

Claudio Musmeci 1000056290
Progetto per Ingegneria del Software

Applicazione EasyRoom

Breve descrizione

EasyRoom è un'applicazione sviluppata nell'ambito del progetto per la materia Ingegneria del Software. Nel documento sono presenti le varie fasi dello sviluppo in accordo con le metodologie fornite nel corso. Oltre alla fase di ideazione, sono presenti cinque iterazioni che coprono tutti i casi d'uso identificati, inoltre è stata realizzata una fase di testing mediante jUnit. I test sono automatizzati.

Il codice sorgente, i file UML in Astah e i vari documenti sono reperibili su GitHub:
<https://github.com/clamusO46002056/EasyRoom>

EasyRoom presenta un'interfaccia semplice basata su console e un sistema di persistenza dei dati basato sull'inserimento su file di testo. Il Sistema presenta due modalità operative: una per gli utenti e una per l'amministratore. L'utente può inserire delle prenotazioni e gestirle, modificandole o annullandole. L'Amministratore può inserire dei nuovi giocatori nel sistema; inserire prenotazioni e gestirle; inserire nuove stanze e modificarle; visualizzare prenotazioni; eseguire le operazioni di check-in e check-out; gestire il magazzino, visualizzando il numero di attrezzatura disponibile per un determinato giorno o modificando le quantità massime.

Documento di visione

Introduzione

EasyRoom è un software che ha come obiettivo la gestione di N stanze ludiche chiamate escape rooms. Esso si occuperà di fornire funzionalità relative alla prenotazione/gestione delle stanze e alla fornitura di eventuali strumenti e attrezzature.

Descrizione

L'obiettivo di EasyRoom è quello di implementare una gestione dinamica delle prenotazioni delle stanze, non solo per ridurre la complessità del lavoro e ridurre la percentuale di errore, ma anche per analizzare i dati in maniera da poter agire attivamente nel business per massimizzare i profitti.

Descrizione delle parti interessate

Le parti interessate sono essenzialmente due:

Amministratore di sistema: è colui che si occupa della gestione delle stanze e del magazzino.

Utenti/Giocatori: coloro che intendono sfruttare l'applicazione per prenotare la stanza ed eventualmente l'attrezzatura.

Riepilogo delle caratteristiche del sistema

EasyRoom ha come obiettivo quello di gestire:

1. Registrazione del giocatore
2. Prenotazione stanza
3. Gestione prenotazione
4. Modifica stanza
5. Storico prenotazioni stanza
6. Noleggio strumenti
7. Disponibilità strumenti

Glossario

Termine	Definizione
Escape Room	Stanza ludica in cui un gruppo di persone è rinchiuso e deve risolvere una serie di enigmi, rompicapi ed indovinelli. Oltre ai giocatori, è possibile che all'interno della stanza siano presenti degli attori.
Gioco	Insieme di azioni all'interno di una escape room volte alla risoluzione degli enigmi per trovare la via d'uscita entro un limite di tempo prestabilito.
Amministratore	Gestore delle stanze, che si occupa del buon andamento del gioco e che utilizza il software EasyRoom.
Utente (Giocatore)	Utente che partecipa attivamente al gioco e che utilizza l'applicazione per prenotare una stanza.
Attrezzatura	Insieme di oggetti facoltativi che possono essere utilizzati per migliorare l'esperienza di gioco. Ad esempio torce, fotocamere, blocco note.
Prenotazione	L'atto di richiedere in maniera esclusiva l'utilizzo di

una escape room. È caratterizzata da una data, un orario di inizio e di fine occupazione della stanza.

Modello dei casi d'uso

Requisiti

EasyRoom è un'applicazione software per la gestione di N stanze ludiche chiamate escape rooms appartenenti ad una certa associazione. Per poter giocare occorre prenotare una stanza. Il numero di giocatori è fissato a due. Affinché si possa realizzare una prenotazione è necessario che: tutti i giocatori siano registrati al sistema; vengano specificati data e orario di inizio/fine occupazione della stanza e che il pagamento venga eseguito contestualmente alla prenotazione. Una stanza può essere occupata per un massimo di un'ora e mezza da un gruppo di giocatori. Un utente può anche modificare o rimuovere una prenotazione in maniera gratuita fino a 24 ore prima del verificarsi dell'evento, altrimenti dovrà pagare una penale pari al 50% della spesa totale, questo si traduce in un rimborso effettivo della metà della spesa originale. L'amministratore, oltre alle funzioni destinate all'utente, può verificare lo storico delle prenotazioni di ciascuna stanza al fine di valutare eventuali modifiche al prezzo/tema. L'amministratore, per aumentare i profitti, potrebbe anche noleggiare ai giocatori degli strumenti utili come videocamere, torce elettriche o eventuali supporti per migliorare l'esperienza di gioco.

EasyRoom ha come obiettivo quello di gestire:

8. Registrazione del giocatore
9. Prenotazione stanza
10. Gestione prenotazione
11. Modifica stanza
12. Storico prenotazioni stanza
13. Noleggio strumenti
14. Disponibilità strumenti

Obiettivi e casi d'uso

Analizzando il testo riportato nel paragrafo precedente è possibile individuare gli attori principali a cui è destinato il sistema e i relativi obiettivi che intendono portare a termine. Questo permette di ricavare i casi d'uso principali.

Attore	Obiettivo	Casi d'uso
Amministratore	Gestire la registrazione dei giocatori all'interno del sistema.	UC1: Inserimento nuovo giocatore.
Utente (Giocatore)	Inserimento e pagamento di una prenotazione con eventuale richiesta di attrezzatura.	UC2: Inserimento e pagamento di una prenotazione.
Utente (Giocatore)	Modifica di una prenotazione con possibilità di rimozione.	UC3: Modifica/Annullamento della prenotazione.
Amministratore	Gestione della registrazione delle stanze.	UC4: Inserimento/Modifica delle stanze.
Amministratore	Gestione di tutte le prenotazioni delle stanze.	UC5: Visualizzazione prenotazioni.
Amministratore	Effettuare check-in.	UC6: Check-in stanza.

Amministratore	Effettuare check-out.	UC7: Check-out stanza.
Amministratore	Visualizzazione del numero di partite giocate in un determinata stanza.	UC8: Conteggio partite in una determinata stanza.
Amministratore	Gestione del magazzino per le attrezzature totali e disponibili.	UC9: Gestione del magazzino.

Casi d'uso

Tra tutti i casi d'uso individuati, si è scelto di descrivere in maniera dettagliata UC1 e UC2 in quanto figurano i due attori principali. Per i restanti casi d'uso verrà fornita una descrizione in formato breve.

UC1: Inserimento nuovo giocatore

Nome del caso d'uso	UC1: Inserimento nuovo giocatore.
Portata	Applicazione EasyRoom.
Livello	Obiettivo utente.
Attore primario	Amministratore.
Parti interessate e interessi	<ul style="list-style-type: none"> - Amministratore: vuole inserire un nuovo giocatore nel sistema; - Utente (Giocatore): vuole essere inserito nel sistema per poter prenotare una stanza/ fare parte di una prenotazione.
Pre-condizioni	Nessuna precondizione.
Garanzia di successo	Le informazioni relative all'Utente (Giocatore) sono inserite con successo nel Sistema.
Scenario principale di successo	<ol style="list-style-type: none"> 1. L'Amministratore vuole inserire un nuovo Utente (Giocatore) nel sistema; 2. L'Amministratore sceglie l'attività "Inserisci nuovo giocatore"; 3. L'Amministratore chiede i dati anagrafici del giocatore e li inserisce nel sistema; 4. L'Amministratore indica di avere finito.
Estensioni	<p>*a. In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.</p> <ol style="list-style-type: none"> 1. L'Amministratore riavvia il software e ripristina lo stato precedente del Sistema; 2. Il Sistema ripristina lo stato. <p>3a. L'Amministratore inserisce un Utente (Giocatore) già inserito nel Sistema.</p> <ol style="list-style-type: none"> 1. Il Sistema genera un messaggio di errore; 2. Il Sistema riporta l'Amministratore al menu principale.
Requisiti speciali	Nessun requisito speciale.
Elenco delle varianti tecnologiche e dei dati	Assenti.
Frequenza di ripetizioni	Legata alla registrazione di nuovi Utenti (Giocatori)
Varie	Assenti.

UC2: Inserimento e pagamento di una prenotazione

Nome del caso d'uso	UC2: Inserimento e pagamento di una prenotazione.
Portata	Applicazione EasyRoom.
Livello	Obiettivo utente.
Attore primario	Utente (Giocatore).
Parti interessate e interessi	<ul style="list-style-type: none"> - Utente (Giocatore): vuole inserire una nuova prenotazione.
Pre-condizioni	<p>L'Utente che effettua la prenotazione deve essere registrato nel Sistema.</p> <p>L'Utente deve essere in possesso del denaro necessario per il pagamento della prenotazione.</p>
Garanzia di successo	La prenotazione della stanza è stata registrata con successo nel Sistema.
Scenario principale di successo	<ol style="list-style-type: none"> 1. L'Utente (Giocatore) vuole inserire una nuova prenotazione per occupare una stanza. 2. L'Utente (Giocatore) sceglie il comando "Inserimento nuova prenotazione". 3. L'Utente (Giocatore) inserisce il giorno, la fascia oraria e la stanza che vuole occupare. 4. L'Utente (Giocatore) inserisce i dati dell'altro giocatore. 5. L'Utente (Giocatore) sceglie se vuole aggiungere il noleggio dell'attrezzatura specificando di cosa ha bisogno con la relativa quantità. Il Sistema verifica e aggiorna le quantità disponibili. 6. Il Sistema genera il prezzo totale della partita. 7. L'Utente (Giocatore) procede al pagamento.
Estensioni	<p>*a. In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.</p> <ol style="list-style-type: none"> 1. L'Utente (Giocatore) riavvia il software e ripristina lo stato precedente del Sistema; 2. Il Sistema ripristina lo stato. <p>3a. La stanza scelta nel giorno e nella fascia inserita è già occupata.</p> <ol style="list-style-type: none"> 1. Il Sistema genera un messaggio in cui specifica che la stanza è occupata. 2. Il Sistema chiede all'Utente (Giocatore) di selezionare un'altra stanza o di prenotare in un altro giorno/un'altra fascia oraria. <p>3b. L'Utente prova a prenotare per più di 90 minuti consecutivi la stessa stanza.</p> <ol style="list-style-type: none"> 1. Il Sistema genera un messaggio in cui viene specificato che una stanza può essere occupata per un massimo di 90 minuti consecutivi. 2. Il Sistema chiede all'Utente (Giocatore) di

	<p>modificare la fascia oraria.</p> <p>4a. L'altro Utente (Giocatore ospite) inserito dall'Utente (Giocatore) non è registrato al Sistema.</p> <ol style="list-style-type: none"> 1. Il Sistema genera un messaggio in cui specifica chi non è registrato. 2. Il Sistema ritorna al menu. <p>5a. L'attrezzatura richiesta non è disponibile al noleggio.</p> <ol style="list-style-type: none"> 1. Il Sistema chiede all'Utente (Giocatore) se vuole ugualmente prenotare la stanza. 2. L'Utente (Giocatore) continua nella prenotazione o in alternativa ritorna al menu.
Requisiti speciali	Nessun requisito speciale.
Elenco delle varianti tecnologiche e dei dati	Assenti.
Frequenza di ripetizioni	Legata ad ogni prenotazione.
Varie	Assenti.

UC3: Modifica/Annullamento della prenotazione

L'Utente (Giocatore) vuole modificare/annullare una prenotazione effettuata in precedenza e registrata nel Sistema. Il Sistema, nel caso di annullamento, calcolerà il rimborso che è legato alla finestra temporale in cui avviene la procedura.

UC4: Inserimento/Modifica di una stanza

L'Amministratore del Sistema vuole modificare il nome/prezzo di una stanza o aggiungerne di nuove.

UC5: Visualizzazione prenotazioni

L'Amministratore del Sistema vuole visualizzare le informazioni relative alle prenotazioni delle stanze.

UC6: Check-in stanza

L'Amministratore del Sistema vuole eseguire il check-in per una prenotazione, legando di fatto la prenotazione all'utilizzo della stanza.

UC7: Check-out stanza

L'Amministratore del Sistema vuole eseguire il check-out per una prenotazione.

UC8: Conteggio partite in una determinata stanza

L'Amministratore del Sistema visualizza il numero di partite complessive effettuate in una specifica stanza.

UC9: Gestione del magazzino

L'Amministratore del Sistema vuole verificare la disponibilità delle attrezzature in magazzino per eventualmente affittarle in fase di prenotazione all'Utente (Giocatore).

Specifiche Supplementari

Usabilità

L'interazione con il sistema non deve presentare un elevato grado di complessità.

Affidabilità

Il software sviluppato deve essere affidabile e deve poter mantenere i propri dati anche in caso di guasti (guasti elettrici, usura dell'hardware, attacchi informatici).

