

PROGETTO INGEGNERIA DEL SOFTWARE



Ingegneria del Software A.A. 2021/2022

SVILUPPATORI:

Marisonia Ferrandino

Alessandro Guaitini

Lorenza Alfano



GESTIONE DI UNA PISCINA COMUNALE

Il 23/11/2021 abbiamo avuto un incontro con il nostro cliente per la progettazione di un'applicazione per gestire una piscina comunale con dei servizi di: abbonamenti, corsi di nuoto, liste di clienti e dipendenti e oggetti smarriti. Parlando con il nostro cliente abbiamo innanzitutto pattuito delle responsabilità professionali tra cui: riservatezza, competenza, diritti di proprietà intellettuale e uso improprio del computer.

INDICE

INTRODUZIONE

- **Intervista**
- **Descrizione in linguaggio naturale**
- **Glossario dei termini**

ANALISI DEI REQUISITI

- 1. Requisiti funzionali**
- 2. Requisiti non funzionali**

CASI D'USO

- **Gestione di attività**
- **Attori**
- **Gestione clienti**
 - 1. Visualizza cliente**
 - a. Inserisci abbonamento
 - b. Visualizza abbonamento
 - 2. Elimina cliente**
 - 3. Inserisci cliente**
- **Gestione dipendenti**
 - 1. Visualizza dipendente**
 - 2. Elimina dipendente**
 - 3. Inserisci dipendente**

- **Gestione Servizio**
 1. Visualizza servizio
 2. Elimina servizio
 3. Inserisci Servizio
- **Gestione prenotazione**
 1. Visualizza prenotazione
 2. Elimina prenotazione
 3. inserisci prenotazione
- **Gestione oggetti**
 1. Visualizza oggetto
 2. Elimina oggetto
 3. Inserisci oggetto
- **Visualizzazione globale casi d'uso**
- **Matrice di Mapping**
- **Mappa dell'architettura**
- **Diagramma delle classi**
- **Diagrammi di sequenza**
- **Diagrammi di attività**
- **Uni Testing**



• INTERVISTA

CLIENTE: Buongiorno, vorrei delle informazioni sulla creazione di un software per la gestione di una piscina comunale

TEAM: Buongiorno, se ci fa sapere cosa le serve nel dettaglio possiamo capire se saremo in grado di progettarlo secondo e sue necessità.

CLIENTE: Quello a cui pensavamo noi era un programma per facilitare la gestione della piscina e la comunicazione tra cliente e gestore.

TEAM: Per noi va bene, ci può elencare tutti i servizi che lei vuole sul programma?

CLIENTE: Vorremo che il nostro programma gestisca:

- abbonamenti (1/3/6/12 mesi)
- corsi di nuoto
- memorizzazione dei dati personali di ogni cliente
- memorizzazione dei dati dei dipendenti
- lista di oggetti smarriti

TEAM: Va bene, prenderemo in considerazione le sue necessità e le faremo sapere appena abbiamo un prototipo funzionante. Se vuole aggiungere qualcos'altro non esiti a contattarci.

CLIENTE: Grazie per la disponibilità, le manderò tutti i dettagli per e-mail il prima possibile.

• DESCRIZIONE IN LINGUAGGIO NATURALE

Lo stabilimento rimarrà aperto tutto l'anno eccetto durante le festività.

Nello specifico i corsi sono:

corso per adulti e bambini ovviamente la piscina sarà disponibile ad allenamenti anche per chi vorrà sostenere gare.
Per chi non è interessato ai corsi può avere accesso a una corsia per il nuoto libero.

Il sistema deve tenere traccia della licenza dei bagnini compreso il loro nome, numero di telefono, data di nascita e luogo, e-mail.

I dipendenti sono registrati con i loro dati anagrafici nome, cognome, e-mail data e luogo di nascita. Inoltre, per ogni cliente si dovranno memorizzare (cognome, nome, data di nascita, indirizzo e-mail e telefono) e abbonamento con ulteriore scadenza.

Un cliente può, inoltre, sottoscrivere un abbonamento, che può essere da uno, tre, sei e dodici mesi. Per ogni abbonamento si dovranno memorizzare la data di rilascio e la data di inizio validità.

Solo ai clienti abbonati è consentita la prenotazione dei servizi, tra cui corsie per nuoto e ciascuna prenotazione verranno memorizzati un codice, la data e l'ora di inizio.

La prenotazione si può riferire ad uno o più servizi.

Come in ogni luogo comune per distrazione delle volte si perdono accessori di valori importanti, abbiamo messo a disposizione su l'applicazione una funzione che ti permette di registrare l'oggetto perso con il tipo, il nome e descrizione ed inoltre la data di ritrovamento e dare modo agli altri clienti di poter visualizzare nel caso in cui proprio loro hanno trovato l'oggetto di un altro cliente.

Nel momento in cui si ritrova l'oggetto bisogna rimuoverlo.

- **GLOSSARIO DEI TERMINI**

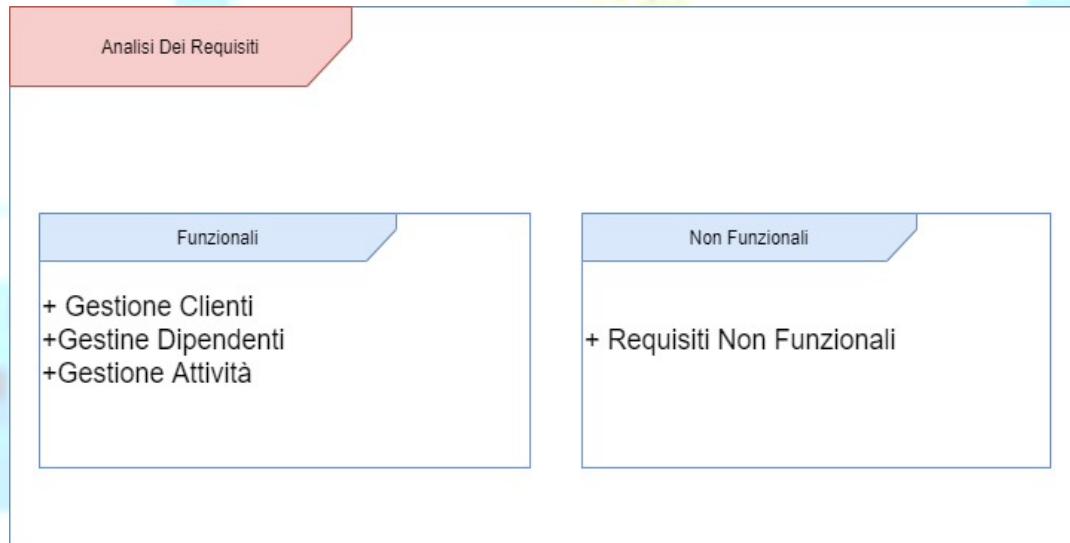
Termine	Descrizione	Tipo	Sinonimi
Piscina comunale	Edificio opportunamente attrezzato, posto in Ancona Centro	Business	Nessuno
Servizio	Complesso di strutture offerte ai clienti	Business	Nessuno
Prenotazione	Atto mediante il quale vengono riservati uno o più servizi ad un cliente per un certo tempo	Business	Assegnamento
Cliente	Colui che usufruisce dei servizi offerti dalla piscina	Business	Utente
Abbonamento	Documento che viene rilasciato al cliente che ne abbia fatto richiesta e che consente di effettuare prenotazioni.	Business	Nessuno
Oggetti	Lista prodotti oggetti persi	Business	Nessuno
Dipendente	Colui che lavora all'interno della piscina con riferito una licenza	Business	Nessuno

• ANALISI DEI REQUISITI

Innanzitutto, dividiamo i nostri requisiti funzionali e non funzionali.

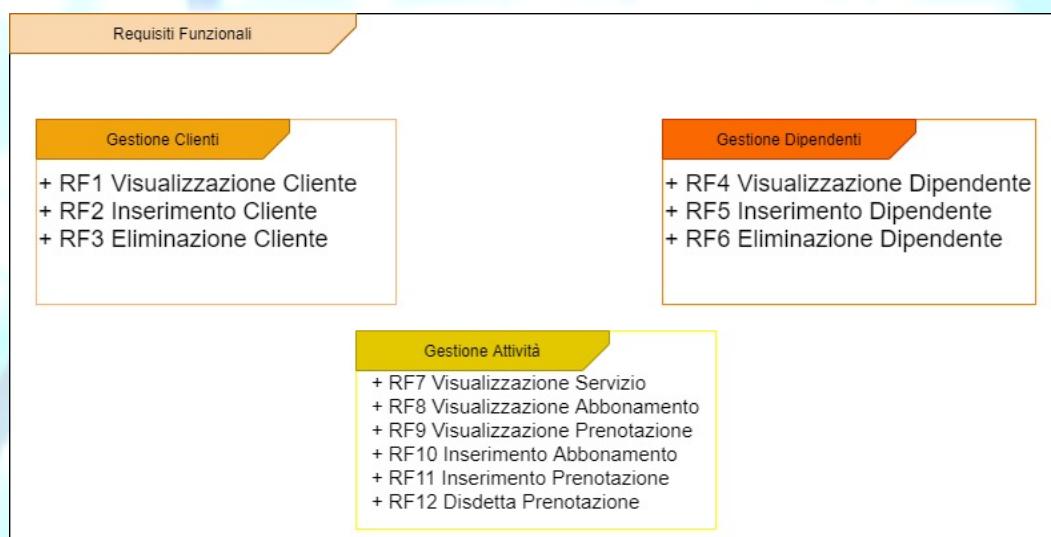
I requisiti funzionali si presentano come liste di funzionalità o servizi che il sistema deve fornire.

