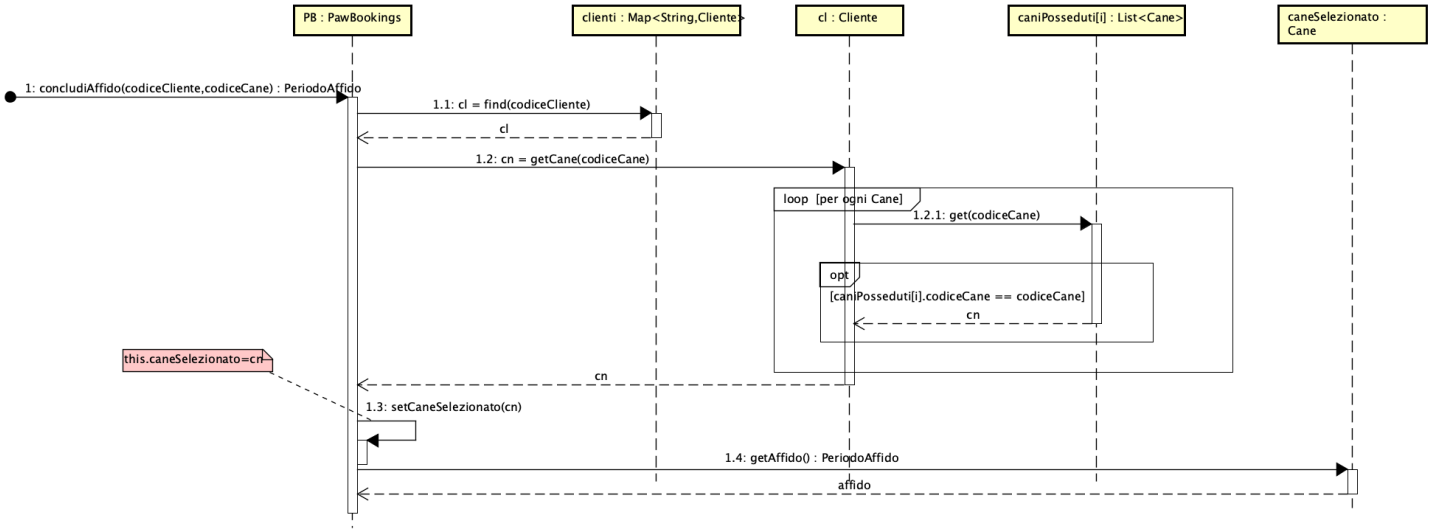
2.2.3. UC3 SD3



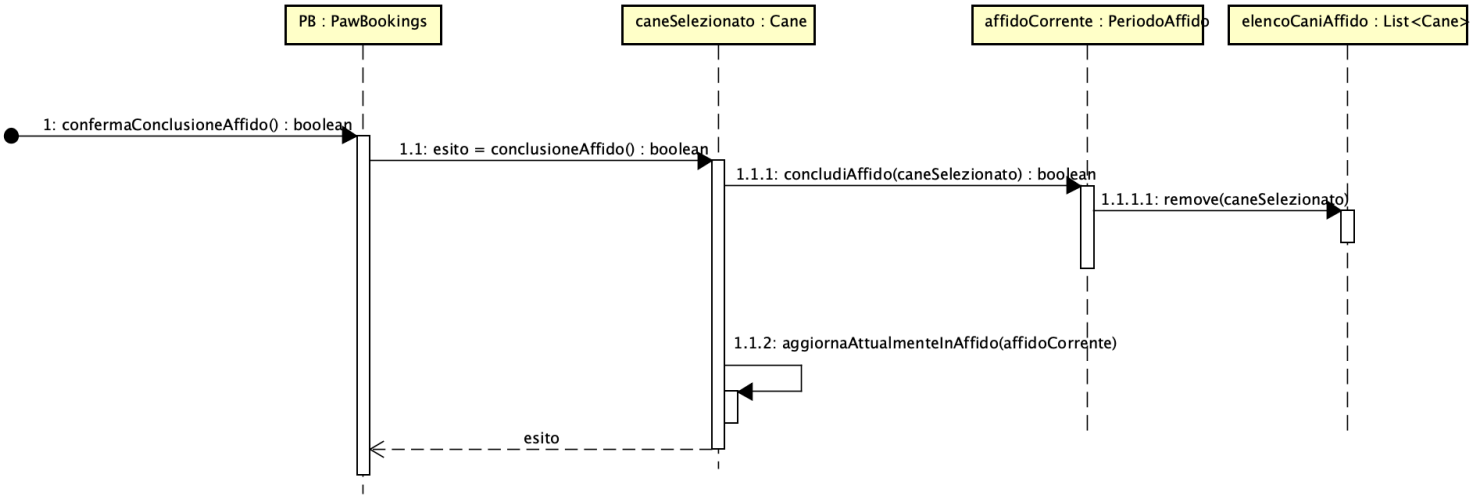
2.2.4. UC4 SD0



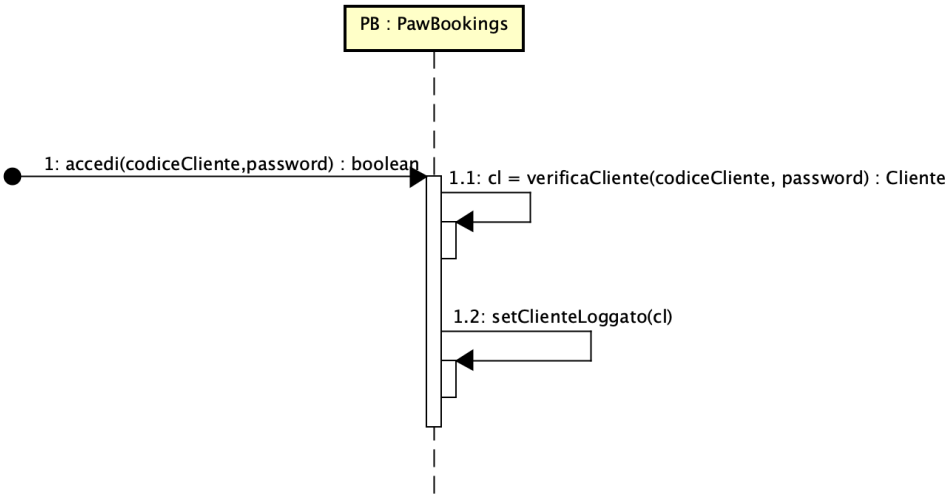
2.2.5. UC4 SD1