Vincoli hardware e software

Per eseguire il software non ci sono particolari requisiti per il sistema operativo purchè sia presente la Java Virtual Machine.

Vincoli di sviluppo del software

Tutto il software è scritto usando il linguaggio Java e sfrutta dei file di testo .txt per gestire la persistenza dei dati. L'ambiente di sviluppo utilizzato è IntelliJ.

Aspetti legali

Le tecnologie utilizzate per la progettazione e realizzazione del sistema proposto sono di tipo open source o freeware.

Regole di dominio

ID	Regola	Modificabilità	Sorgente
R1	La richiesta del noleggio dell'attrezzatura ha un costo variabile, calcolato in base agli strumenti utilizzati.	Media, l'Amministratore può modificare i prezzi a seconda della richiesta.	Politica interna dell'applicazione.
R2	Una prenotazione rimossa entro 24h dall'evento implica una penale pari al 50% della spesa totale, il che si traduce in un rimborso pari al 50% della spesa.	Bassa.	Politica interna dell'applicazione.
R3	Le stanze hanno dei prezzi fissati.	Alta, l'Amministratore può modificare i prezzi in base ai dati raccolti.	Politica interna dell'applicazione.
R4	Il numero di Utenti (Giocatori) massimo è pari a due.	Bassa.	Politica di utilizzo delle stanze.
R5	Una partita ha durata massima di 90 minuti.	Media, sulla base di alcuni dati l'Amministratore potrebbe modificare la durata massima.	Politica di utilizzo delle stanze.

Elaborazione – Iterazione 1

L'obiettivo dell'Iterazione 1 è implementare lo scenario principale di successo e tutte le estensioni individuate nel caso d'uso UC1: "Inserimento nuovo Giocatore(Utente)" nella fase di ideazione.

Analisi Orientata agli Oggetti

L'analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti. Vengono utilizzati diversi strumenti per fornire tale descrizione: Modello di Dominio, SSD, e Contratti delle operazioni.

Modello di Dominio

Per il caso d'uso scelto UC1 sono stati identificati le seguenti classi concettuali:

- EasyRoom: rappresenta l'applicazione;
- Amministratore: rappresenta l'Amministratore di Sistema, l'attore primario di questo caso d'uso;
- Giocatore: utente che vuole utilizzare il Sistema per prenotare una stanza.

È stato ricavato il seguente Modello di Dominio:

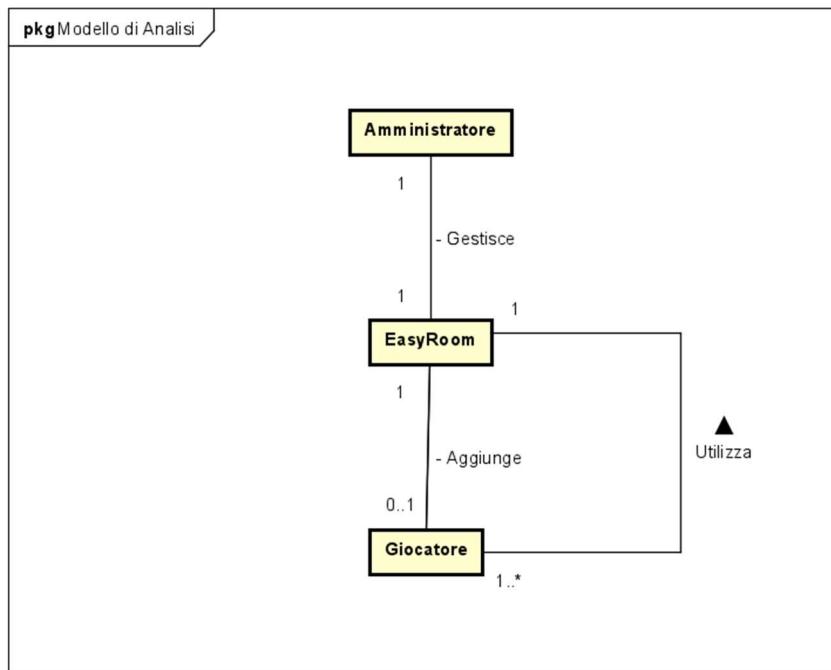
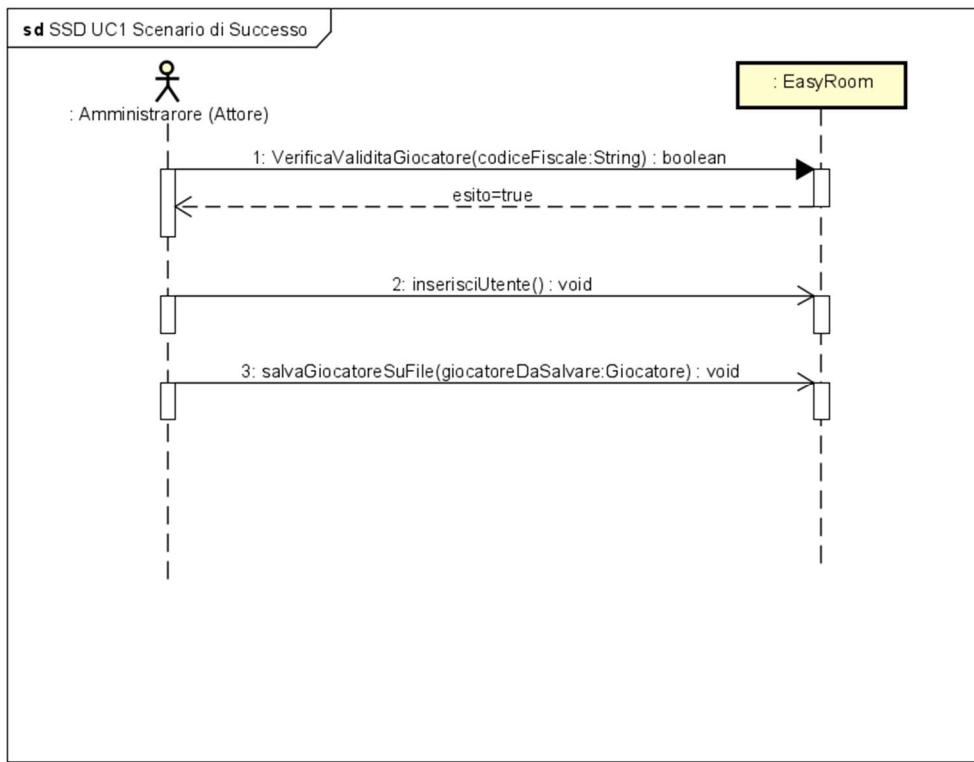


Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC1:



In questo diagramma, l'Amministratore chiede al sistema di verificare se il giocatore è presente o meno in memoria. Se il giocatore non è registrato nel sistema, l'amministratore inserisce i dati del giocatore nel Sistema e ne chiede la registrazione. In caso contrario, il Sistema segnala all'amministratore che il giocatore è presente in memoria e il processo di registrazione si interrompe.

Contratto delle operazioni

Di seguito viene indicato il Contratto delle operazioni per l'UC1:

Operazione:	VerificaValiditaGiocatore(codiceFiscale).
Riferimenti:	Caso d'uso UC1: Registrazione nuovo Giocatore.
Pre-condizioni:	-
Post-condizioni:	È stato restituito un Messaggio di Verifica dal Sistema.

Operazione:	inserisciUtente (nome, cognome, codiceFiscale, email, dataNascita). I dati vengono chiesti in maniera interattiva dal sistema.
Riferimenti:	Caso d'uso UC1: Registrazione nuovo Giocatore.
Pre-condizioni:	Il Messaggio di Verifica ha dato esito positivo, cioè nel Sistema non esiste nessun giocatore (Utente) associato al codice fiscale inserito.
Post-condizioni:	<ol style="list-style-type: none"> È stata creata l'istanza nuovoGiocatore di Giocatore. Gli attributi nome, cognome, codiceFiscale, ,email, dataNascita di nuovoGiocatore sono stati inizializzati correttamente. nuovoGiocatore è stato associato a EasyRoom. Viene inserito in una lista

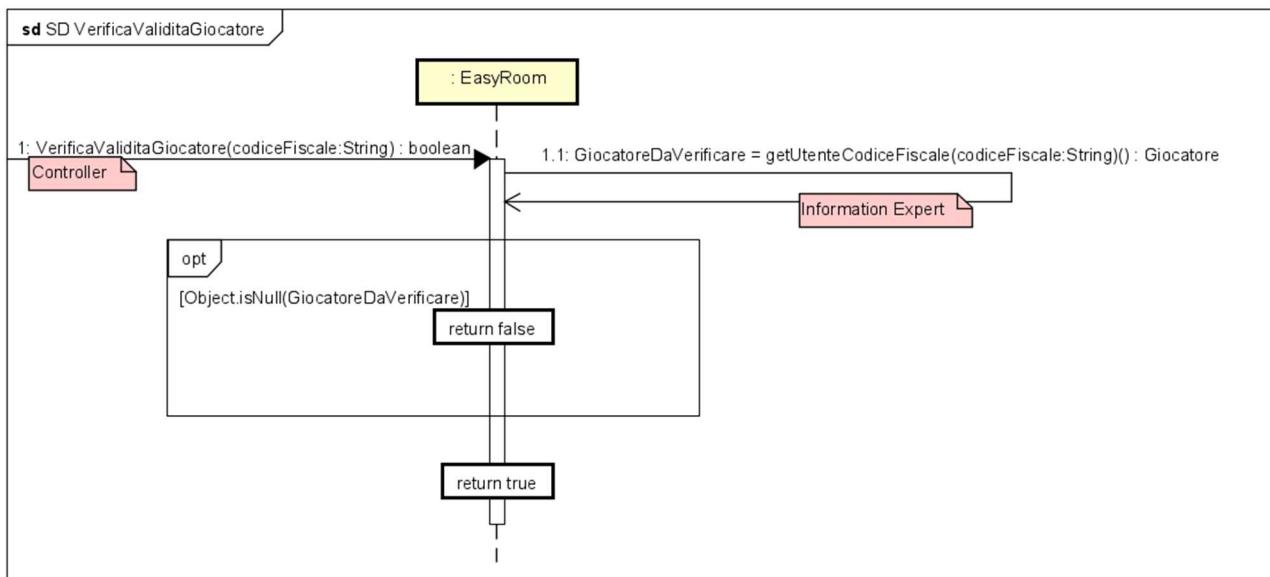
	mantenuta in memoria.
--	-----------------------

Operazione:	salvaGiocatoreSuFile(giocatoreDaSalvare)
Riferimenti:	Caso d'uso UC1: Registrazione nuovo giocatore.
Pre-condizioni:	È in corso l'inserimento di un nuovo Giocatore.
Post-condizioni:	giocatoreDaSalvare viene memorizzato in un file di testo che agisce da supporto per la persistenza dei dati.

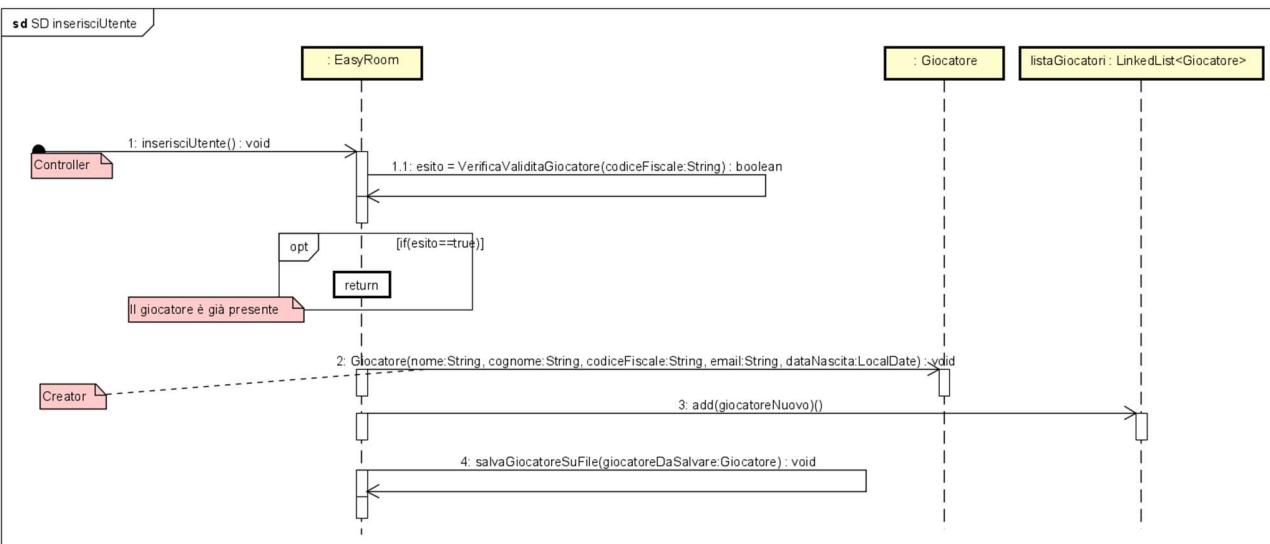
Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