I requisiti non funzionali rappresentano vincoli e le proprietà relative al sistema, vincoli sul processo di sviluppo e sugli standard da adottare. Non riguardano solo il sistema software sviluppato ma alcuni vincolano il processo usato per sviluppare il sistema.



I requisiti possono essere raggruppati deducibili dalla descrizione in linguaggio naturale, successivamente si elencano tutti i requisiti del nostro sistema.

1. REQUISITI FUNZIONALI



Requisito	Descrizione
RF1 Visualizzazione Cliente	Il sistema dovrà gestire la visualizzazione di un cliente.
RF2 Inserimento Cliente	Il sistema dovrà gestire l'inserimento di un cliente.
RF3 Eliminazione Cliente	Il sistema dovrà gestire l'eliminazione di un cliente.
RF4 Visualizzazione Dipendente	Il sistema dovrà gestire la visualizzazione di un dipendente.
RF5 Inserimento Dipendente	Il sistema dovrà gestire l'inserimento di un dipendente.
RF6 Eliminazione Dipendente	Il sistema dovrà gestire l'eliminazione di un dipendente.
RF7 Visualizza Servizio	Il sistema dovrà consentire la visualizzazione delle informazioni e dello stato di un servizio.
RF8 Visualizza Abbonamento	Il sistema dovrà consentire la visualizzazione delle informazioni e dello stato di un abbonamento.
RF9 Visualizza Prenotazione	Il sistema dovrà consentire la visualizzazione delle informazioni e dello stato di una prenotazione.
RF10 Inserimento Abbonamento	Il sistema dovrà gestire l'inserimento di un abbonamento.
RF11 Inserimento Prenotazione	Il sistema dovrà gestire l'inserimento di una prenotazione.
RF12 Elimina Prenotazione	Il sistema dovrà gestire l'eliminazione di una prenotazione.
RF13 Visualizza Oggetto	Il sistema dovrà consentire la visualizzazione delle informazioni e dello stato di un oggetto.
RF14 Inserimento Oggetto	Il sistema dovrà gestire l'inserimento di un oggetto.
RF15 Elimina Oggetto	Il sistema dovrà gestire l'eliminazione di un oggetto.

2. REQUISITI NON FUNZIONALI

Requisiti non funzionali

Requisiti non funzionali

+RNF1 Implementazioni

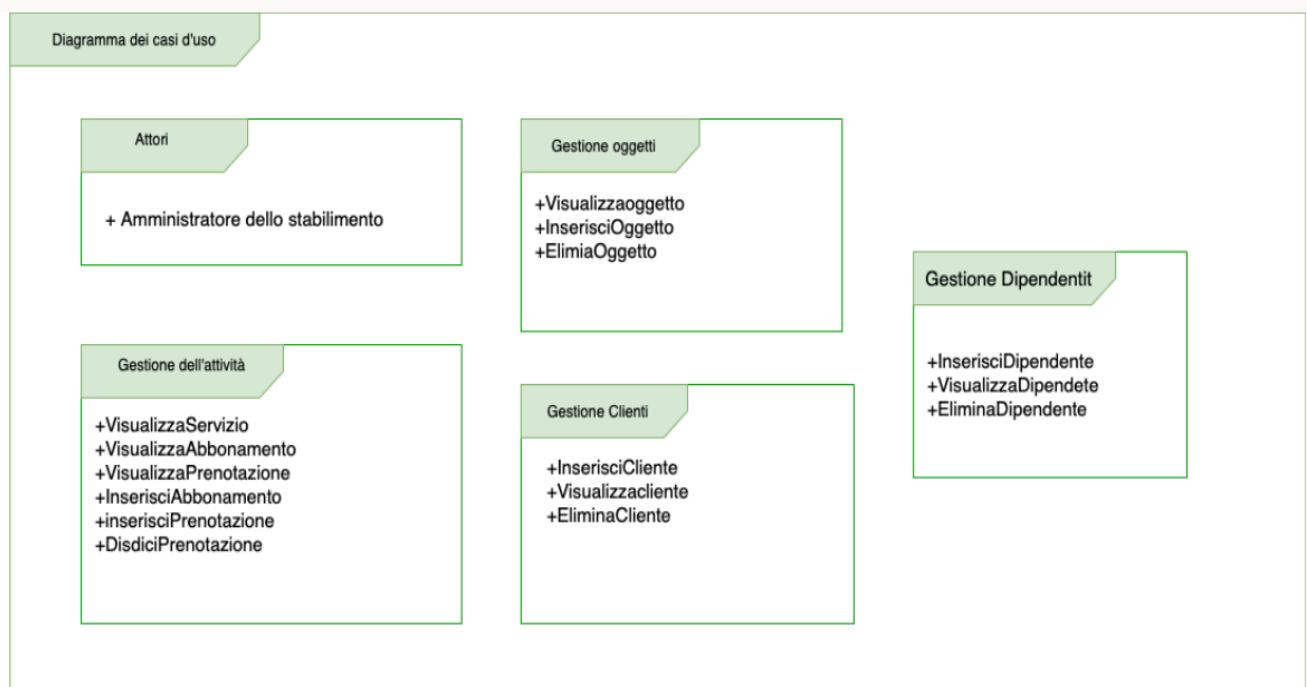
+RNF2 Interfaccia Grafica

Requisito	Descrizione
RNF1 Implementazione	Il sistema dovrà essere realizzato in tecnologia Python3
RNF2 Interfaccia Grafica	Il sistema dovrà essere dotato di interfaccia grafica

• CASI D'USO

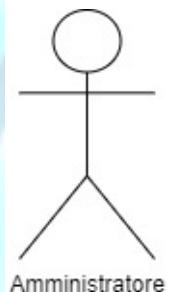
• GESTIONE ATTIVITA'

Adesso andiamo a definire i casi d'uso, che si potranno verificare durante l'utilizzo del nostro software.



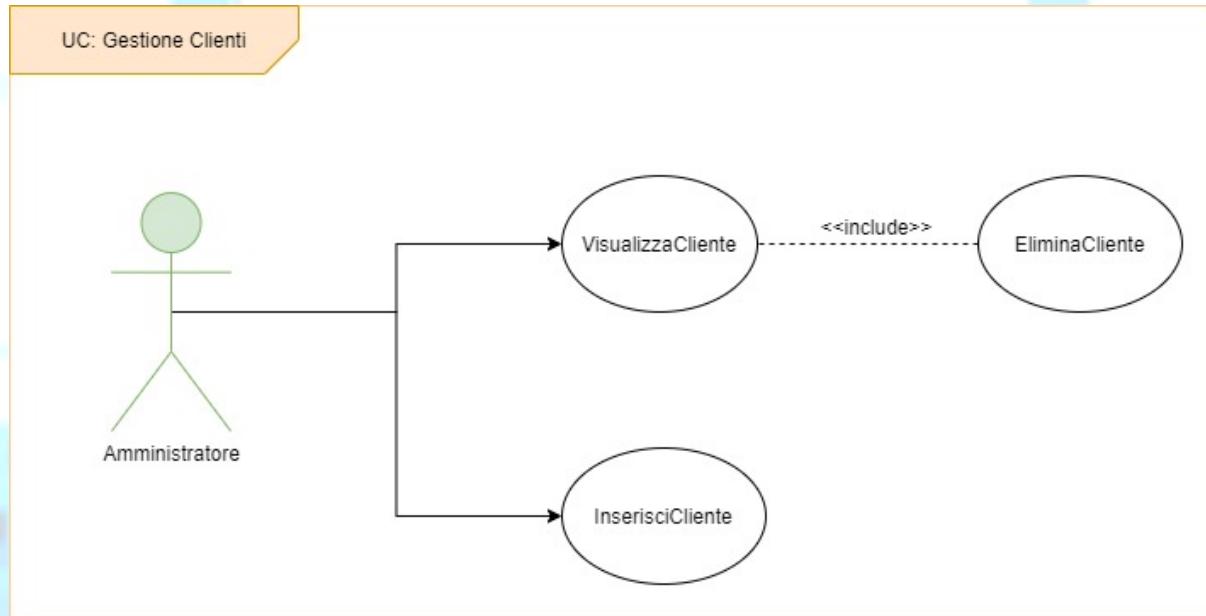
• CASI D'USO: ATTORI

Nel nostro software abbiamo un solo attore che interagisce con il sistema, l'Amministratore della piscina comunale.



- CASI D'USO: GESTIONE CLIENTI

Per ogni gruppo abbiamo definito un corrispettivo diagramma e descrizione.



1. CASI D'USO: VISUALIZZA CLIENTI

Questo caso d'uso si verifica qualora l'amministratore voglia visualizzare le informazioni di un cliente.

Pre-condizioni

Il cliente esiste a sistema

Post-condizioni

Nessuna

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole visualizzare le informazioni di un cliente
2. Il sistema legge le informazioni del cliente scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni del cliente

Sequenza degli eventi alternativa

Nessuna

2. CASI D'USO: ELIMINA CLIENTE

Questo caso d'uso si verifica qualora l'amministratore voglia eliminare un cliente dal sistema.

Pre-condizioni

Il cliente esiste a sistema

Post-condizioni

Il cliente non esiste più a sistema

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole eliminare le informazioni di un cliente
2. Il sistema legge le informazioni del cliente scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni del cliente
4. L'amministratore avvia la procedura di eliminazione
5. Il sistema elimina dal sistema le informazioni del cliente

Sequenza degli eventi alternativa

Nessuna

3. CASI D'USO: INSERISCICLIENTE

Questo caso d'uso si verifica qualora l'amministratore voglia inserire un nuovo cliente a sistema.