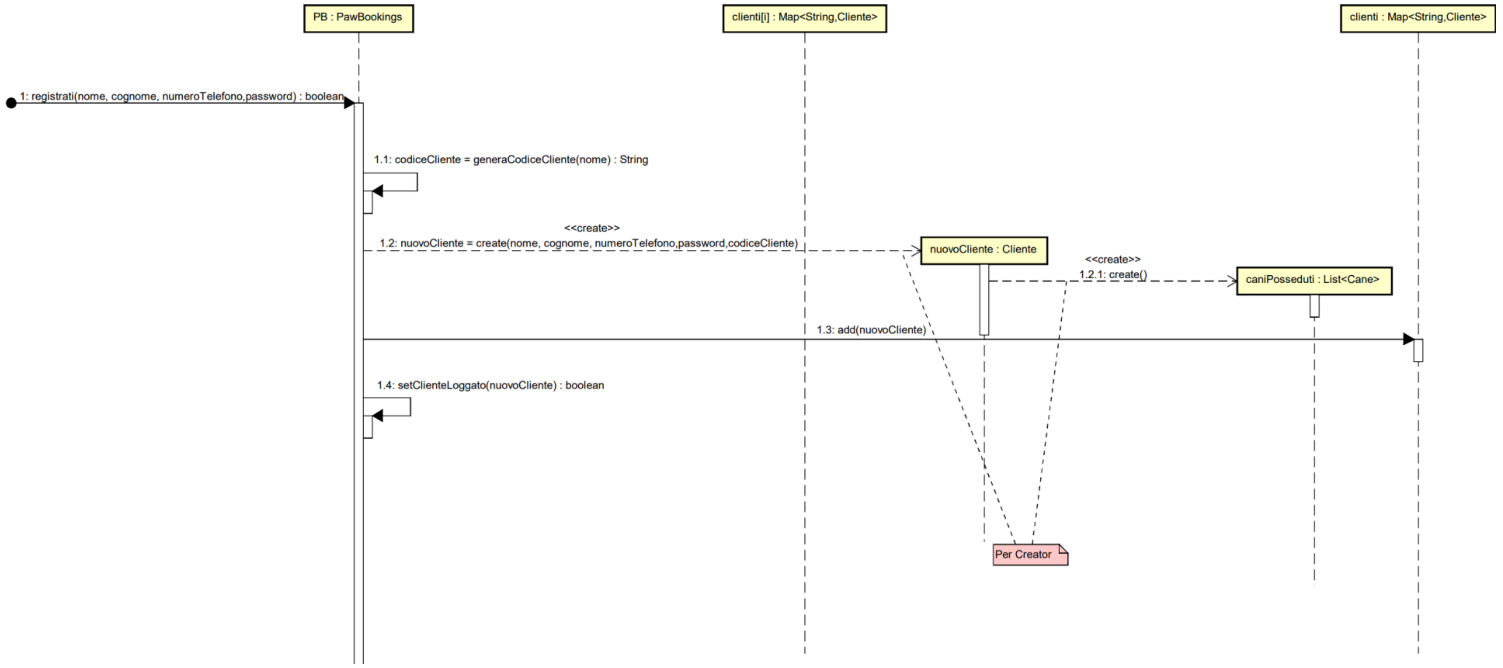
2.2.6. UC4 SD2



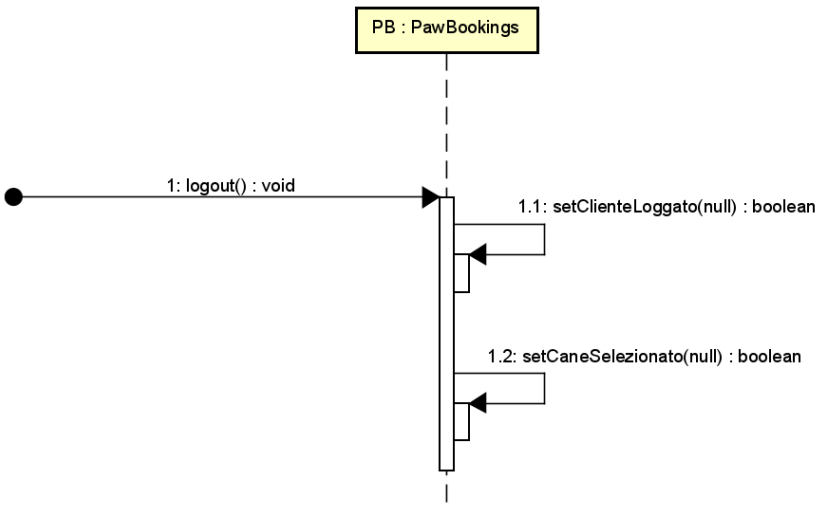
2.2.7. UC5 SD1



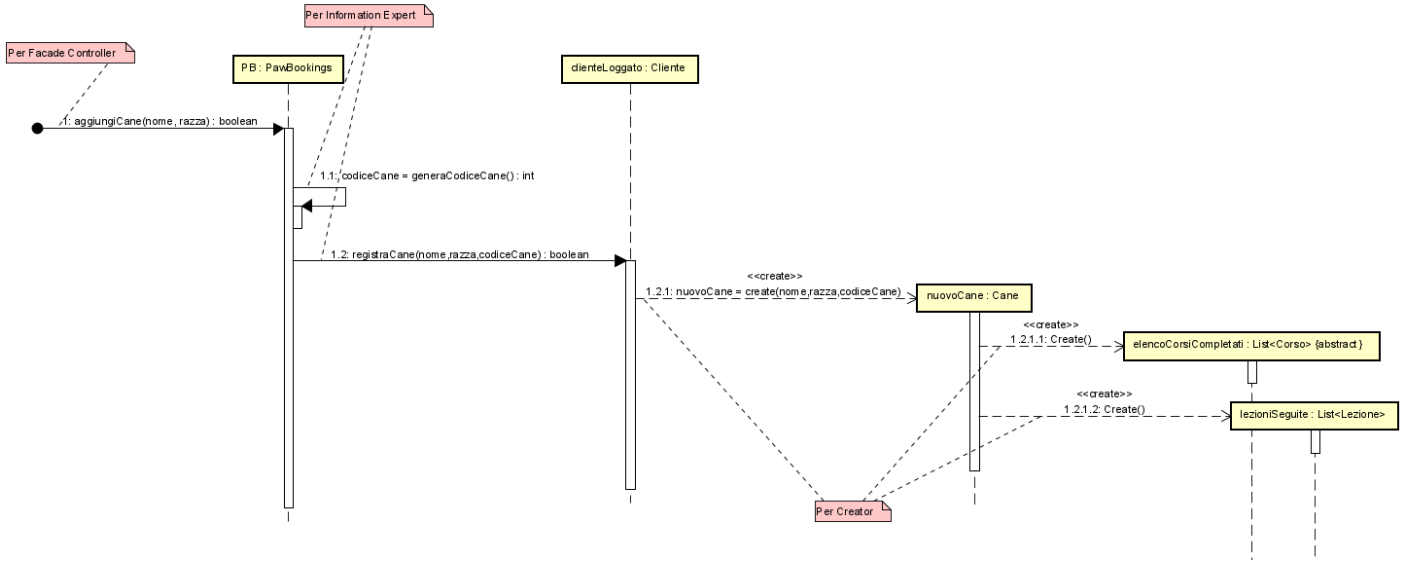
2.2.8. UC5 SD2



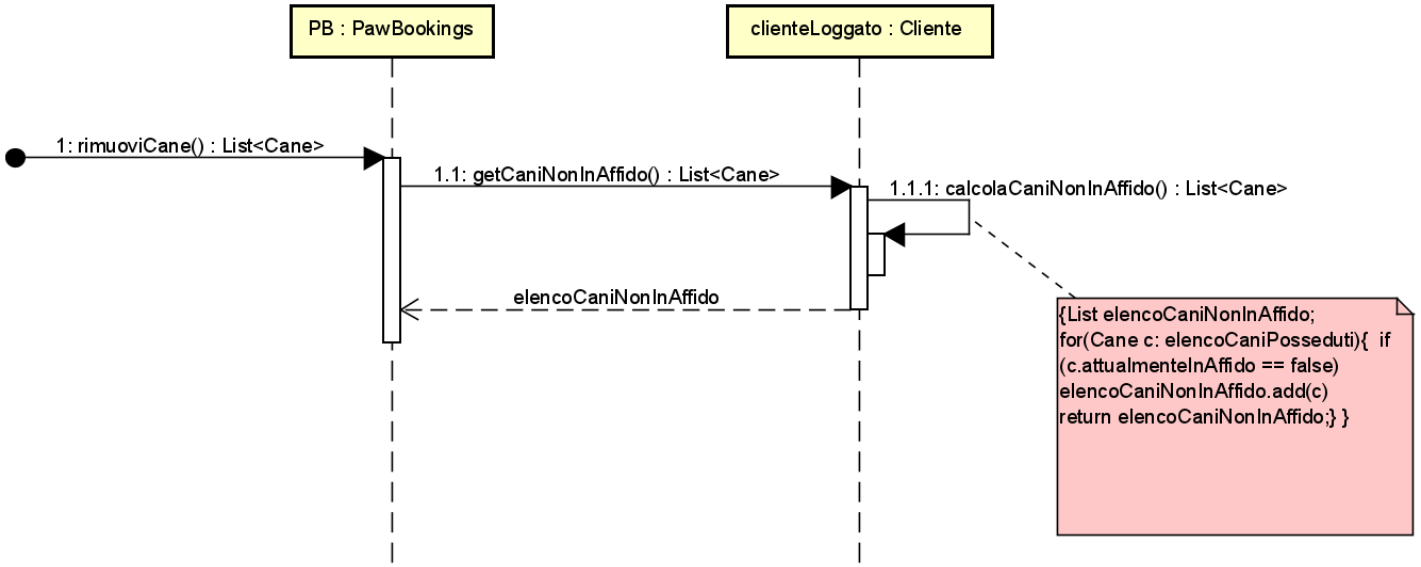