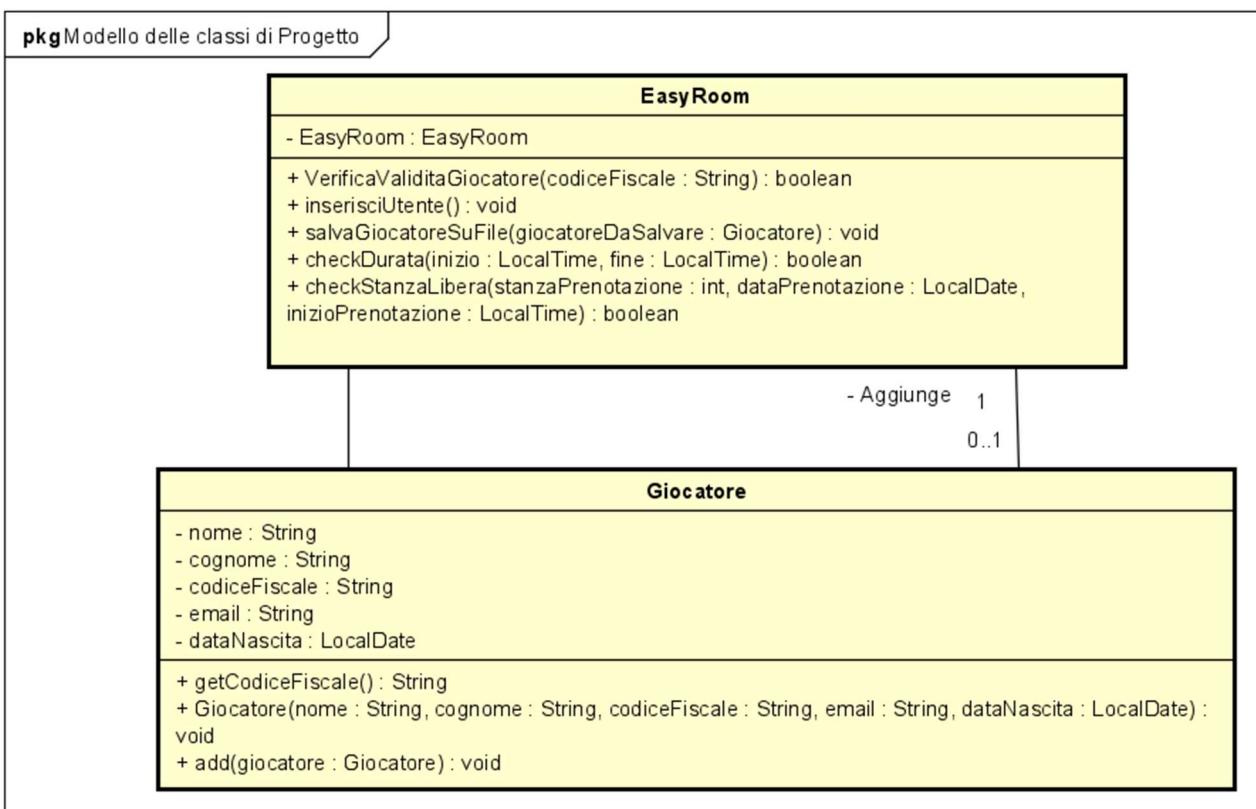
VerificaValiditaGiocatore(codiceFiscale : String)



inserisciUtente() : void



Modello delle classi di Progetto



Elaborazione – Iterazione 2

L'obiettivo dell'Iterazione 2 è implementare lo scenario principale di successo e tutte le estensioni individuate nel caso d'uso UC2: "Inserimento e pagamento di una prenotazione" nella fase di ideazione.

Analisi Orientata agli Oggetti

L'analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti. Vengono utilizzati diversi strumenti per fornire tale descrizione: Modello di Dominio, SSD, e Contratti delle operazioni.

Modello di Dominio

Per il caso d'uso scelto UC2 sono stati identificati le seguenti classi concettuali:

- EasyRoom: rappresenta l'applicazione;
- Giocatore: utente che vuole utilizzare il Sistema per prenotare una stanza, è l'attore primario di questo caso d'uso.
- Prenotazione: entità che rappresenta l'esclusiva sull'utilizzo di una stanza in un intervallo di tempo ben definito.
- Stanza: contiene le informazioni relative ad una escape room.
- Magazzino: contiene le informazioni e tiene traccia delle richieste effettuate.
- Attrezzatura: contiene la quantità di utilities richieste.

È stato ricavato il seguente Modello di Dominio:

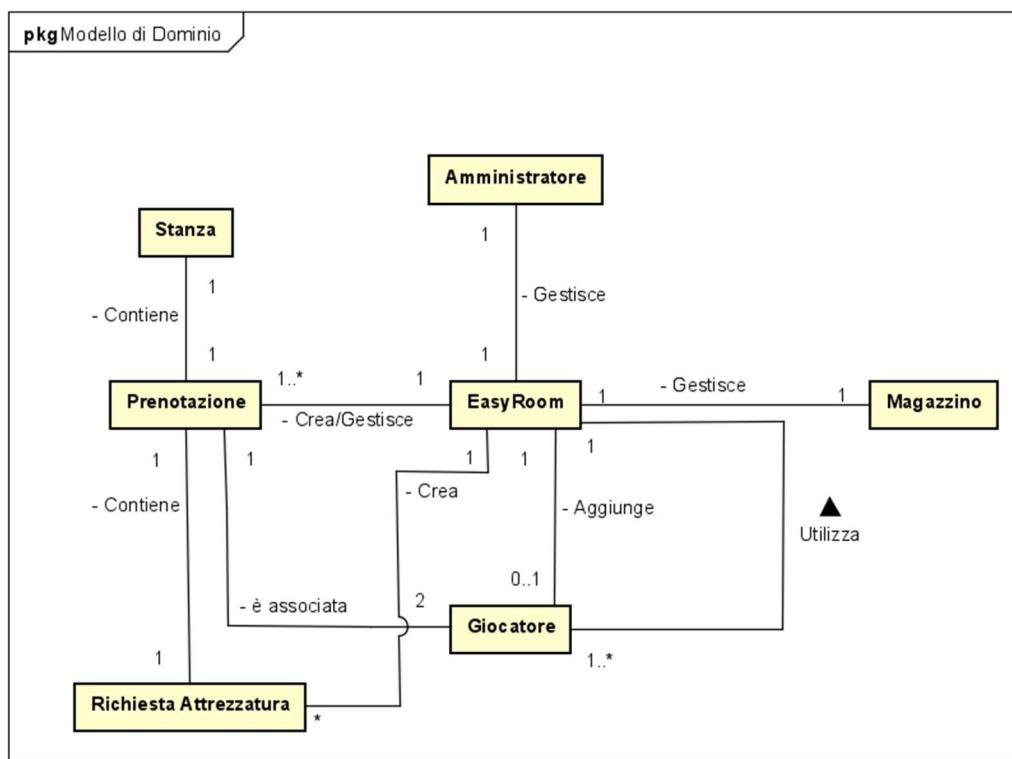
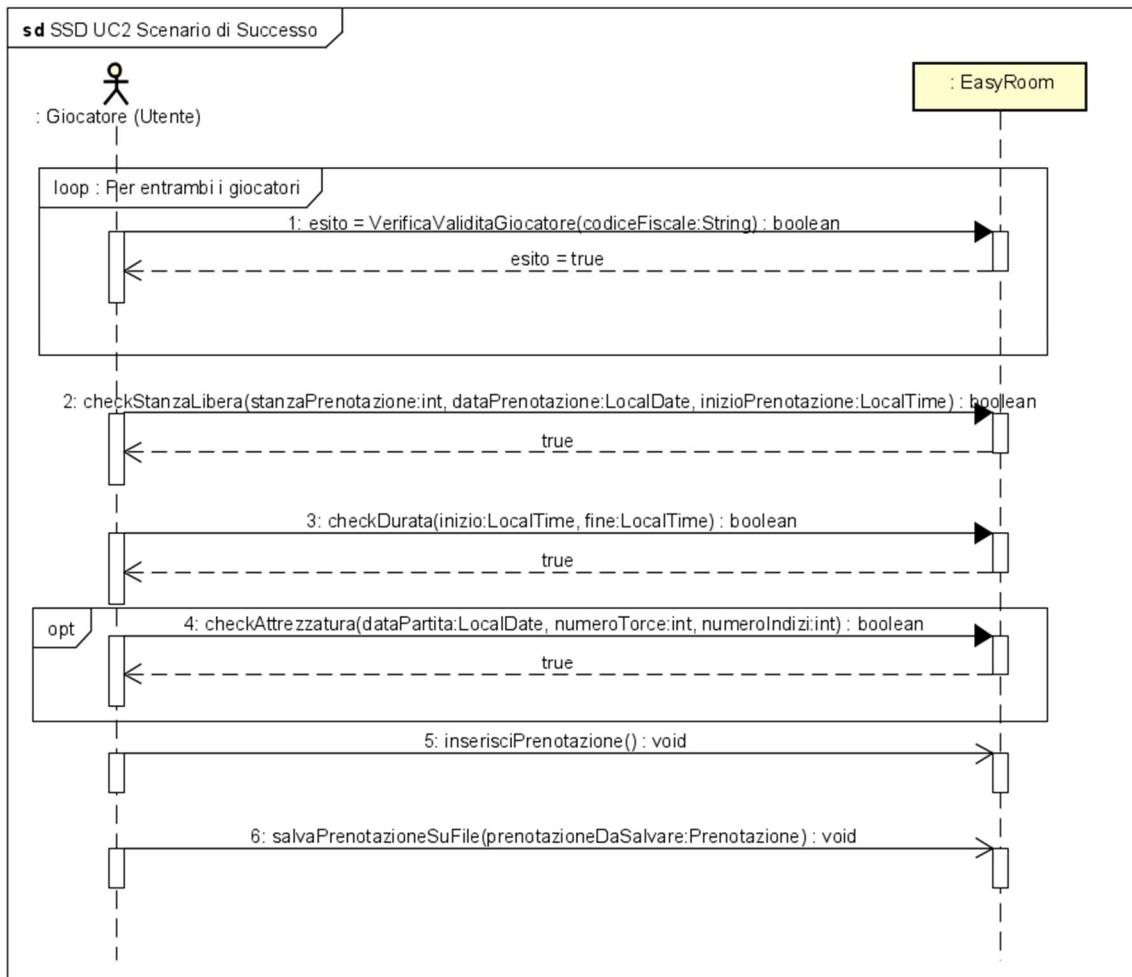


Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC2:



In questo diagramma, il giocatore(Utente) chiede al sistema di inserire una prenotazione. Il sistema verifica che i giocatori siano registrati, altrimenti stampa a schermo un messaggio di errore. Per poter prenotare una stanza è necessario che essa sia libera nell'intervallo di tempo scelto e che l'occupazione non superi un tempo limite, queste condizioni vengono verificate dal sistema con `checkStanzaLibera(...)` e `checkDurata(...)`, nel caso in cui esse non vengano rispettate, il sistema chiede all'utente se vuole continuare con la prenotazione o meno. La richiesta dell'attrezzatura è facoltativa, tuttavia viene eseguito un controllo della disponibilità nel magazzino. Infine, il sistema inserisce la prenotazione in memoria e per rendere il dato persistente lo memorizza in un file di testo.

Contratto delle operazioni

Di seguito viene indicato il Contratto delle operazioni per l'UC2:

Operazione:	<code>VerificaValiditaGiocatore(codiceFiscale)</code> .
Riferimenti:	Caso d'uso UC2: Inserimento nuova prenotazione.
Pre-condizioni:	-
Post-condizioni:	È stato restituito un Messaggio di Verifica dal Sistema. Viene restituito un booleano che da informazioni sull'esito.

Operazione:	<code>checkDurata(inizioPartita, finePartita)</code>
Riferimenti:	Caso d'uso UC2: Inserimento nuova prenotazione.
Pre-condizioni:	È in corso l'inserimento di una nuova prenotazione. Il Messaggio di Verifica ha dato esito positivo, cioè

	gli utenti immessi sono presenti nel Sistema.
Post-condizioni:	È stato restituito un Messaggio di Verifica dal Sistema. Viene restituito un booleano che da informazioni sull'esito.

Operazione:	checkStanzaLibera(stanzaPrenotazione, dataPrenotazione, inizioPrenotazione)
Riferimenti:	Caso d'uso UC2: Inserimento nuova prenotazione.
Pre-condizioni:	È in corso l'inserimento di una nuova prenotazione. Il Messaggio di Verifica ha dato esito positivo, cioè gli utenti immessi sono presenti nel Sistema.
Post-condizioni:	È stato restituito un Messaggio di Verifica dal Sistema. Viene restituito un booleano che da informazioni sull'esito.