Pre-condizioni

Il cliente non esiste a sistema

Post-condizioni

Il nuovo cliente esiste a sistema (o si è verificata l'impossibilità di aggiungerlo)

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole inserire le informazioni di un nuovo cliente
2. Il sistema visualizza la schermata di inserimento informazioni del nuovo cliente 3. L'amministratore fornisce tutte le informazioni richieste
4. L'amministratore avvia la procedura di inserimento nuovo cliente
5. Il sistema aggiunge con successo il nuovo cliente a sistema

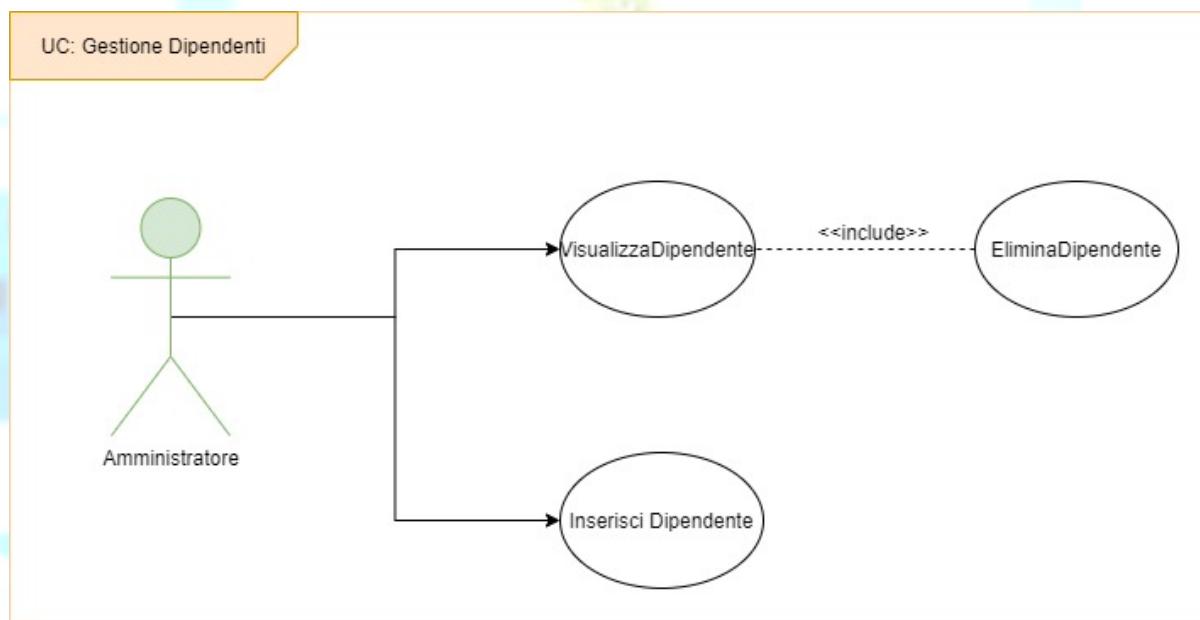
Sequenza degli eventi alternativa

La sequenza alternativa inizia dal punto 4.

5. Le informazioni inserite dall'amministratore sono incomplete

6. L'inserimento nuovo cliente fallisce.

• CASI D'USO: GESTIONE DIPENDENTE



1. CASI D'USO: VISUALIZZA DIPENDENTE

Questo caso d'uso si verifica qualora l'amministratore voglia visualizzare le informazioni di un dipendente.

Pre-condizioni

Il dipendente esiste a sistema

Post-condizioni

Nessuna

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole visualizzare le informazioni di un dipendente
2. Il sistema legge le informazioni del dipendente scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni del dipendente

Sequenza degli eventi alternativa: Nessuna

2. CASI D'USO: ELIMINADIPENDENTE

Questo caso d'uso si verifica qualora l'amministratore voglia eliminare un dipendente dal sistema.

Pre-condizioni

Il dipendente esiste a sistema

Post-condizioni

Il dipendente non esiste più a sistema

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole eliminare le informazioni di un dipendente
2. Il sistema legge le informazioni del dipendente scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni del dipendente
4. L'amministratore avvia la procedura di eliminazione
5. Il sistema elimina dal sistema le informazioni del dipendente

Sequenza degli eventi alternativa

Nessuna

3. CASI D'USO: INSERISCI DIPENDENTE

Questo caso d'uso si verifica qualora l'amministratore voglia inserire un nuovo dipendente a sistema.

Pre-condizioni

Il dipendente non esiste a sistema

Post-condizioni

Il nuovo dipendente esiste a sistema (o si è verificata l'impossibilità di aggiungerlo)

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole inserire le informazioni di un nuovo dipendente
2. Il sistema visualizza la schermata di inserimento informazioni del nuovo dipendente
3. L'amministratore fornisce tutte le informazioni richieste
4. L'amministratore avvia la procedura di inserimento nuovo dipendente
5. Il sistema aggiunge con successo il nuovo dipendente a sistema

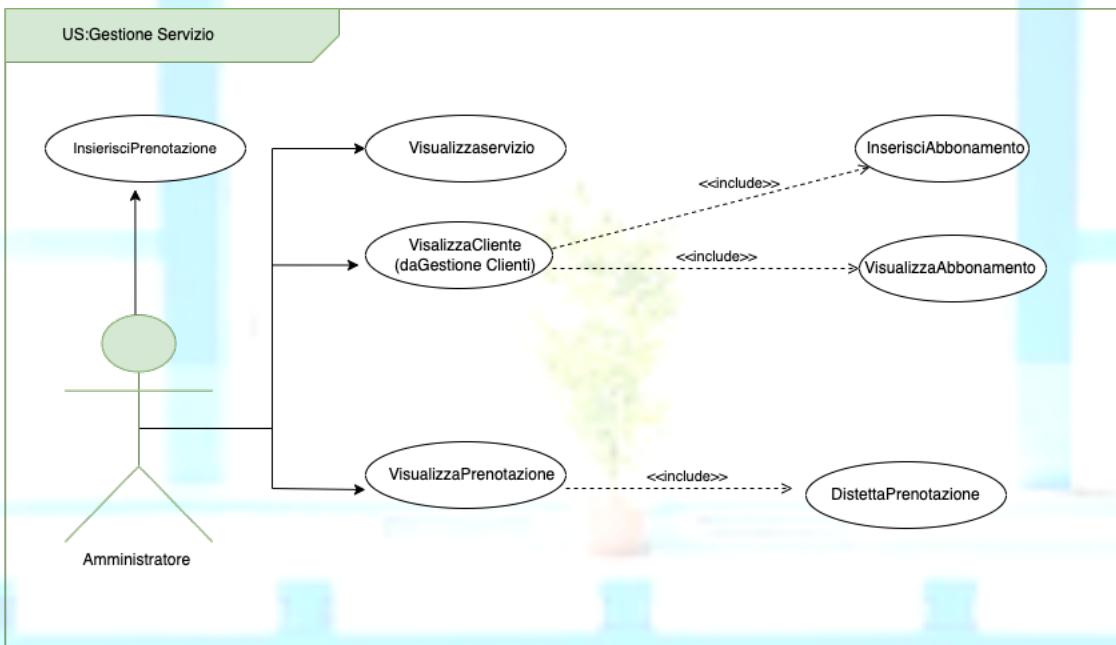
Sequenza degli eventi alternativa

La sequenza alternativa inizia dal punto 4.

5. Le informazioni inserite dall'amministratore sono incomplete

6. L'inserimento nuovo dipendente fallisce

• CASI D'USO: GESTIONE SERVIZIO



1. CASI D'USO: VISUALIZZAZIONE SERVIZIO

Questo caso d'uso si verifica qualora l'amministratore voglia visualizzare le informazioni di un servizio.

Pre-condizioni

Il servizio esiste a sistema

Post-condizioni

Nessuna

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole visualizzare le informazioni di un servizio
2. Il sistema legge le informazioni del servizio scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni del servizio

Sequenza degli eventi alternativa

Nessuna

2. CASI D'USO: INSERISCI ABBONAMENTO

Questo caso d'uso si verifica qualora l'amministratore voglia inserire un nuovo abbonamento pagato da un cliente.

Pre-condizioni

Il cliente non ha un abbonamento attivo

Post-condizioni

Il cliente ha un abbonamento attivo (o si è verificata l'impossibilità di aggiungerlo)

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole inserire un nuovo abbonamento pagato da un cliente
2. Il sistema legge le informazioni del cliente scelto dall'amministratore.
3. Se il cliente non ha un abbonamento attivo (data di scadenza maggiore della data odierna):
 - 3.1. L'amministratore inserisce tutte le informazioni richieste.
 - 3.2. L'amministratore avvia la procedura di registrazione dell'abbonamento
 - 3.3. Il sistema registra il nuovo abbonamento per il cliente.
4. *Altrimenti*
 - 4.1 L'amministratore non può inserire un nuovo abbonamento al posto di quello attivo.

Sequenza degli eventi alternativa

Dal punto 3.2

3.3 Il procedimento fallisce in quanto l'amministratore non ha completato tutti i campi richiesti.

3. CASI D'USO: VISUALIZZA ABBONAMENTO

Questo caso d'uso si verifica qualora l'amministratore voglia visualizzare le informazioni di un abbonamento.