2.2.9. UC5 SD3



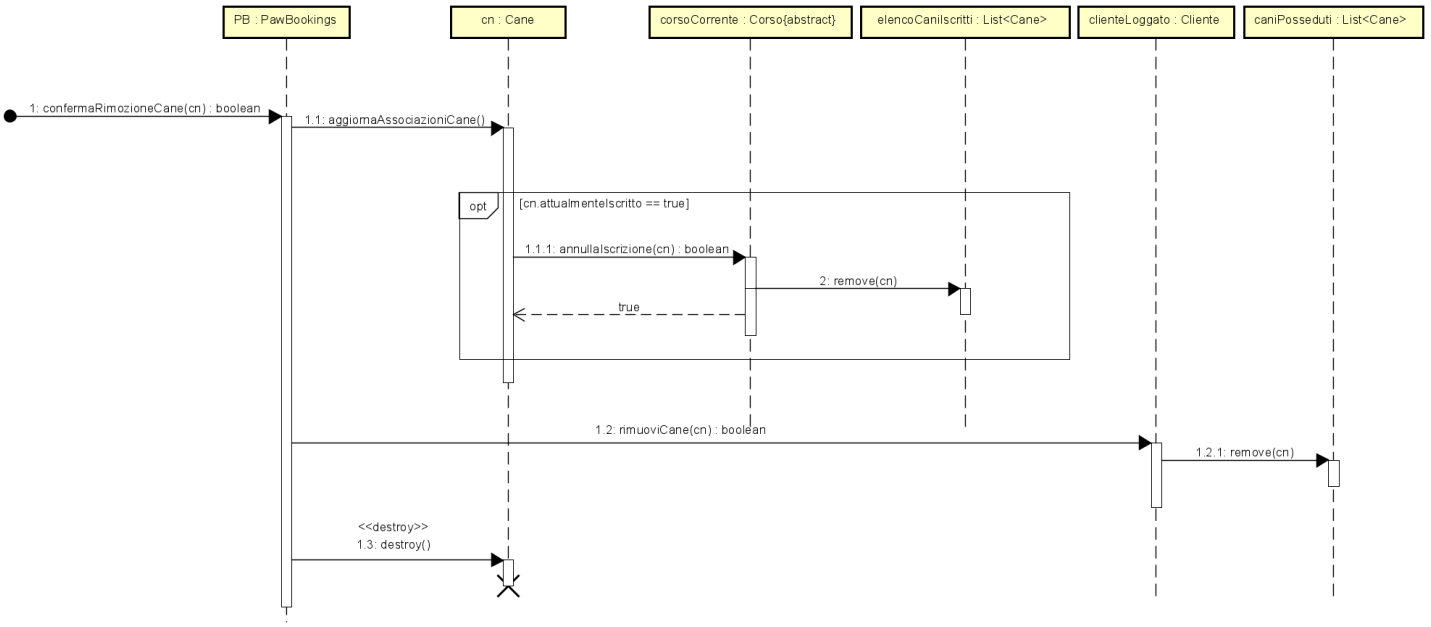
2.2.10. UC6 SD1



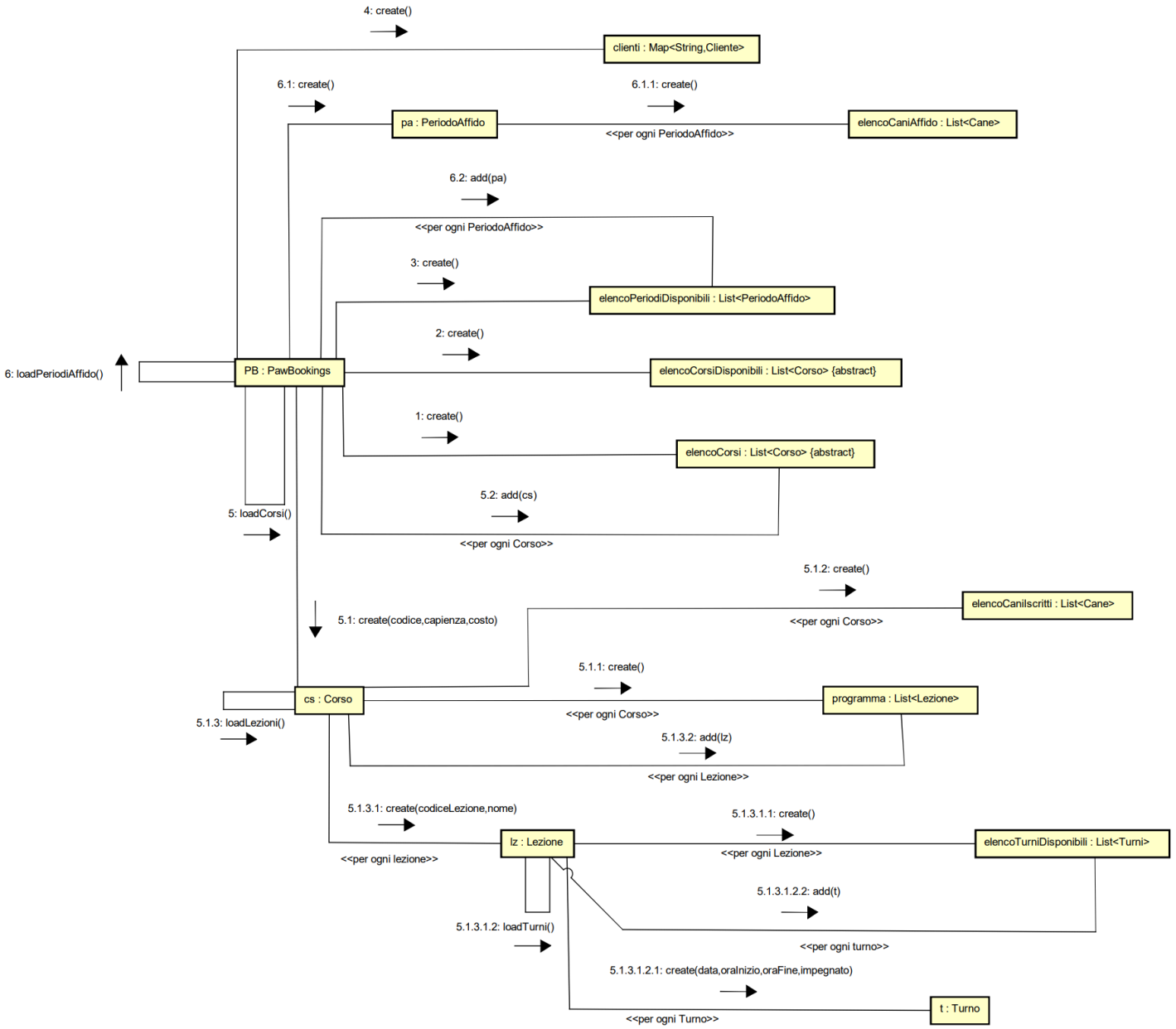
2.2.11. UC6 SD2



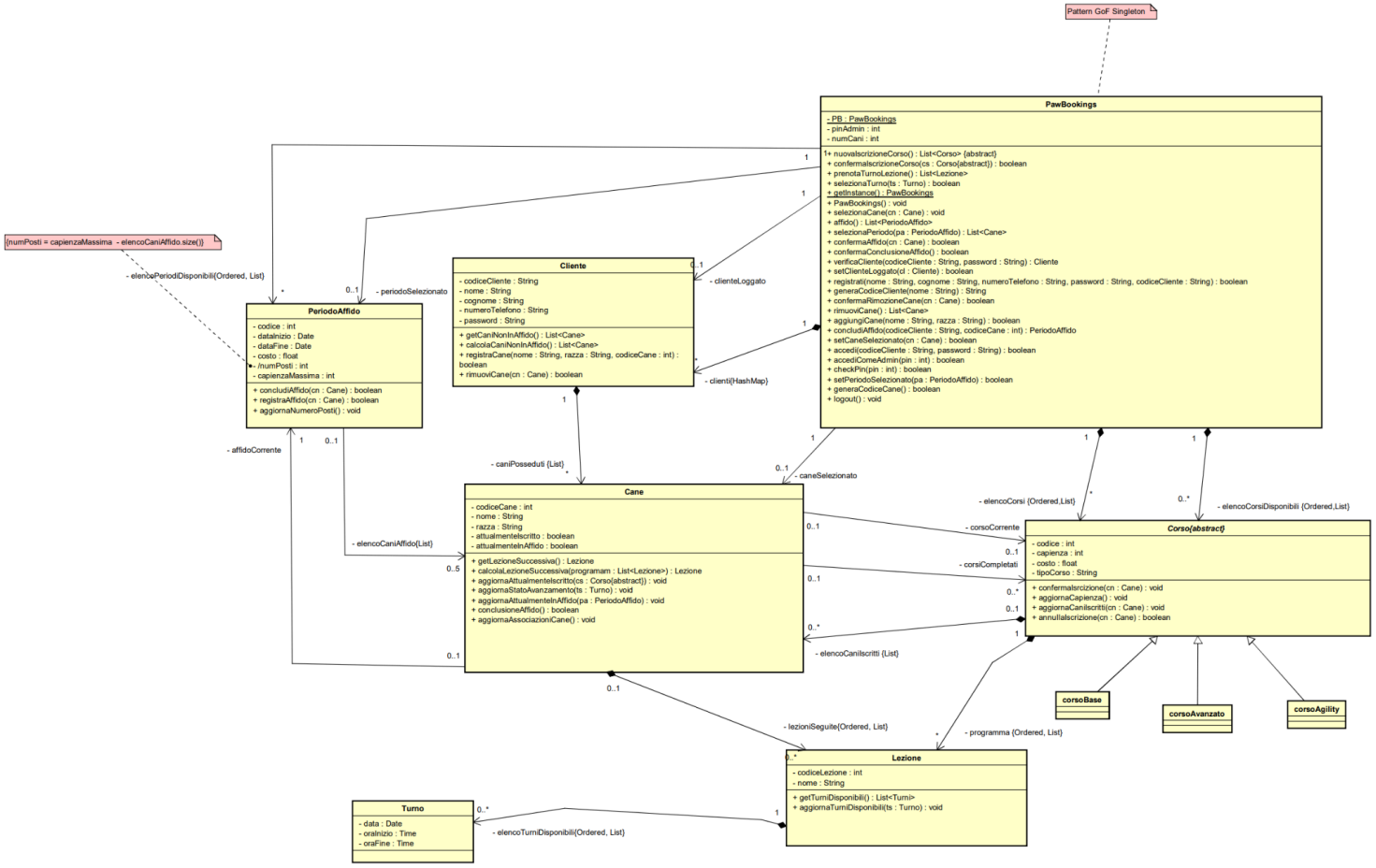
2.2.12. UC6 SD3



2.2.13. SD Caso d'uso d'avviamento



2.3. Diagramma delle Classi di Progetto



3. Testing

Per questa seconda iterazione si è scelto di concentrare il test essenzialmente sulla classe PawBookings, utilizzando un criterio di scelta dei metodi da testare che tenga conto del flusso degli scenari principali di successo UC3, UC4, e di tutti gli scenari dei casi d'uso UC5, UC6 .

Per questa seconda iterazione i test unitari relativi ai casi d'uso sopracitati sono stati implementati all'interno della classe di test PawBookingsTestIterazione2, mentre i test di unità dei metodi di PawBookings implementati nell'iterazione precedenti sono stati organizzati nella classe di test PawBookingsTestIterazione1.
In particolar modo sono stati individuati i seguenti metodi delle relative classi:

- PawBookings
  - accediComeAdmin
    - viene verificato che effettuando l'accesso, inserendo il pin corretto, venga restituito true
    - viene verificato che effettuando l'accesso, inserendo il pin errato, venga restituito false
  - registrati
    - viene verificato che la lunghezza della lista clienti di PawBookings sia stata incrementata rispetto alla lunghezza precedente
  - generaCodiceCliente
    - viene verificato che chiamando il metodo sotto test e passando come parametri il nome e il (numeroClienti +1), venga restituito il nome seguito dal valore numeroClienti+1
  - accedi
    - viene verificato che effettuando l'accesso, inserendo il codice cliente corretto e la password corretta, venga restituito true
    - viene verificato che effettuando l'accesso, inserendo il codice cliente corretto e la password corretta, il cliente loggato sia il cliente che ha effettuato il login
    - viene verificato che effettuando l'accesso, inserendo il codice cliente corretto e la password errata, venga restituito false
    - viene verificato che effettuando l'accesso, inserendo il codice cliente corretto e la password errata, il cliente loggato sia null
  - logout
    - viene verificato che chiamando il metodo sotto test il metodo getClienteLoggato restituisca null
    - viene verificato che chiamando il metodo sotto test il metodo getCaneSelezionato restituisca null
  - verificaCliente
    - viene verificato che chiamando il metodo sotto test e passando come parametri il codice cliente corretto e la password corretta, venga restituito il nome del cliente corrispondente al codice cliente inserito
    - viene verificato che chiamando il metodo sotto test e passando come parametri il codice cliente corretto e la password errata, non venga restituito il nome del cliente corrispondente al codice cliente inserito
  - setClienteLoggato
    - viene verificato che chiamando il metodo sotto test e passando come parametro il cliente, venga restituito il cliente passato come parametro
    - viene verificato che chiamando il metodo sotto test e passando come parametro il cliente, non venga restituito il cliente passato come parametro

- aggiungiCane
  - viene verificato che chiamando il metodo sotto test, la lista dei cani posseduti dal cliente loggato sia stata incrementata
  - viene verificato che chiamando il metodo sotto test e passando come parametri il nome del cane e la razza, venga restituito true se la lista dei cani posseduti dal cliente contiene il cane inserito dal metodo sotto test
- generaCodiceCane
  - viene verificato che chiamando il metodo sotto test venga incrementato di 1 il codice del cane
- rimuoviCane
  - viene verificato che chiamando il metodo sotto test venga restituita la lista dei cani posseduti dal cliente che non si trovano attualmente in affido
- confermaRimozioneCane
  - viene verificato che chiamando il metodo sotto test e passando come parametro il cane, quest'ultimo venga rimosso dall'elenco dei cani iscritti al corso in cui il cane era registrato
  - viene verificato che chiamando il metodo sotto test e passando come parametro il cane, quest'ultimo venga rimosso dall'elenco dei cani posseduti dal cliente
- selezionaPeriodo
  - viene verificato che chiamando il metodo sotto test e passando come parametro il periodo di affido, nelle lista dei cani non in affido non sia presente il cane precedentemente messo in affido
  - viene verificato che chiamando il metodo sotto test e passando come parametro il periodo selezionato, venga restituito il periodo selezionato
- confermaAffido
  - viene verificato che chiamando per 4 volte il metodo sotto test, nell'elenco cani in affido del periodo selezionato siano presenti 4 cani
  - viene verificato che nell'elenco dei cani in affido del periodo selezionato sia presente il cane passato come parametro del metodo sotto test
  - viene verificato che chiamando il metodo sotto test, il numero di posti disponibili del corso nel periodo selezionato sia uguale al numero di posti del corso nel periodo selezionato meno il numero di cani in affido nel periodo selezionato
  - viene verificato che chiamando il metodo sotto test , il periodo di affido selezionato non appartenga più all'elenco dei periodi di affido disponibili
- concludiAffido
  - viene verificato che chiamando il metodo sotto test e passando come parametro il periodo di affido, venga restituito il periodo di affido
- confermaConclusioneAffido
  - viene verificato che chiamando il metodo sotto test, venga rimosso il cane dall'elenco dei cani in affido del periodo selezionato
  - viene verificato che chiamando il metodo sotto test, l'attributo attualmenteInAffido del Cane selezionato diventi false
  - viene verificato che chiamando il metodo sotto test, l'attributo affidoCorrente del Cane selezionato diventi null