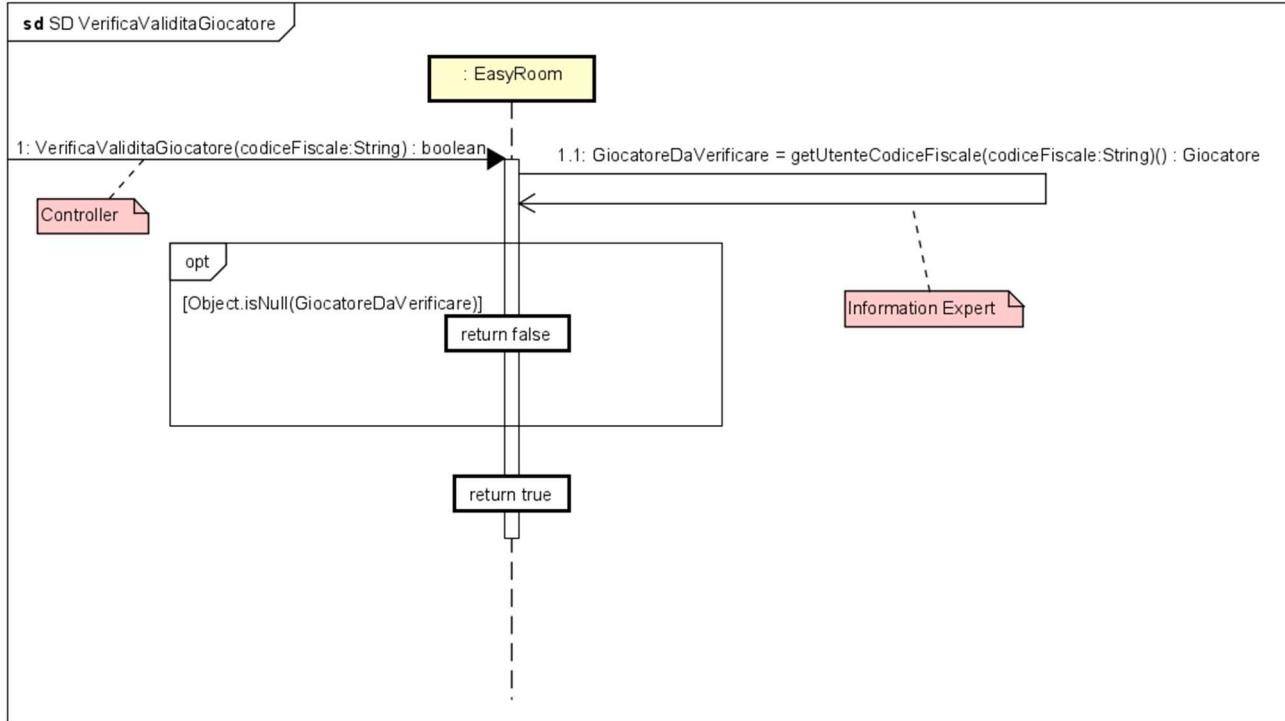
Operazione:	calcoloPrezzo(stanzaPrenotazione, attrezzaturaRichiesta, durata)
Riferimenti:	Caso d'uso UC2: Inserimento nuova prenotazione.
Pre-condizioni:	È in corso l'inserimento di una nuova prenotazione. I Messaggi di Verifica hanno dato esito positivo, cioè gli utenti immessi sono presenti nel Sistema, la stanza è disponibile e si è scelta l'attrezzatura.
Post-condizioni:	È stato calcolato un prezzo dal Sistema. Viene restituito un float che tiene conto del prezzo della stanza e dell'attrezzatura scelta.

Operazione:	inserisciPrenotazione () . I dati vengono chiesti in maniera interattiva dal sistema.
Riferimenti:	Caso d'uso UC2: Inserimento nuova prenotazione.
Pre-condizioni:	Il Messaggio di Verifica ha dato esito positivo, cioè nel Sistema non esiste nessun giocatore (Utente) associato al codice fiscale inserito.
Post-condizioni:	<p>4. È stata creata l'istanza nuovaPrenotazione di Giocatore.</p> <p>5. Gli attributi idPrenotazione, stanzaPrenotata, primoGiocatore, secondoGiocatore, dataPartita, tempoinizio, tempoFine, attrezzaturaRichiesta, costo di nuovaPrenotazione sono stati inizializzati correttamente con valori validi che rispettano le regole di dominio.</p> <p>nuovaPrenotazione è stato associato a EasyRoom. Viene inserito in una lista mantenuta in memoria.</p>

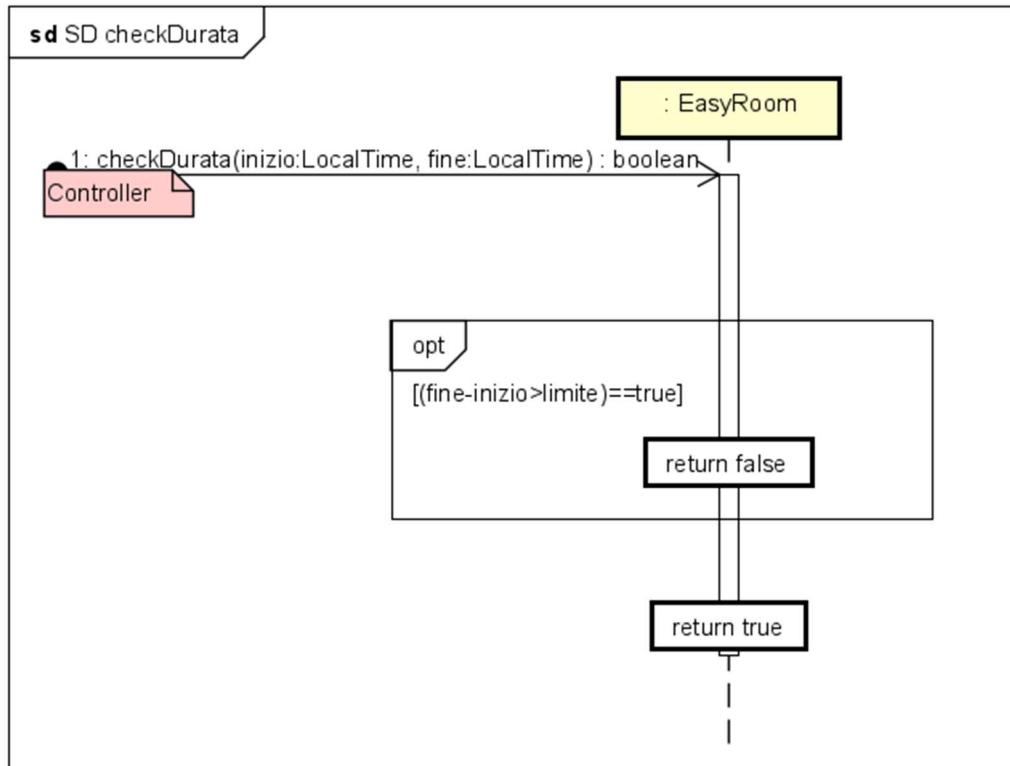
Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

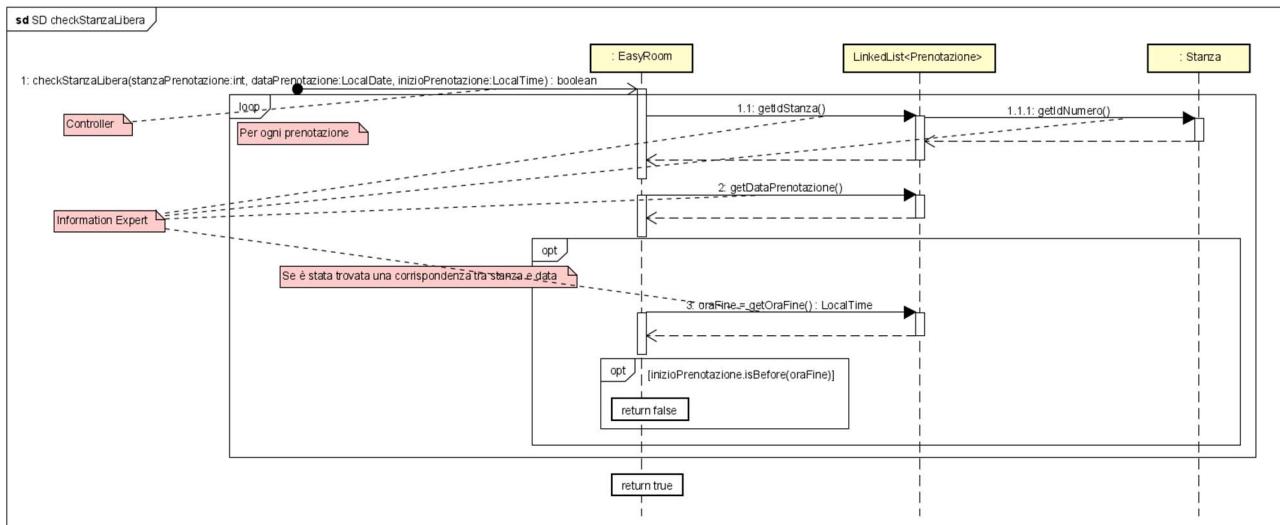
verificaValiditaGiocatore(codiceFiscale : String) : boolean



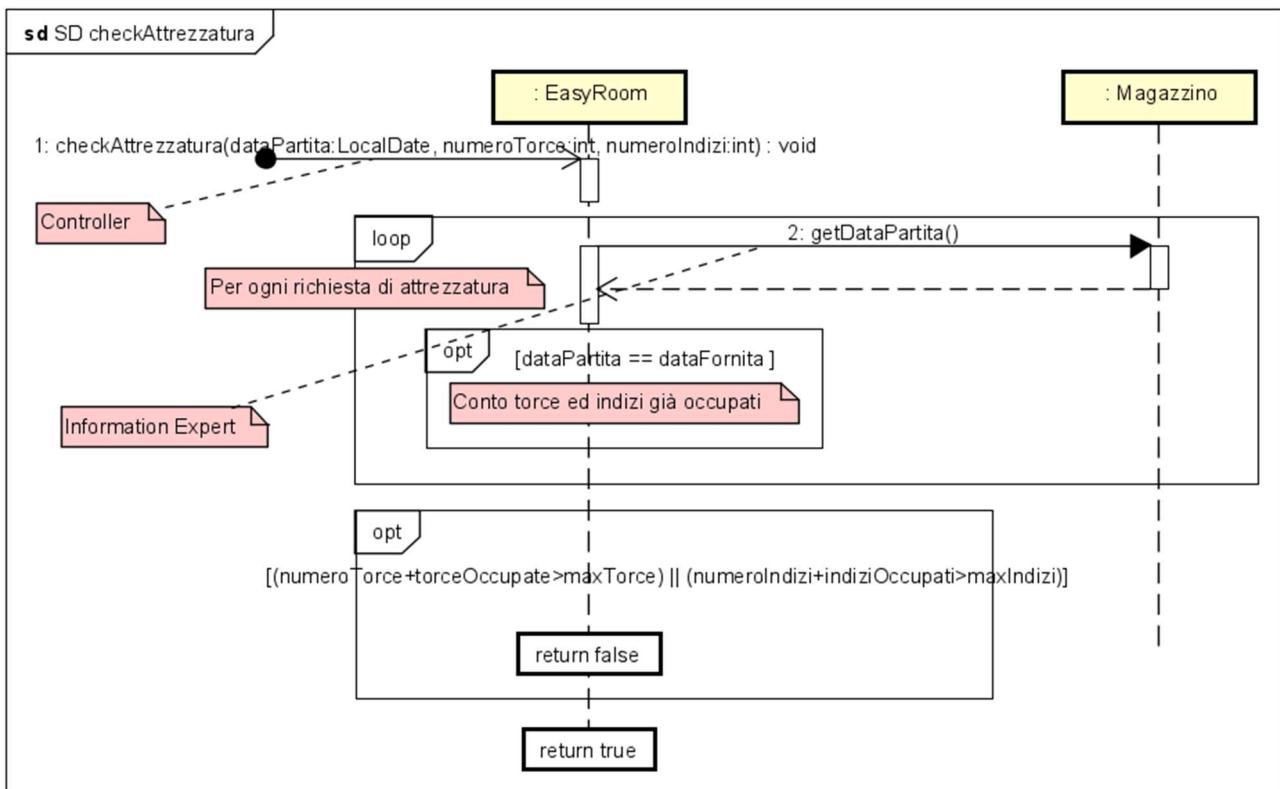
checkDurata(inizio : LocalTime, fine: LocalTime) : boolean