Pre-condizioni

L'abbonamento esiste a sistema

Post-condizioni

Nessuna

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole visualizzare le informazioni di un abbonamento pagato da un cliente
2. Il sistema legge le informazioni del cliente scelto dall'amministratore
3. Se il cliente ha un abbonamento attivo (data di scadenza maggiore della data odierna):

3.1. Il sistema visualizza le informazioni dell'abbonamento 4. *Altrimenti*

4.1. Il sistema comunica che il cliente non ha un abbonamento attivo

Sequenza degli eventi alternativa

Nessuna

4. CASI D'USO: INSERISCI PRENOTAZIONE

Questo caso d'uso si verifica qualora l'amministratore voglia inserire una prenotazione di un cliente.

Pre-condizioni

Il cliente ha un abbonamento attivo

Post-condizioni

Il cliente ha una nuova prenotazione per un determinato servizio

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole inserire una nuova prenotazione

2. L'amministratore inserisce le informazioni della nuova prenotazione

3. L'amministratore avvia la procedura di inserimento nuova prenotazione, selezionando un servizio disponibile e un cliente con un abbonamento attivo

4. Il sistema registra con successo la nuova prenotazione per il cliente

Sequenza degli eventi alternativa

Dal punto 3.

4. L'amministratore non ha inserito tutte le informazioni necessarie

5. La procedura termina con errore

5. CASI D'USO: VISUALIZZA PRENOTAZIONE

Questo caso d'uso si verifica qualora l'amministratore voglia visualizzare le informazioni di una prenotazione a sistema.

Pre-condizioni

La prenotazione esiste a sistema

Post-condizioni

Nessuna

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole visualizzare le informazioni di una prenotazione
2. Il sistema legge le informazioni della prenotazione scelta dall'amministratore 3. Il sistema visualizza a schermo le informazioni della prenotazione

Sequenza degli eventi alternativa

Nessuna

6. CASI D'USO: DISDETTA PRENOTAZIONE

Questo caso d'uso si verifica qualora l'amministratore voglia disdire (eliminare) una prenotazione a sistema.

Pre-condizioni

La prenotazione esiste a sistema

Post-condizioni

La prenotazione non esiste più a sistema

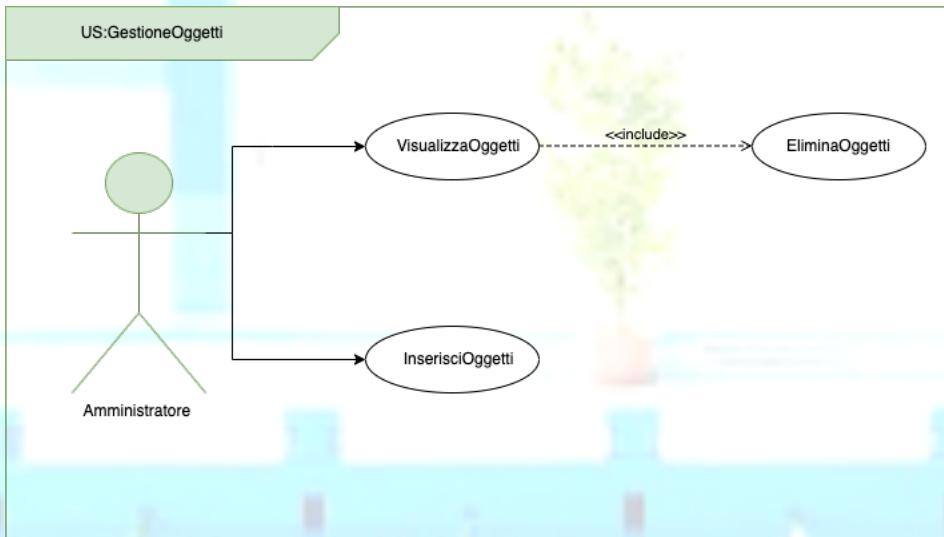
Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole eliminare una prenotazione
2. Il sistema legge le informazioni della prenotazione scelta dall'amministratore 3. Il sistema visualizza a schermo le informazioni della prenotazione
4. L'amministratore avvia la procedura di eliminazione
5. Il sistema elimina dal sistema le informazioni della prenotazione

Sequenza degli eventi alternativa

Nessuna

• CASI D'USO: GESTIONE OGGETTI



1. CASI D'USO: VISUALIZZA OGGETTI

Questo caso d'uso si verifica qualora l'amministratore voglia visualizzare le informazioni di un oggetto.

Pre-condizioni

L'oggetto esiste a sistema

Post-condizioni

Nessuna

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole visualizzare le informazioni di un oggetto
2. Il sistema legge le informazioni dell'oggetto scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni dell'oggetto

Sequenza degli eventi alternativa

Nessuna

2. CASI D'USO: ELIMINA OGGETTO

Questo caso d'uso si verifica qualora l'amministratore voglia eliminare un oggetto dal sistema.

Pre-condizioni

L'oggetto esiste a sistema

Post-condizioni

L'oggetto non esiste più a sistema

Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole eliminare le informazioni di un oggetto
2. Il sistema legge le informazioni dell'oggetto scelto dall'amministratore
3. Il sistema visualizza a schermo le informazioni dell'oggetto
4. L'amministratore avvia la procedura di eliminazione
5. Il sistema elimina dal sistema le informazioni dell'oggetto

Sequenza degli eventi alternativa

Nessuna

3. CASI D'USO: INSERISCI OGGETTO

Questo caso d'uso si verifica qualora l'amministratore voglia inserire un nuovo oggetto a sistema.

Pre-condizioni

L'oggetto non esiste a sistema

Post-condizioni

Il nuovo oggetto esiste a sistema (compilare tutti i campi richiesti)

Sequenza degli eventi principale

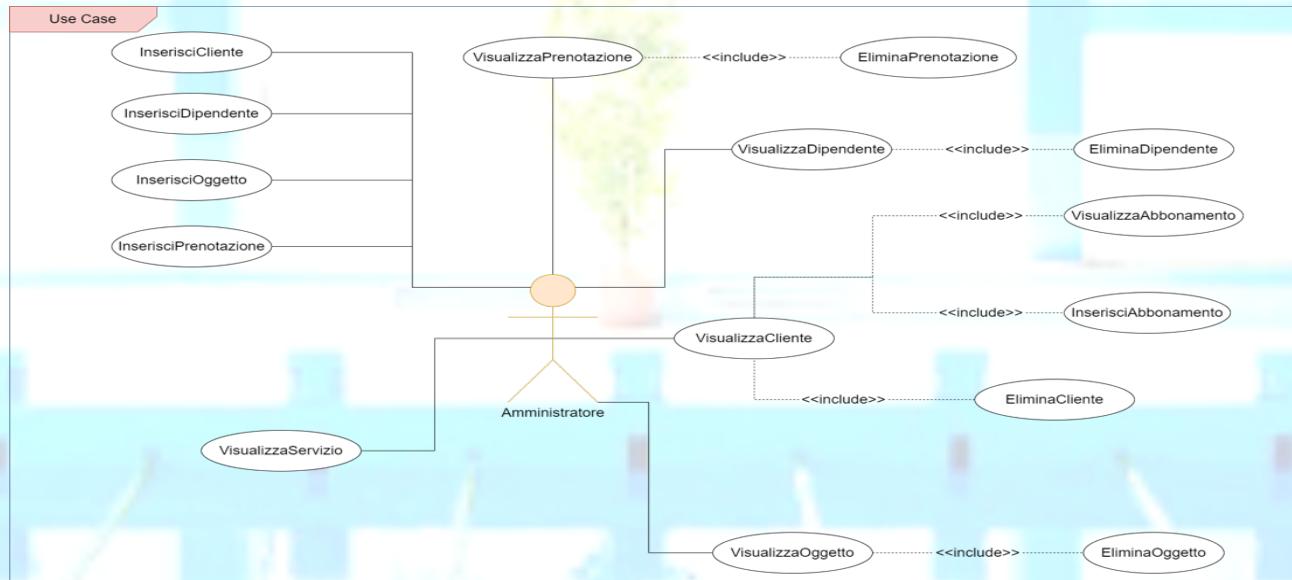
1. Il caso d'uso inizia quando l'amministratore dello stabilimento vuole inserire le informazioni di un nuovo oggetto
2. Il sistema visualizza la schermata di inserimento informazioni del nuovo oggetto.
4. L'amministratore avvia la procedura di inserimento nuovo oggetto
5. Il sistema aggiunge con successo il nuovo oggetto a sistema

Sequenza degli eventi alternativa

nessuna

VISUALIZZAZIONE CASI D'USO PER INTERO

Questo è il diagramma dei casi d'uso per intero, del nostro software. Abbiamo separato i diagrammi in modo di avere una visualizzazione più chiara.



Matrice di Mapping

Di seguito è riportata la matrice di mapping tra requisiti funzionali e casi d'uso. Questa matrice è importante per capire se ad ogni requisito è collegato almeno uno scenario, in modo tale che tutti i requisiti siano rispettati.

- **Gestione piscina**

	RF1 Visualizzazione cliente	RF2 Inserimento Cliente	RF3 Eliminazione Cliente	RF4 Visualizzazione Dipendete	RF5 Inserimento Dipendente	RF6 Eliminazione Dipendente
Visualizza cliente	X					
Visualizza dipendete				X		
Inserisci clienti		X				
Inserisci dipendente					X	
Elimina cliente	X		X			
Elimina dipendente						X
Inserisci abbonamento	X					

	RF7 Visualizzazione Servizio	RF8 Visualizzazione Abbonamento	RF9 Visualizza Prenotazione	RF10 Inserimento Abbonamento	R11 Inserimento Prenotazione	RF12 Elimina Prenotazione
Visualizza servizio	X					
Visualizza abbonamento		X				
Visualizza prenotazione			X			
Elimina prenotazione			X			X
Inserisci abbonamento				X		
Inserisci prenotazione					X	

	RF13 Visualizzazione oggetto	RF14 Inserimento oggetto	RF15 Emina oggetto
Visualizza oggetto	X		
Inserisci oggetto		X	
Elimina oggetto			X

Mappa dell'architettura

La mappa dell'architettura descrive come il programma interagisce con gli utenti. Per la nostra applicazione opteremo per un'architettura MVC (Model-View-Controller)

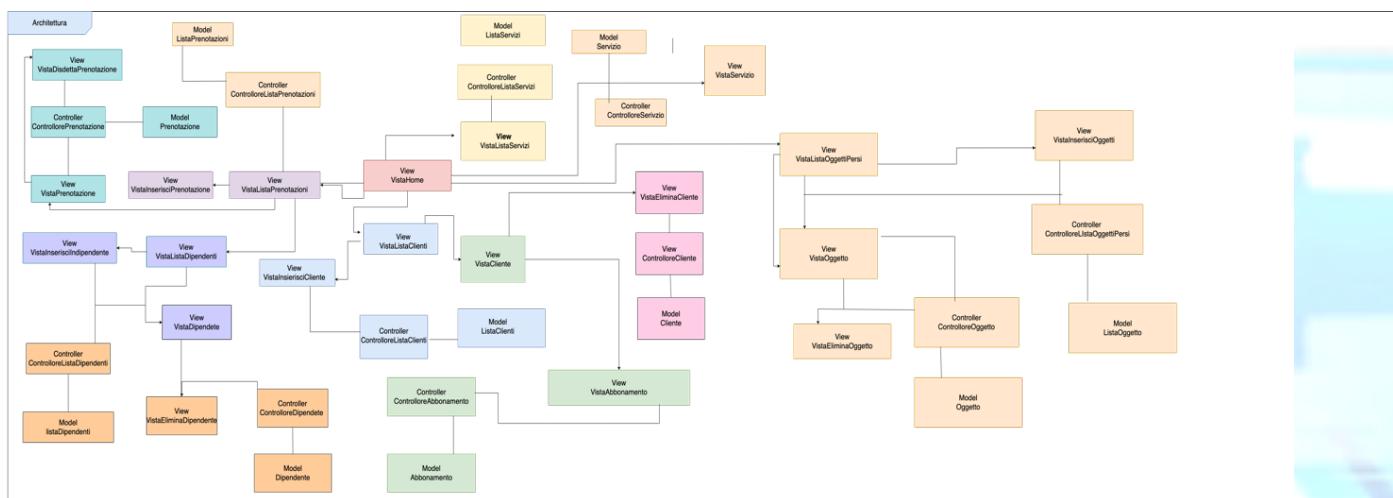
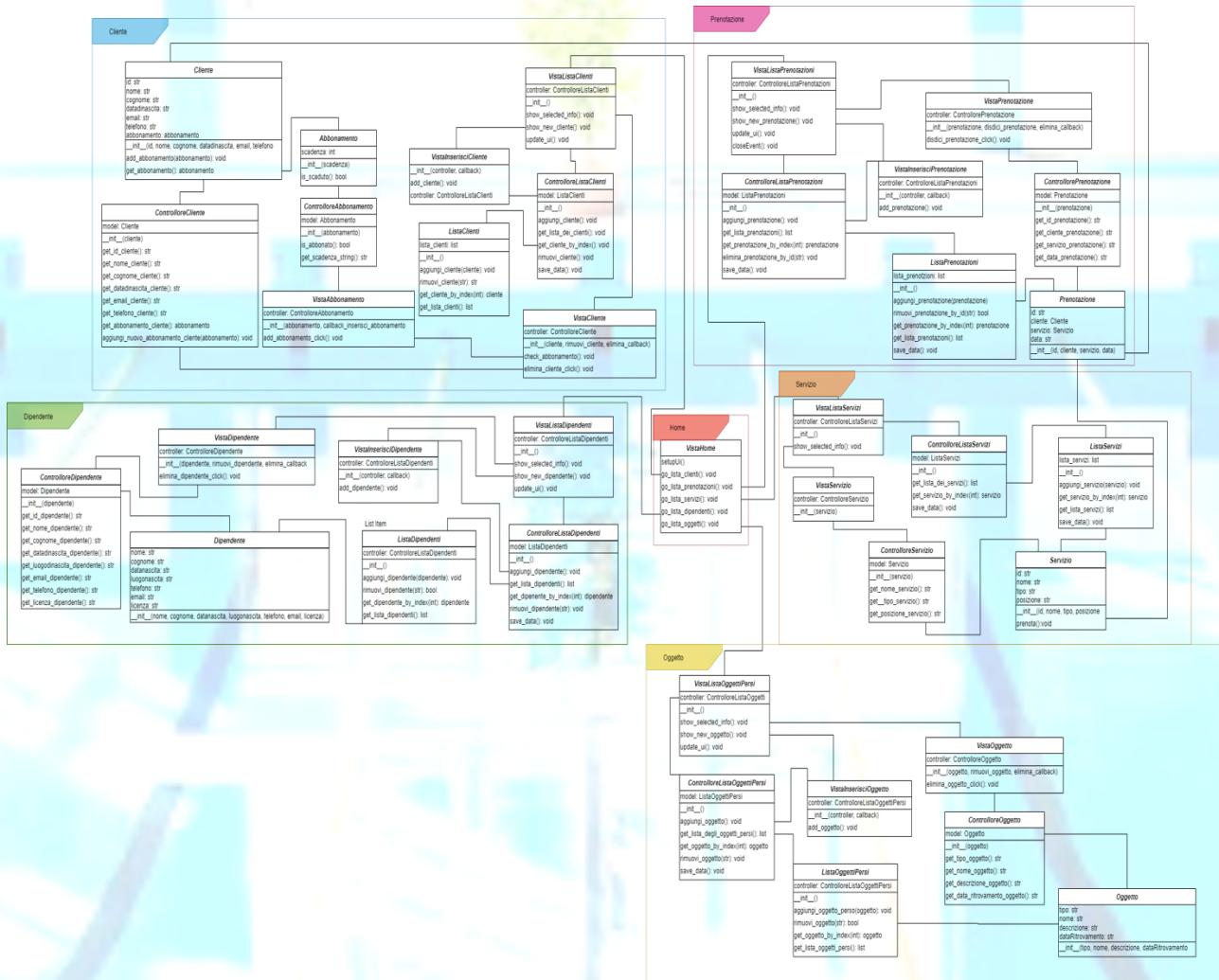
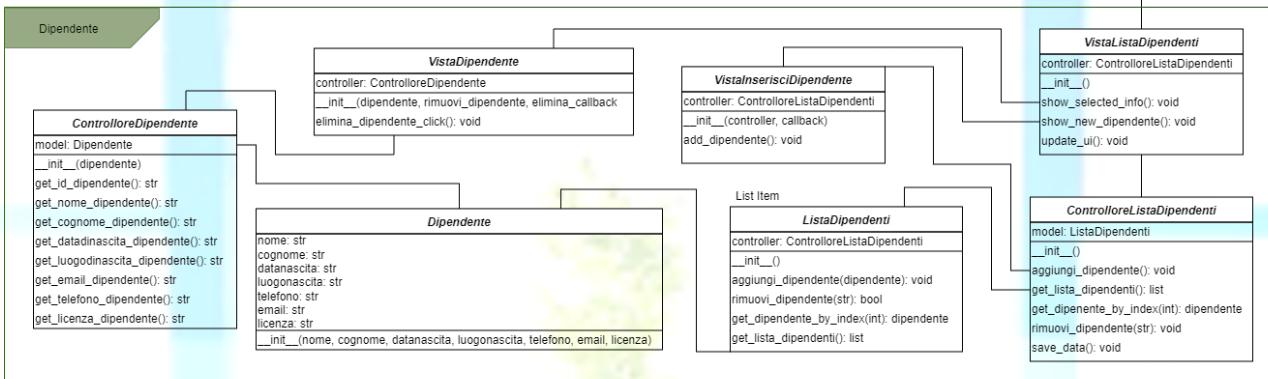


Diagramma delle classi

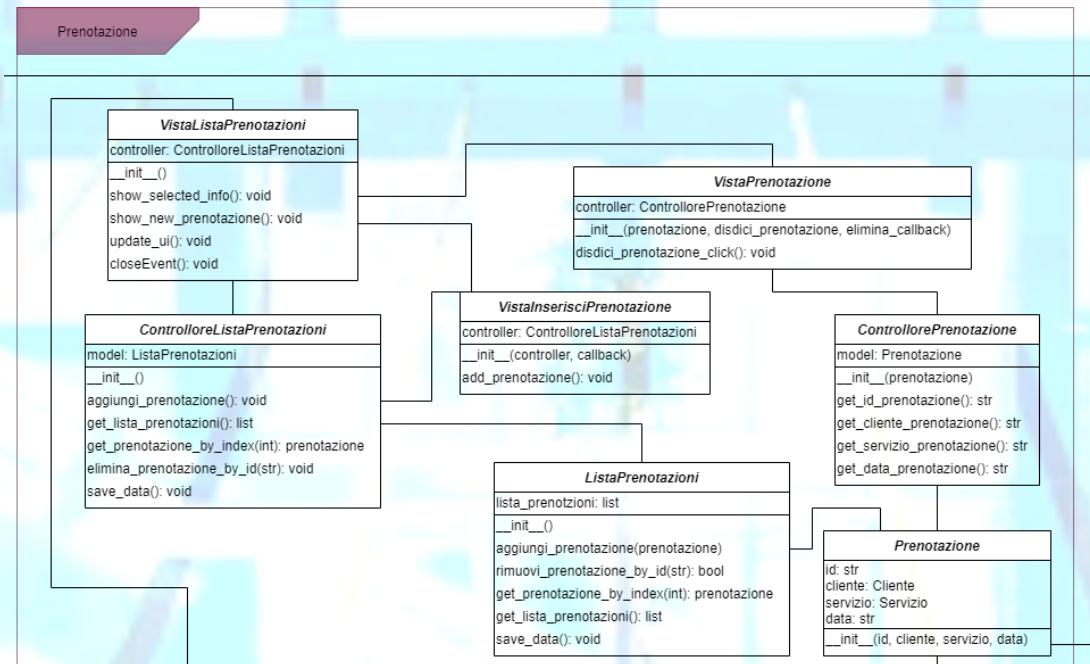
Il diagramma delle classi rappresenta l'intera struttura del software.