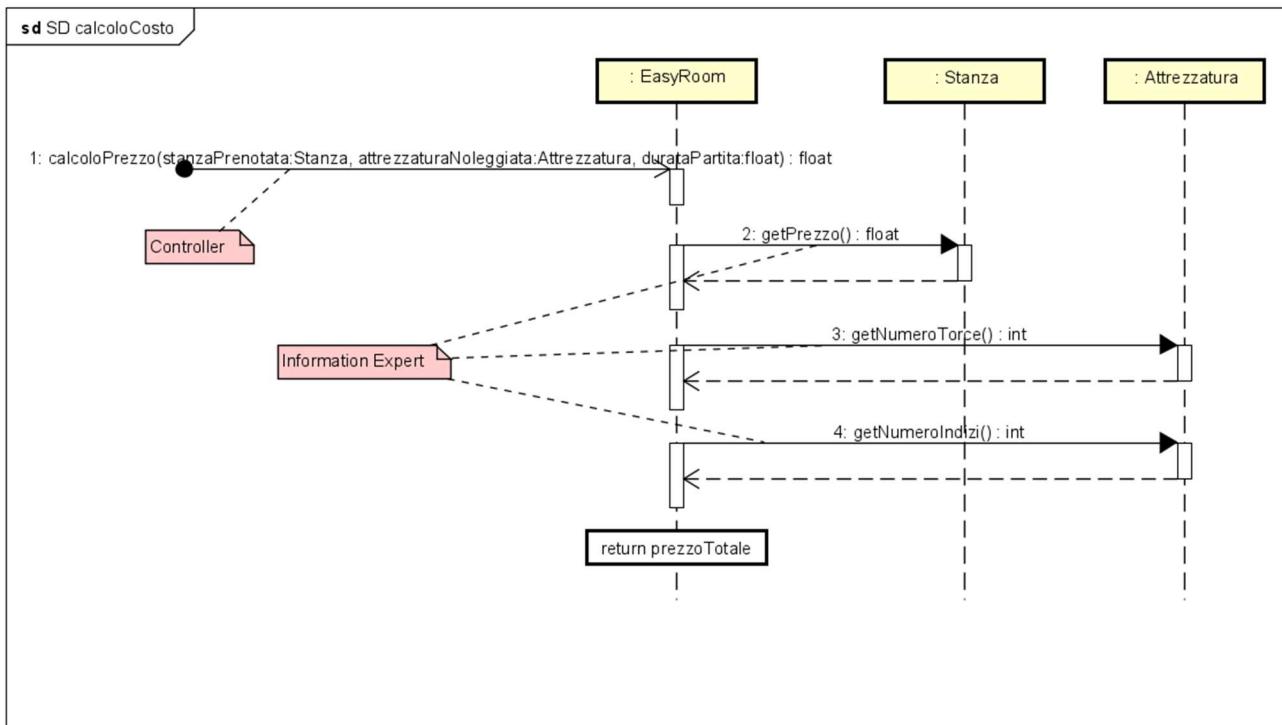
checkStanzaLibera(stanzaPrenotazione : int, dataPrenotazione : LocalData, inizioPrenotazione : LocalTime) : boolean



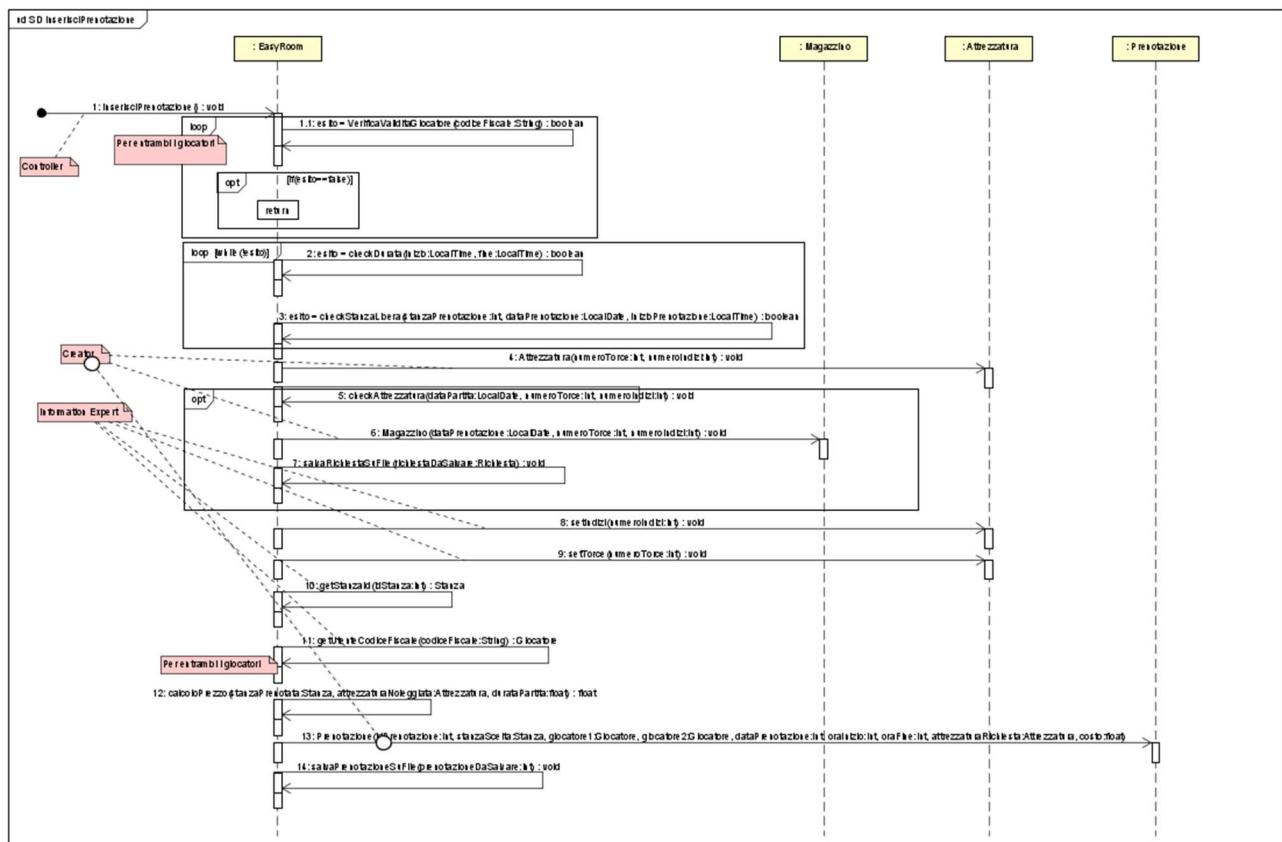
checkAttrezzatura(dataPartita : LocalDate, numeroTorce : int, numeroIndizi : int) : void



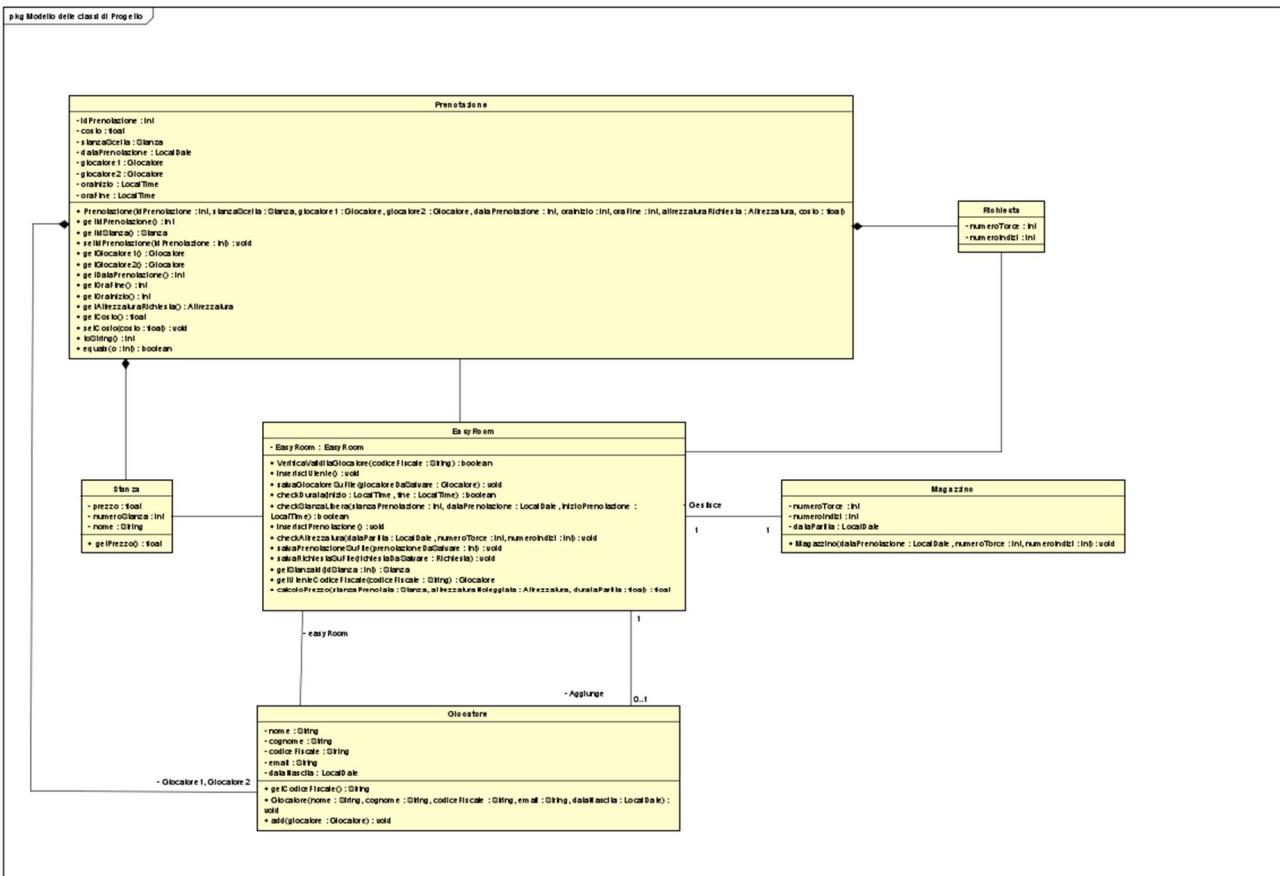
calcoloPrezzo(stanzaPrenotata : Stanza, attrezzaturaNoleggiata : Attrezzatura, durataPartita : float) : float



inserisciPrenotazione() : void



Modello delle classi di Progetto



Elaborazione – Iterazione 3 (Pattern GoF)

L’obiettivo dell’Iterazione 3 è implementare lo scenario principale di successo e tutte le estensioni individuate nel caso d’uso UC3: “Modifica/annullamento di una prenotazione” nella fase di ideazione. Un altro obiettivo è quello di implementare alcuni pattern GoF per rendere più robusta l’applicazione.

Pattern GoF Singleton

Lo scopo del pattern GoF Singleton è quello di garantire che una classe abbia una sola istanza e fornire un punto di accesso globale a quella istanza. Per fare ciò, viene implementato un metodo statico chiamato “getInstance()” che ritorna l’unica istanza, se questa non dovesse esistere allora verrebbe creata e restituita.

```
//Pattern GoF Singleton
1 usage
public static EasyRoom getInstance(){
    if (easyroom==null) easyroom = new EasyRoom();
    return easyroom;
}
```

Affinché possa funzionare, nella classe EasyRoom è stato definito un attributo “private static EasyRoom easyroom”. Nel main dell’applicazione viene invocato “getInstance()”:

```
EasyRoom applicazione = EasyRoom.getInstance(); //Pattern GoF Singleton
```

Pattern GoF Facade

Lo scopo del pattern GoF Facade è quello di fornire un’interfaccia unificata per semplificare l’uso di un sistema complesso nascondendo la complessità interna e fornendo una serie di punti di ingresso semplificati. All’interno del sistema, nella classe “Main” viene applicato il pattern Facade, essa contiene i metodi statici “applicazioneAdmin()” e “applicazioneUtente()”. In questo modo si ha una distinzione anche tra i ruoli degli attori che accedono al sistema.