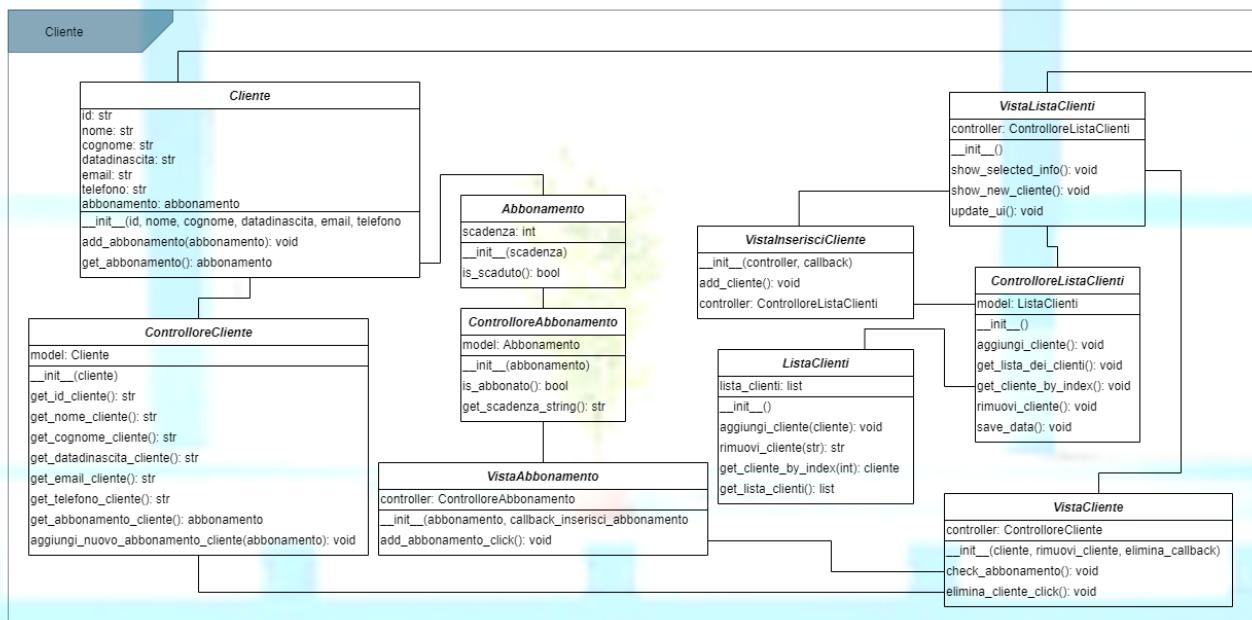
Dipendente:



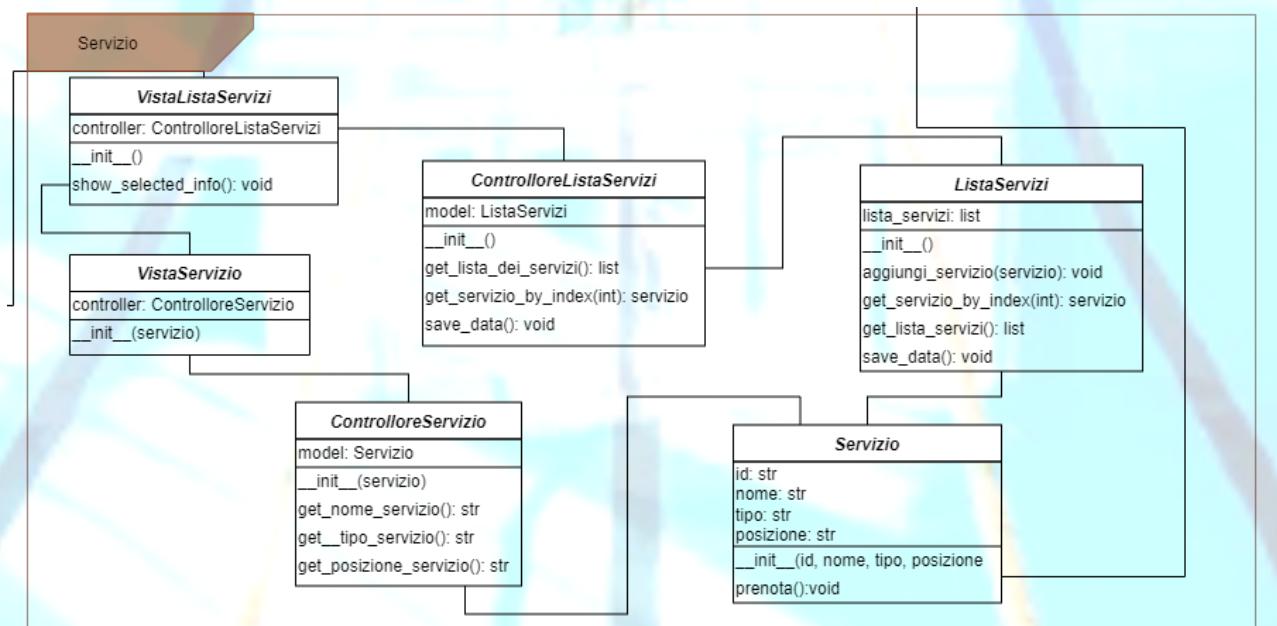
Prenotazione:



Cliente:



Servizio:



Oggetto:

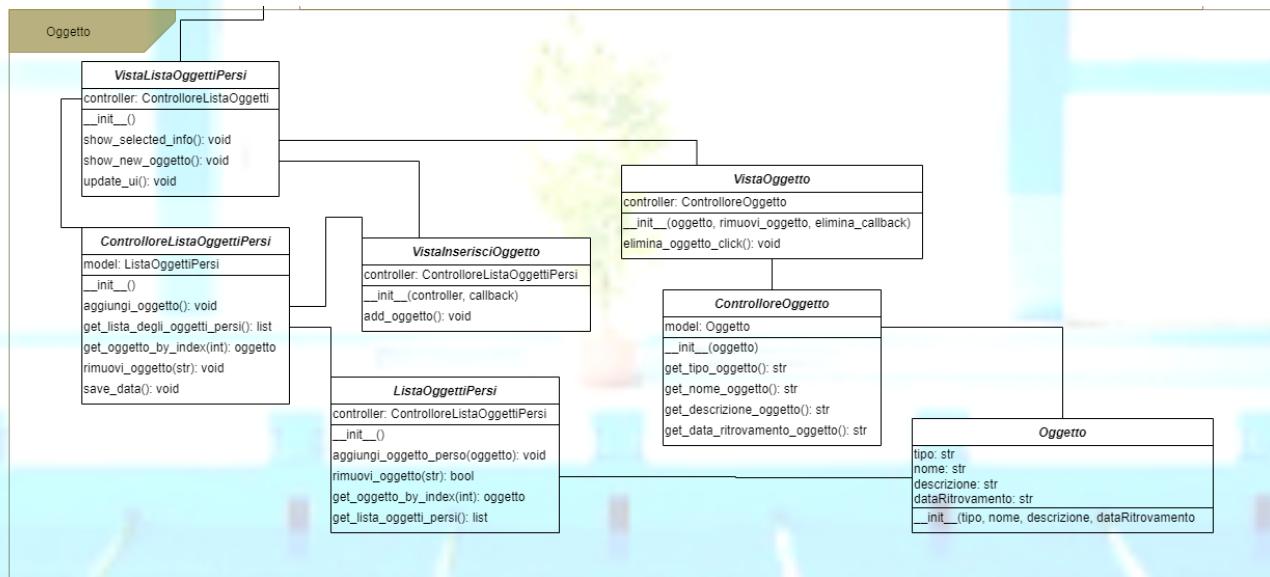
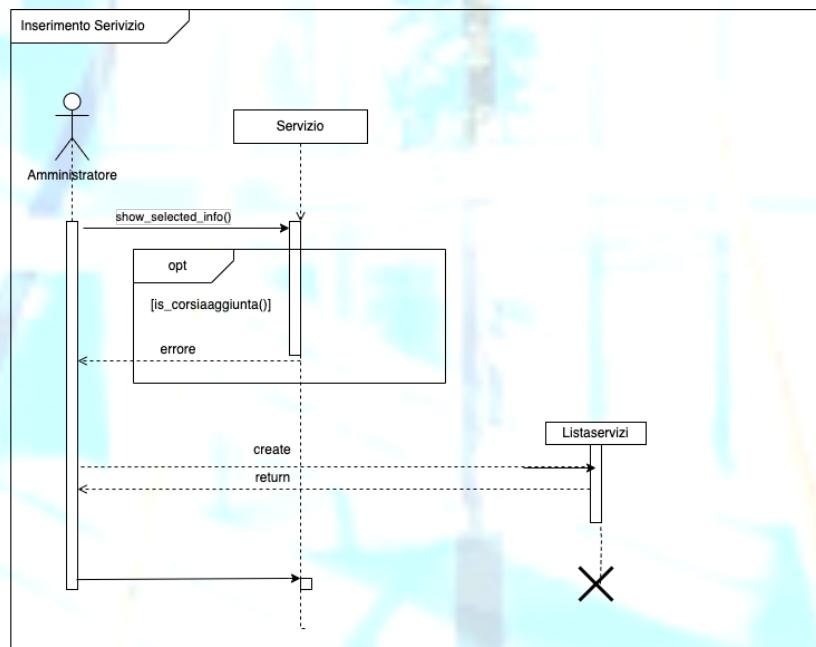
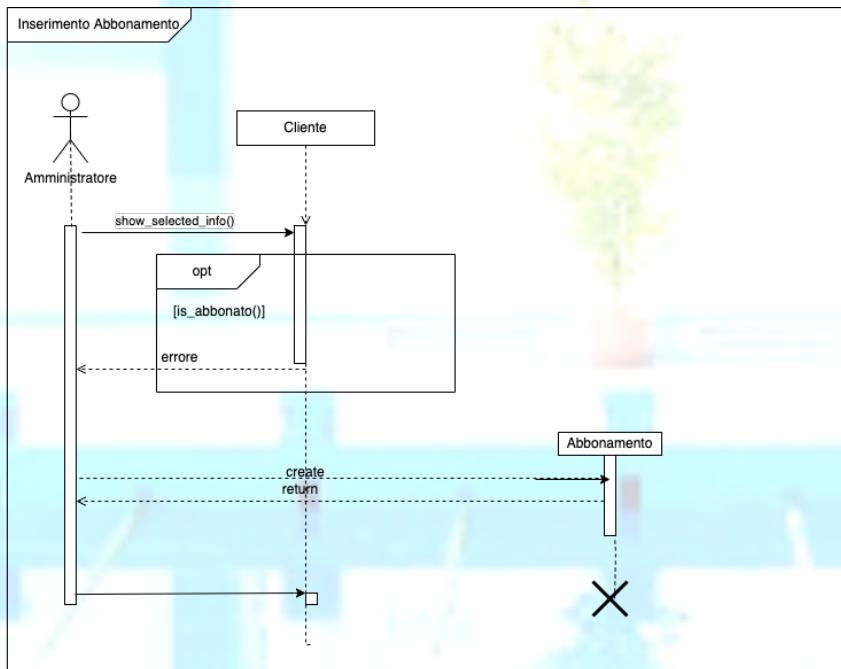


Diagramma di sequenza

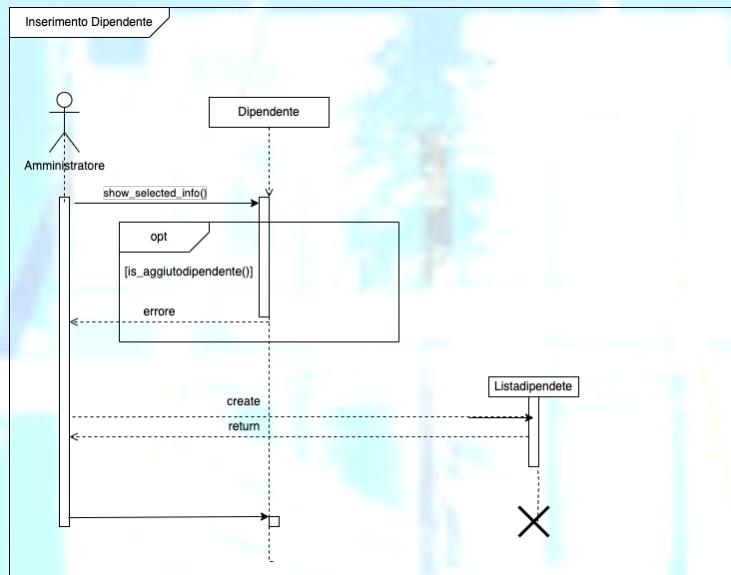
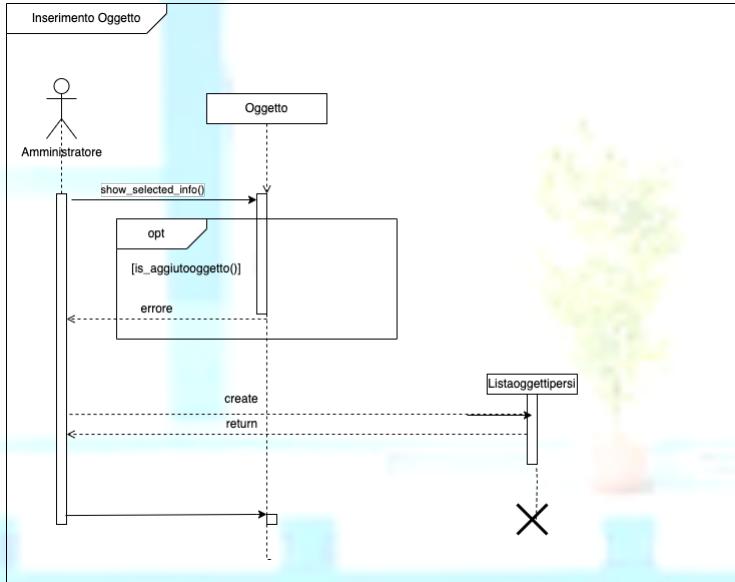
In una progettazione efficace è necessario presentare un diagramma di sequenza per scenario per rappresentare come interagiscono tra loro le diverse parti del software

Abbonamento:



Servizio:

Oggetto:

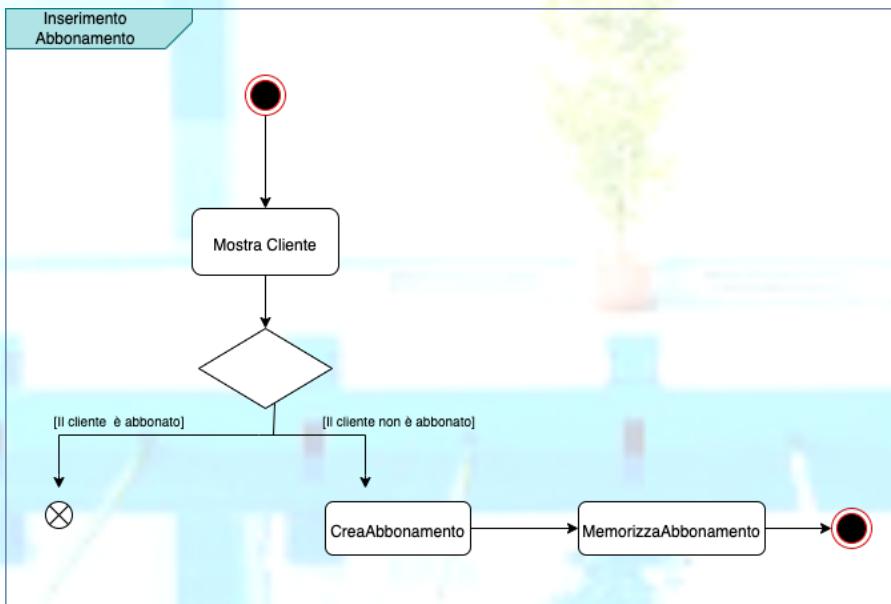


Dipendente:

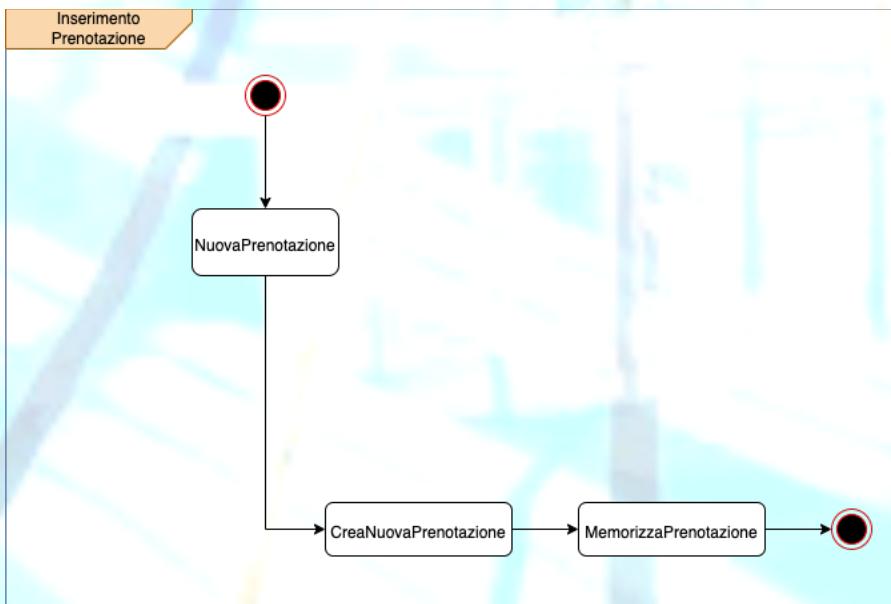
Diagramma di attività:

Come per i diagrammi di sequenza, per ogni scenario possibile nel vostro software è necessario fornire un diagramma di attività.