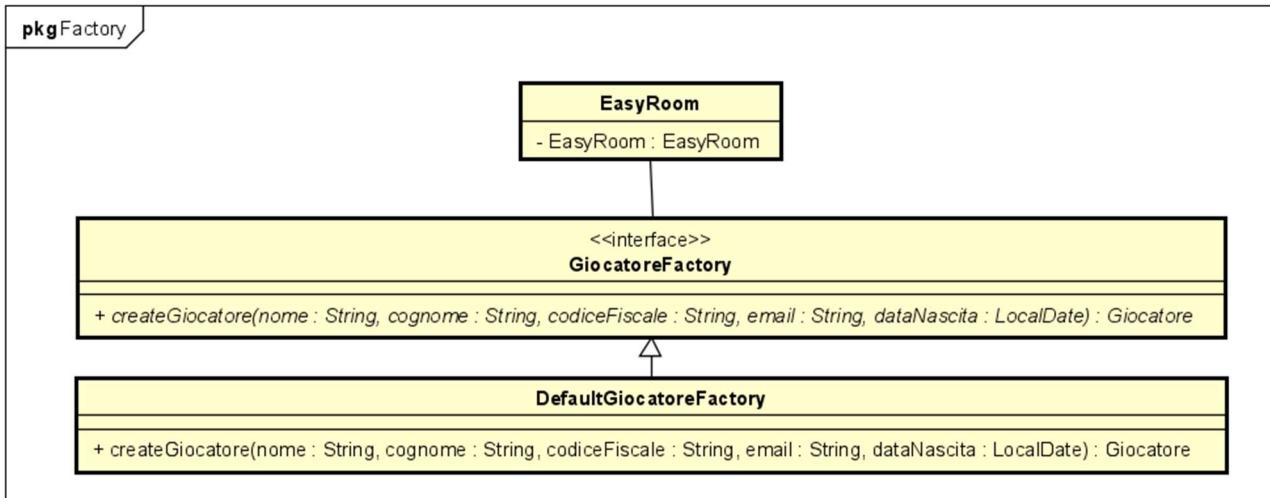
```
//Pattern Gof Facade
try{
    boolean admin = Boolean.parseBoolean(tastiera.readLine());
    if(admin){
        applicazioneAdmin(applicazione, tastiera);
    } else {
        applicazioneUtente(applicazione, tastiera);
    }
} catch (IOException e){
    System.out.println(e);
}
}

1 usage
public static void applicazioneAdmin(EasyRoom applicazione, BufferedReader tastiera){...}
1 usage
public static void applicazioneUtente(EasyRoom applicazione, BufferedReader tastiera){...}
```

Pattern GoF Factory Method

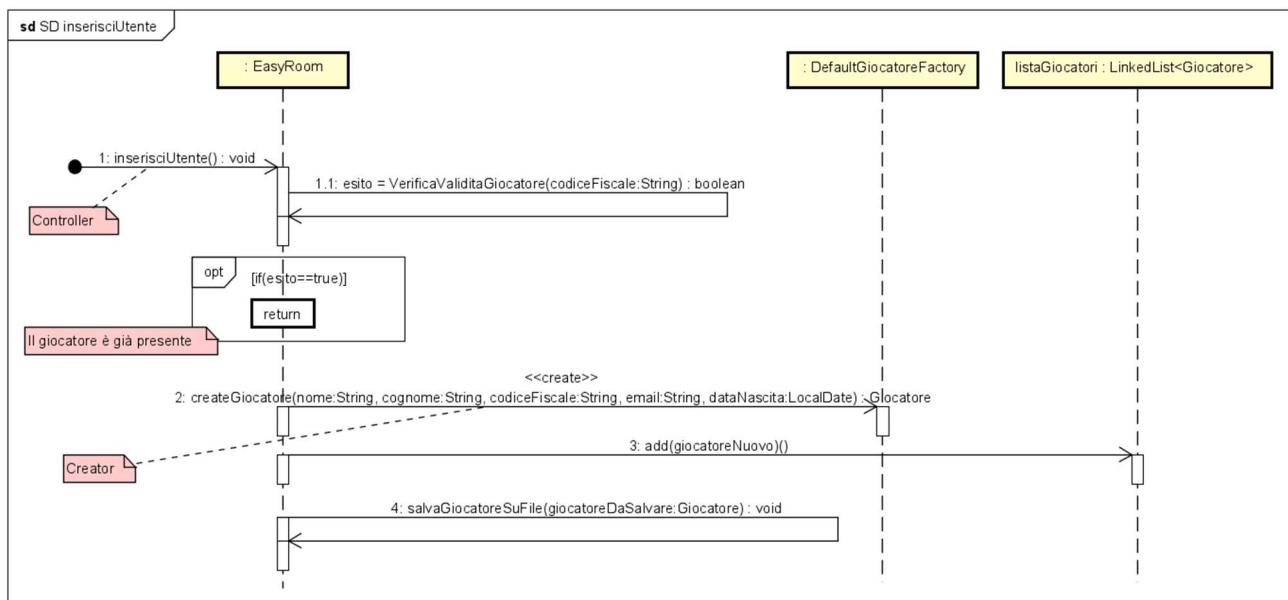
Il pattern Factory Method appartiene ai pattern creazionali, esso si concentra sulla creazione di oggetti, fornendo un’interfaccia per la creazione, ma delegando alle sottoclassi (che implementano materialmente

la creazione) la responsabilità. Tra i motivi per cui viene utilizzato il pattern Factory Method rientrano la flessibilità, in quanto è sufficiente creare una nuova implementazione della factory senza modificare il codice che utilizza il Factory Method; separazione della responsabilità, in quanto l'utente non deve preoccuparsi di conoscere materialmente come devono essere creati gli oggetti, ma delega una sottoclassa utilizzando un metodo conosciuto.



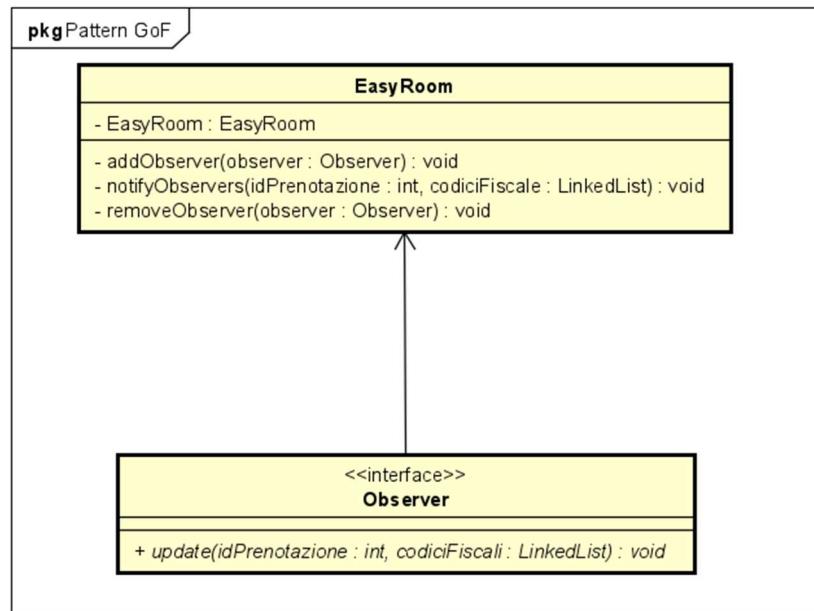
Nel sistema è stata realizzata una interfaccia chiamata “GiocatoreFactory” che contiene la definizione di un metodo “createGiocatore()”, che è materialmente implementato all’interno della classe “DefaultGiocatoreFactory”. Per poter creare un’istanza di Giocatore, è necessario creare una “DefaultGiocatoreFactory” ed invocare il metodo “createGiocatore()”, passandogli i parametri corretti.

L’utilizzo di tale pattern ha portato ad una modifica di “SD inserisciUtente()” in cui non si fa più riferimento alla classe “Giocatore” per la creazione dell’istanza, ma a “DefaultGiocatoreFactory”



Pattern GoF Observer

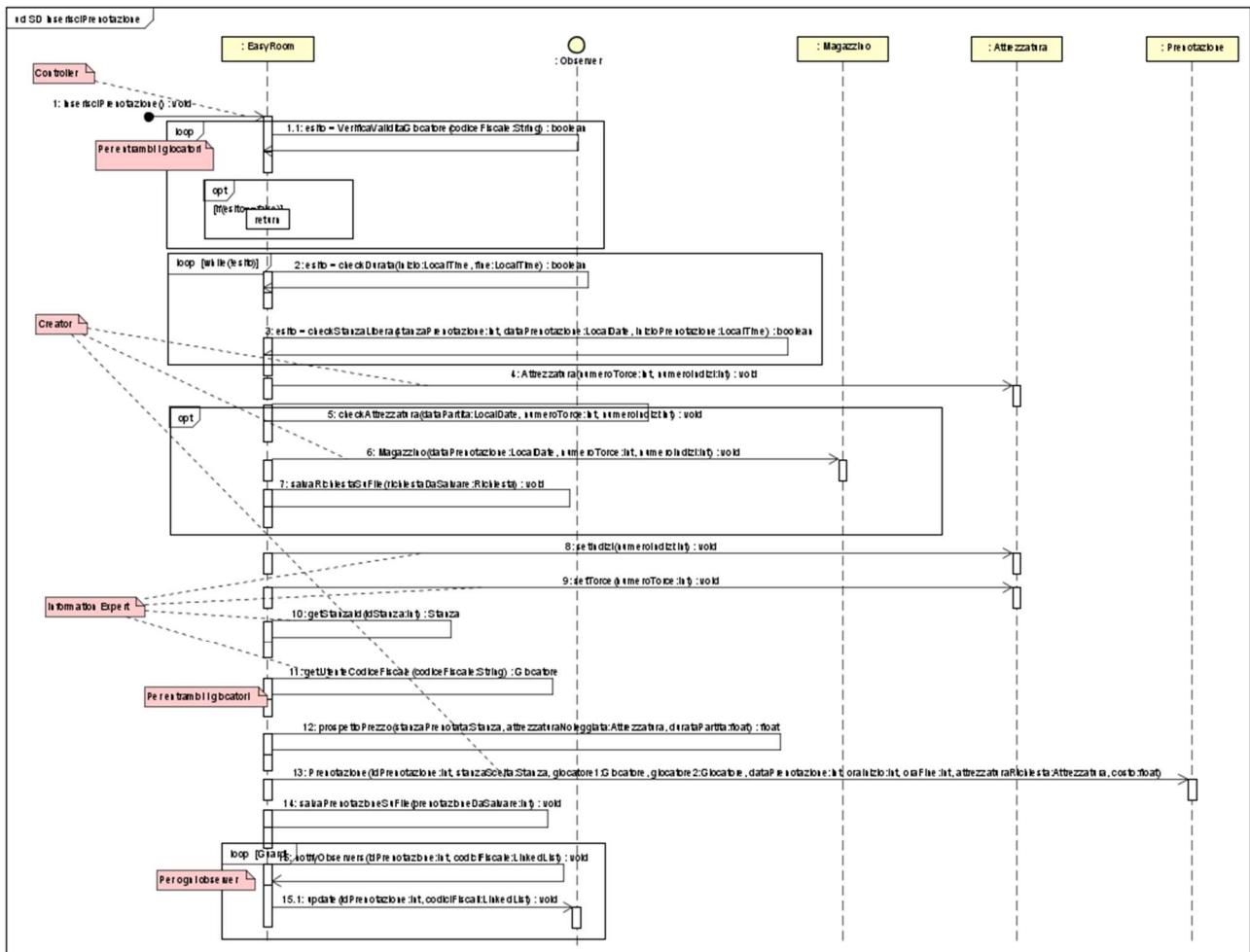
Il pattern Observer appartiene ai pattern comportamentali, esso permette di mantenere una lista di osservatori che saranno notificati quando si verificano dei cambiamenti nello stato del soggetto.



Il pattern Observer è stato implementato direttamente nella classe EasyRoom (che funge da Subject), inoltre sono stati aggiunti i seguenti metodi per gestire gli osservatori:

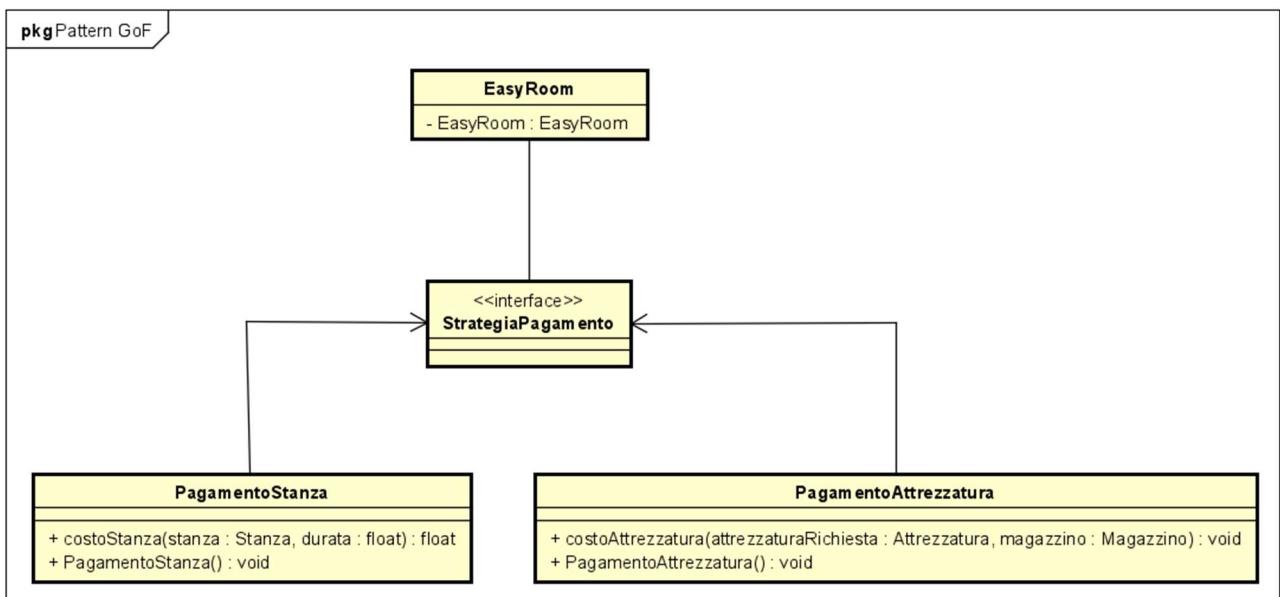
- `addObserver(...)`: Aggiunge un nuovo osservatore alla lista degli osservatori interessati al Subject.
- `removeObserver(...)`: Rimuove un nuovo osservatore alla lista degli osservatori interessati al Subject.
- `notifyObservers(...)`: Notifica tutti gli osservatori nella lista quando viene inserita una prenotazione. Gli osservatori ricevono l'ID della prenotazione e l'elenco dei codici fiscali dei partecipanti.

L'utilizzo di tale pattern ha portato ad una modifica di “SD: inserisciPrenotazione”

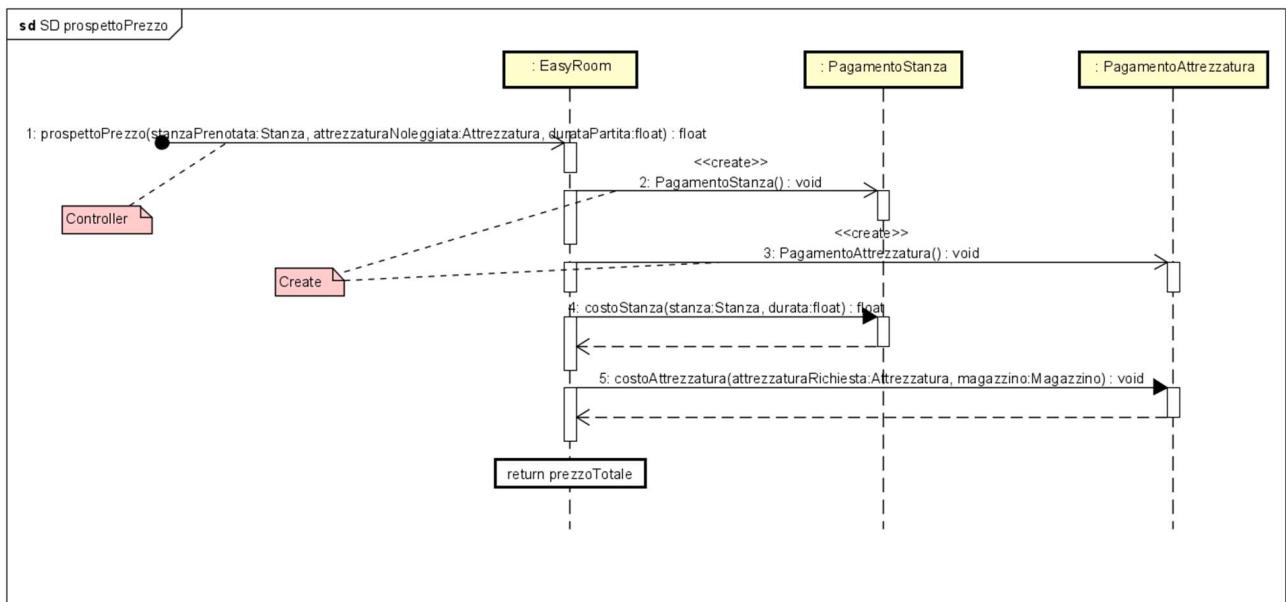


Pattern GoF Strategy

Il pattern Strategy permette di definire una serie di algoritmi , incapsularli e renderli intercambiabili. In questo caso, il pattern è stato utilizzato per ottenere delle strategie per il calcolo del costo della prenotazioni di una stanza.



L'applicazione di questo pattern ha portato ad una modifica di "SD: calcoloPrezzo" che per comodità è stato rinominato in "SD: prospettoPrezzo"



Elaborazione – Iterazione 3

L'obiettivo dell'Iterazione 3 è implementare lo scenario principale di successo e tutte le estensioni individuate nel caso d'uso UC3: "Modifica/annullamento di una prenotazione" nella fase di ideazione. Un altro obiettivo è quello di implementare alcuni pattern GoF per rendere più robusta l'applicazione.

Analisi Orientata agli Oggetti

L'analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti. Vengono utilizzati diversi strumenti per fornire tale descrizione: Modello di Dominio, SSD, e Contratti delle operazioni.

Modello di Dominio

Per il caso d'uso scelto UC3 sono stati identificati le seguenti classi concettuali:

- EasyRoom: rappresenta l'applicazione;
- Giocatore: utente che vuole utilizzare il Sistema per prenotare una stanza, è l'attore primario di questo caso d'uso.
- Prenotazione: entità che rappresenta l'esclusiva sull'utilizzo di una stanza in un intervallo di tempo ben definito.
- Stanza: contiene le informazioni relative ad una escape room.
- Magazzino: contiene le informazioni e tiene traccia delle richieste effettuate.
- Attrezzatura: contiene la quantità di utilities richieste.

È stato ricavato il seguente Modello di Dominio:

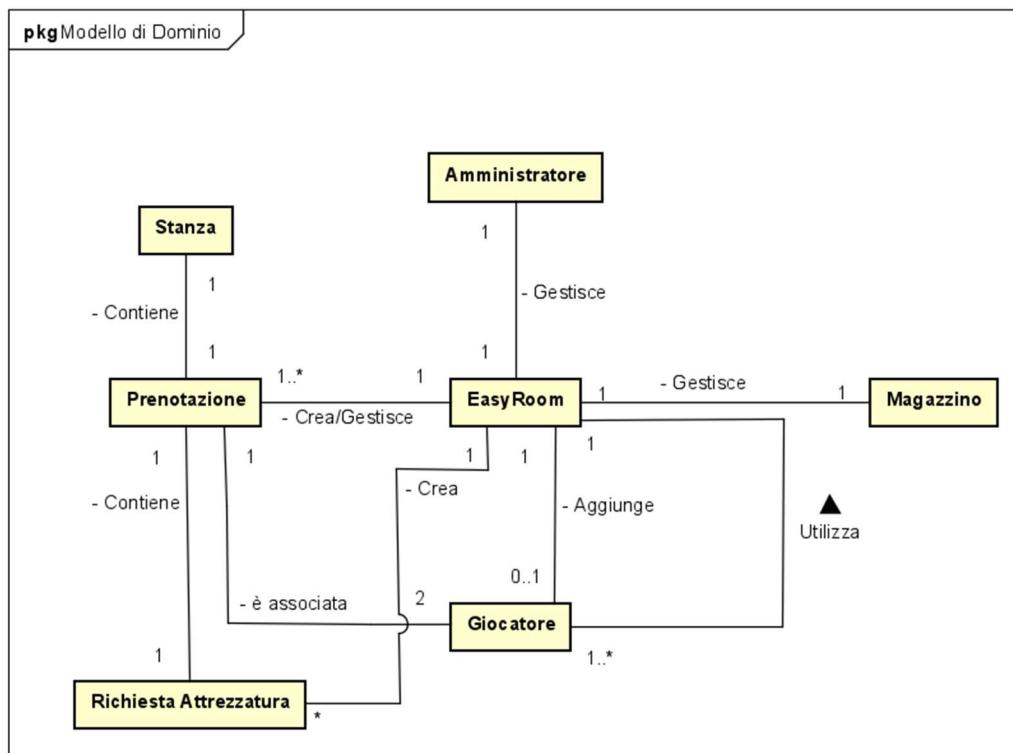
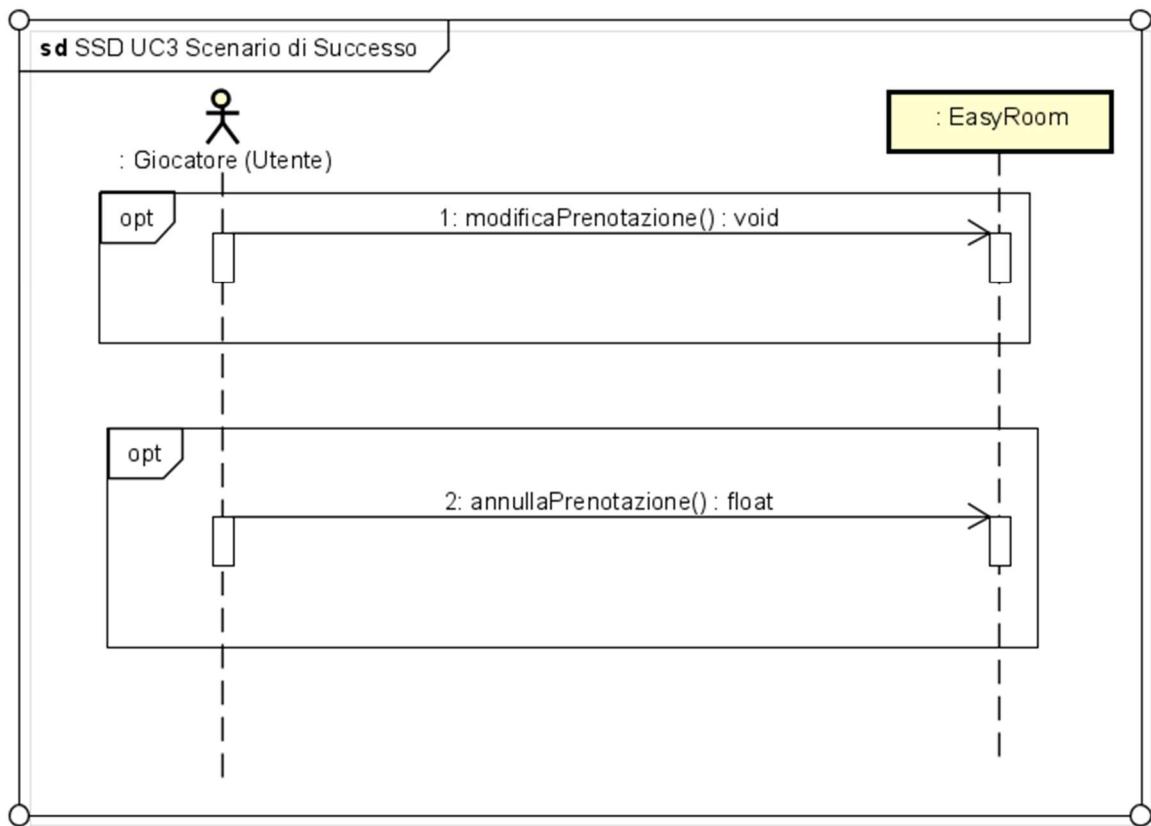


Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC3:



In questo diagramma, il giocatore(Utente) chiede al sistema di modificare o annullare una prenotazione. Il sistema gestisce queste due operazioni in modo separato tramite i metodi `modificaPrenotazione()` e `annullaPrenotazione()`, che vengono invocati in base alla scelta dell'utente.

Il metodo `modificaPrenotazione()` si occupa di chiedere all'utente quale prenotazione debba essere modificata, il sistema verifica la presenza della prenotazione e, in caso di successo, chiede all'utente di reinserire i valori tipici di una prenotazione modificando la prenotazione corrente senza creare una nuova. Sulla prenotazione modificata verranno eseguiti i controlli tipici della prenotazione, in caso di inserimento di una prenotazione non valida, il sistema ritornerà al menù e non modificherà la prenotazione scelta.

Il metodo `annullaPrenotazione()` si occupa di chiedere all'utente quale prenotazione debba essere annullata, il sistema verifica la presenza della prenotazione e, in caso di successo, elimina la prenotazione dal sistema. Il sistema, in base al periodo temporale in cui viene eliminata la prenotazione, calcola un valore di rimborso che può essere pari al valore totale o ad una certa percentuale a causa di una penale dovuta all'eliminazione con poco preavviso.

Contratto delle operazioni

Di seguito viene indicato il Contratto delle operazioni per l'UC2:

Operazione:	<code>modificaPrenotazione()</code>
Riferimenti:	Caso d'uso UC3: Modifica/annullamento di una prenotazione.
Pre-condizioni:	-
Post-condizioni:	<ul style="list-style-type: none"> 6. È stata modificata l'istanza <code>prenotazioneDaModificare</code> di <code>Prenotazione</code>. 7. Gli attributi <code>idPrenotazione</code>, <code>stanzaPrenotata</code>, <code>primoGiocatore</code>, <code>secondoGiocatore</code>, <code>dataPartita</code>, <code>tempoInizio</code>, <code>tempoFine</code>,

	attrezzaturaRichiesta, costo di nuovaPrenotazione sono stati aggiornati correttamente con valori validi che rispettano le regole di dominio. prenotazioneDaModificare è stato associato a EasyRoom. Viene inserito in una lista mantenuta in memoria in maniera persistente (file di testo).
--	--

Operazione:	annullaPrenotazione()
Riferimenti:	Caso d'uso UC3: Modifica/annullamento di una prenotazione.
Pre-condizioni:	-
Post-condizioni:	È stata eliminata dalla memoria e dai file l'istanza prenotazioneDaModificare di Prenotazione.

Operazione:	checkDurata(inizioPartita, finePartita)
Riferimenti:	Caso d'uso UC3: Modifica/annullamento di una prenotazione.
Pre-condizioni:	È in corso la modifica di una nuova prenotazione. L'utente ha fornito degli orari di inizio e fine della partita.
Post-condizioni:	È stato restituito un Messaggio di Verifica dal Sistema. Viene restituito un booleano che da informazioni sull'esito.