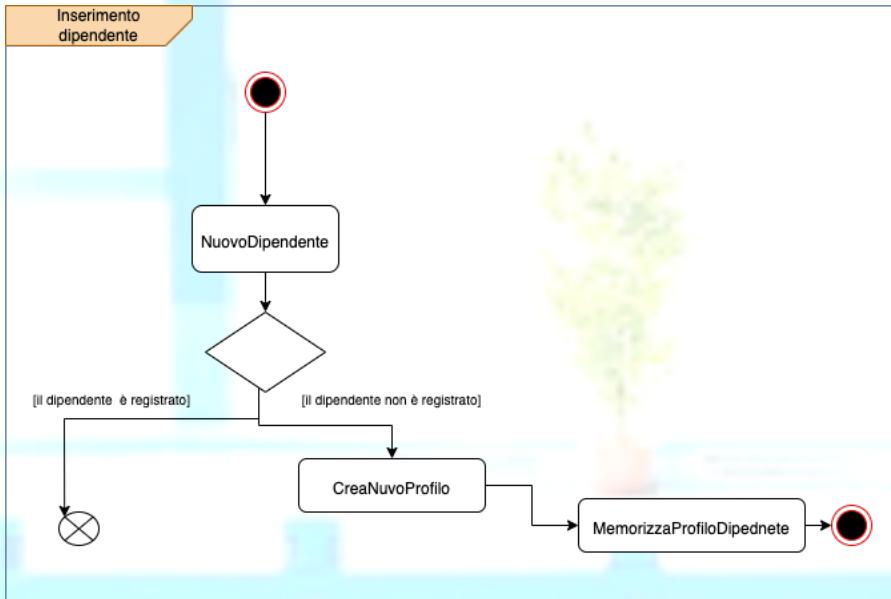
Abbonamento:



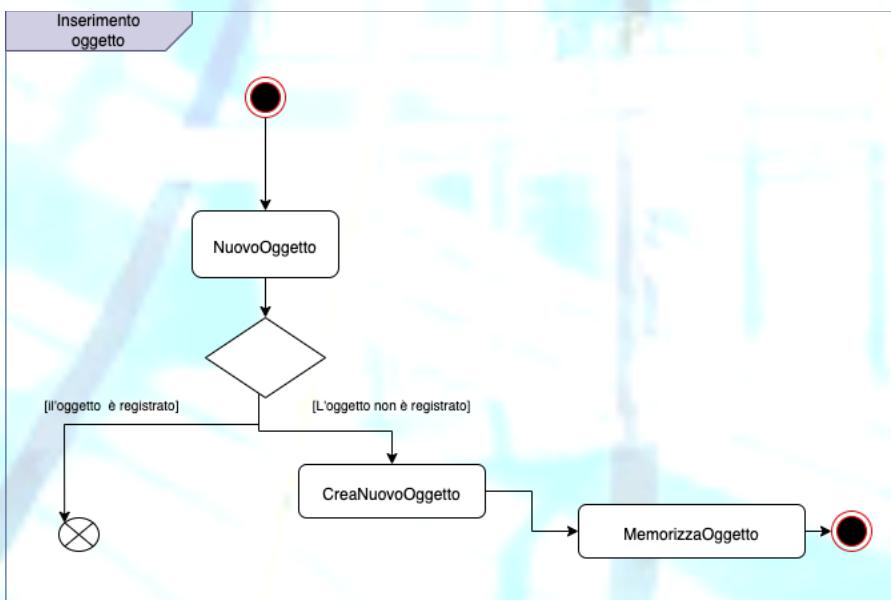
Prenotazione:



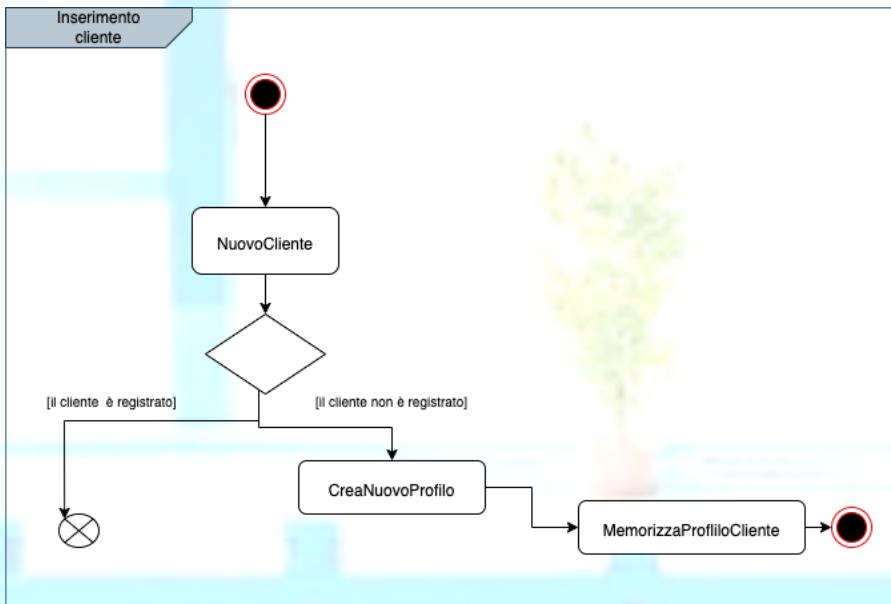
Dipendente:



Oggetto:



Cliente:



Unit Testing

Il test del software ha il compito di trovare i bug presenti nel codice. Per i tests è stato utilizzato lo Unit testing framework.

Test controllore lista dipendenti: effettuati per verificare la correttezza della sezione lista dipendenti.

```
7  class TestControlloreListaDipendenti(TestCase):
8      def setUp(self):
9          self.lista = ListaDipendenti()
```

Test aggiungi dipendente: serve per verificare un'aggiunta di un dipendente nella lista dei dipendenti.

```
11     def test_aggiungi_dipendente(self):
12         self.test_dip = Dipendente("Marisonia", "Ferrandino", "19/08/1999", "San Giovanni Rotondo", "s1093762@studenti.univpm.it", "1234567890", "segretaria")
13         self.lista.aggiungi_dipendente(self.test_dip)
14         self.assertIsNotNone(self.test_dip, "Non esiste")
```

Test controlla dipendente: serve a verificare se il dipendente esiste già nel sistema.

```
16     def test_controlla_dipendente(self):
17         self.test_dip1 = Dipendente("Marisonia", "Ferrandino", "19/08/1999", "San Giovanni Rotondo",
18                                     "s1093762@studenti.univpm.it", "1234567890", "segretaria")
19         self.test_dip2 = Dipendente("Marisonia", "Ferrandino", "19/08/1999", "San Giovanni Rotondo",
20                                     "s1093762@studenti.univpm.it", "1234567890", "segretaria")
21         self.assertNotEqual(self.test_dip1, self.test_dip2, "Dipendente già esistente")
```

Test lista vuota: verifica che la lista non sia vuota.

```
23     def test_lista_vuota(self):
24         self.test_dip3 = Dipendente("Marisonia", "Ferrandino", "19/08/1999", "San Giovanni Rotondo",
25                                     "s1093762@studenti.univpm.it", "1234567890", "segretaria")
26         self.lista.aggiungi_dipendente(self.test_dip3)
27         self.assertNotEmpty(self.lista)
```

Test elimina dipendente: funzione che serve per verificare che un dipendente venga rimosso correttamente.

```
29     def test_elimina_dipendente(self):
30         self.test_dip4 = Dipendente("Marisonia", "Ferrandino", "19/08/1999", "San Giovanni Rotondo",
31                                     "s1093762@studenti.univpm.it", "1234567890", "segretaria")
32         self.lista.aggiungi_dipendente(self.test_dip4)
33         self.lista.rimuovi_dipendente("Marisonia")
34         self.assertEmpty(self.lista.get_lista_dipendenti())
```

Secondo Test controllore lista prenotazioni: effettuati per verificare la correttezza della sezione lista prenotazioni.

```
8  class TestControlloreListaPrenotazioni(TestCase):
9      def setUp(self):
10         self.lista = ListaPrenotazioni()
11         self.controller = ControlloreListaPrenotazioni()
```

Test aggiungi prenotazione: serve per verificare un'aggiunta di una prenotazione nella lista delle prenotazioni.

```
13     def test_aggiungi_prenotazione(self):
14         self.test_prenotaz = Prenotazione("mariorossi", "Mario Rossi", "Seconda Corsia", "12/12/2022")
15         self.lista.aggiungi_prenotazione(self.test_prenotaz)
16         self.assertIsNotNone(self.test_prenotaz)
```

Test lista vuota: verifica che la lista non sia vuota.

```
18     def test_lista_vuota(self):
19         self.test_prenotaz1 = Prenotazione("mariorossi", "Mario Rossi", "Seconda Corsia", "12/12/2022")
20         self.controller.aggiungi_prenotazione(self.test_prenotaz1)
21         self.assertNotEmpty(self.controller)
```

Test elimina prenotazione: funzione che serve per verificare che una prenotazione venga rimossa correttamente.

```
def test_elimina_prenotazione(self):
    self.test_prenotaz2 = Prenotazione("mariorossi", "Mario Rossi", "Seconda Corsia", "12/12/2022")
    print(
        "Prenotazione per " + self.test_prenotaz2.cliente + " in " + self.test_prenotaz2.servizio + " del " + self.test_prenotaz2.data + " è stata cancellata")
    self.controller.aggiungi_prenotazione(self.test_prenotaz2)
    self.controller.elimina_prenotazione_by_id(self.test_prenotaz2)
    self.assertEmpty(self.lista.get_lista_prenotazioni())
```