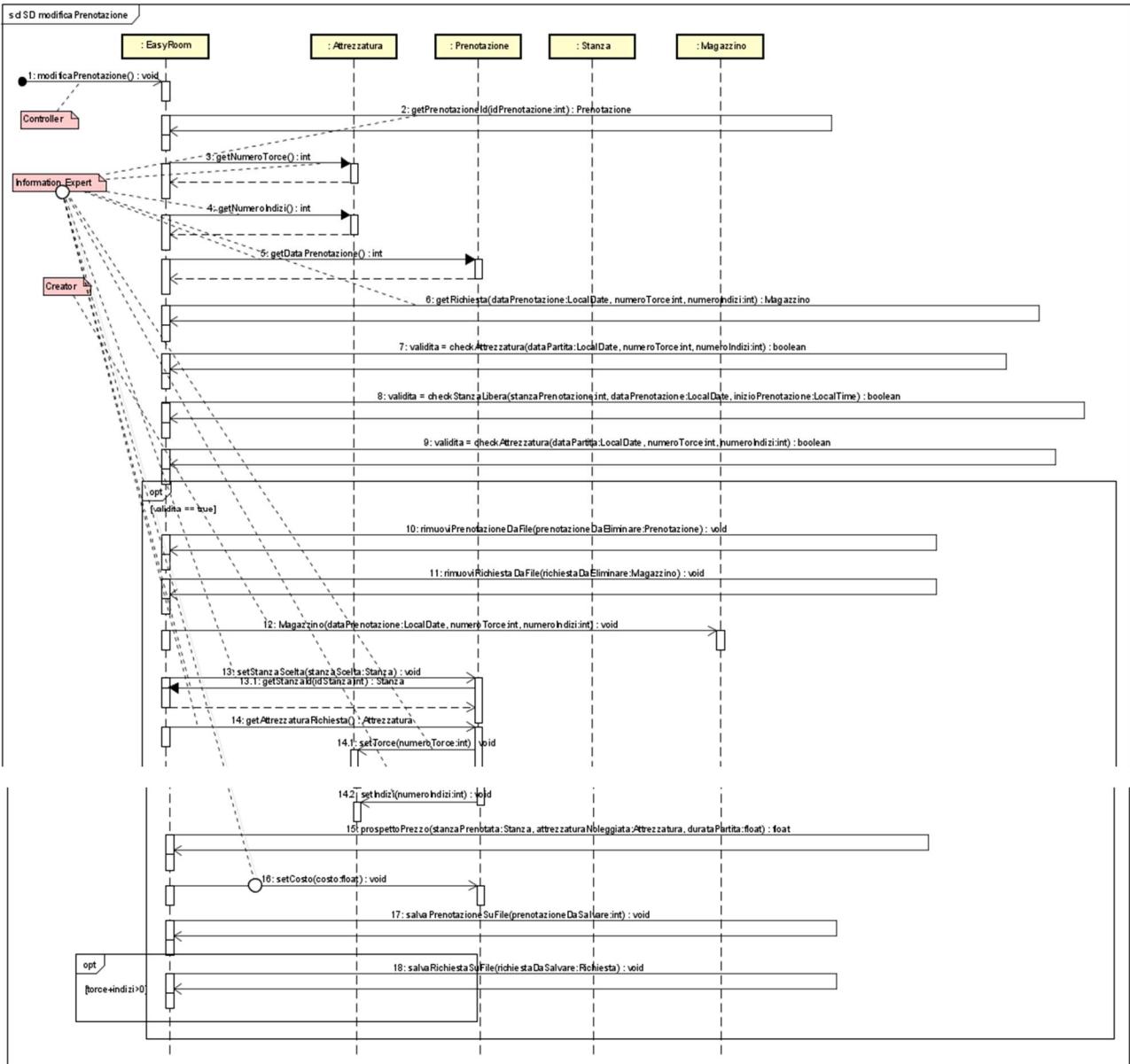
Operazione:	checkStanzaLibera(stanzaPrenotazione, dataPrenotazione, inizioPrenotazione)
Riferimenti:	Caso d'uso UC3: Modifica/annullamento di una prenotazione.
Pre-condizioni:	È in corso l'inserimento di una nuova prenotazione. L'utente ha immesso correttamente i dati.
Post-condizioni:	È stato restituito un Messaggio di Verifica dal Sistema. Viene restituito un booleano che da informazioni sull'esito.

Operazione:	prospettoPrezzo(stanzaPrenotazione, attrezzaturaRichiesta, durata)
Riferimenti:	Caso d'uso UC3: Modifica/annullamento di una prenotazione.
Pre-condizioni:	È in corso la modifica di una nuova prenotazione. I Messaggi di Verifica hanno dato esito positivo, cioè la durata della partita è nei limiti, la stanza è disponibile e si è scelta l'attrezzatura.
Post-condizioni:	È stato calcolato un prezzo dal Sistema. Viene restituito un float che tiene conto del prezzo della stanza e dell'attrezzatura scelta.

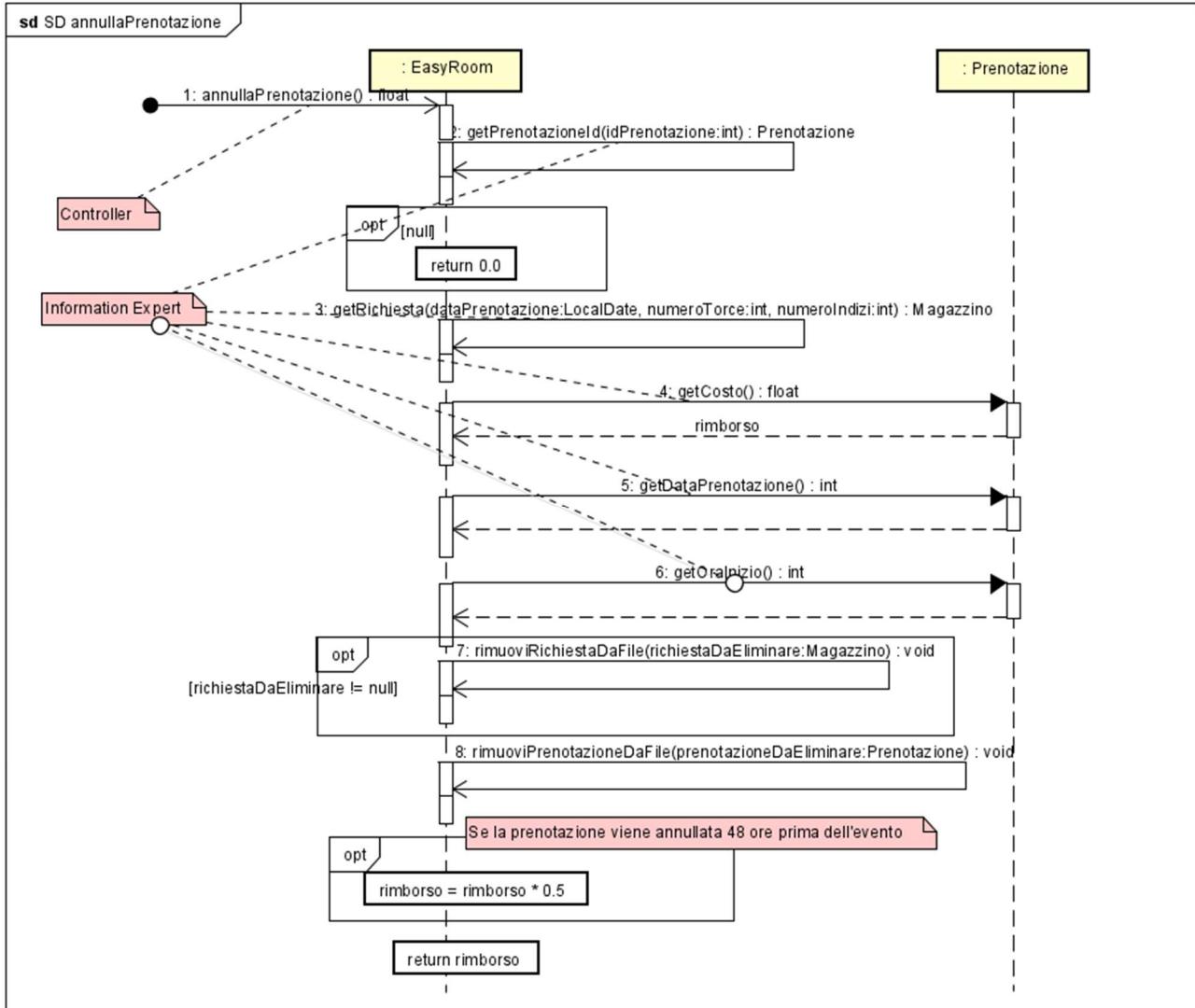
Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

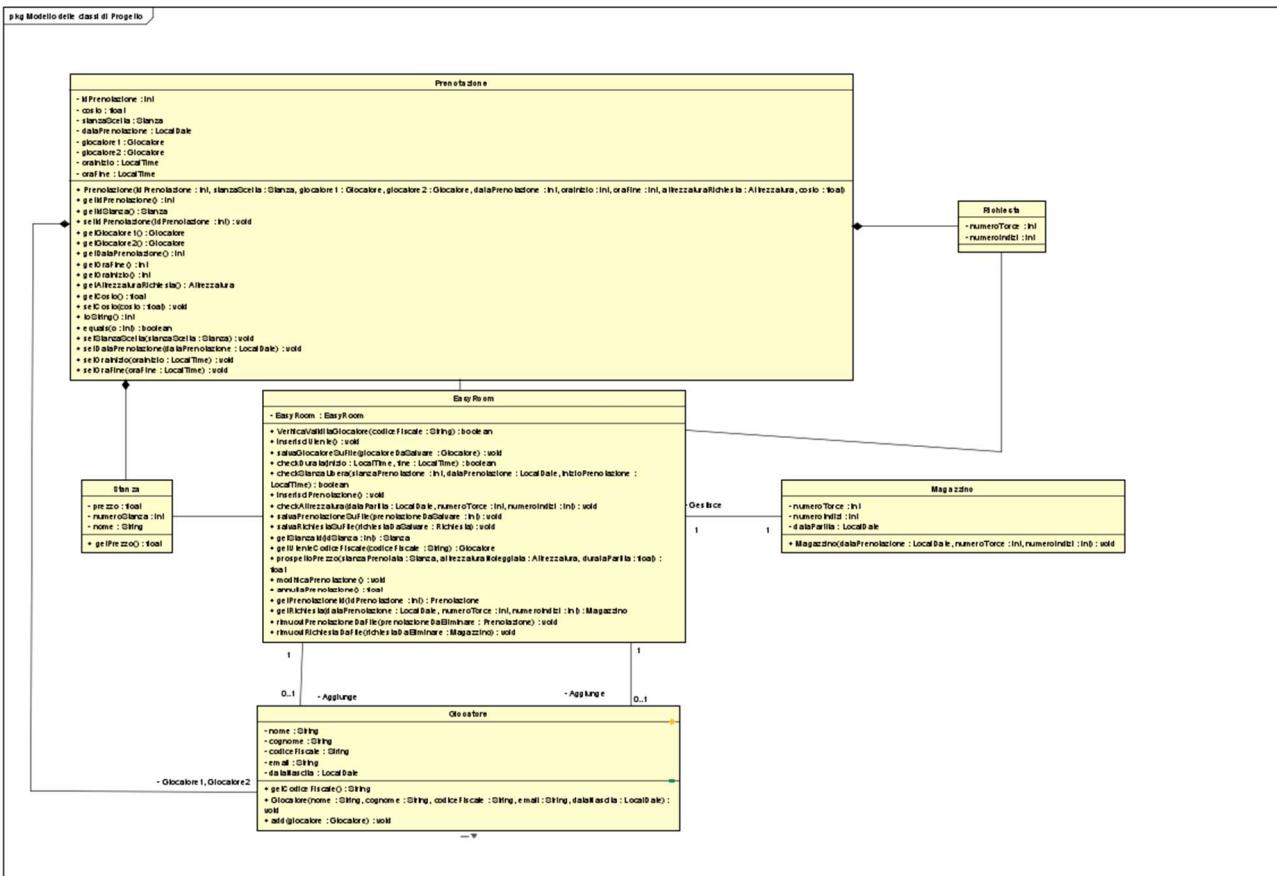
`modificaPrenotazione() : void`



`annullaPrenotazione() : float`



Modello delle classi di Progetto



Elaborazione – Iterazione 4

L'obiettivo dell'Iterazione 4 è implementare gli scenari di successo e tutte le estensioni individuate nei casi d'uso UC4: "Inserimento/modifica di una stanza", UC5: "Visualizzazione prenotazioni", UC6: "Check-in stanza", UC7: "Check-out stanza", UC8: "Conteggio partite in una determinata stanza".

Analisi Orientata agli Oggetti

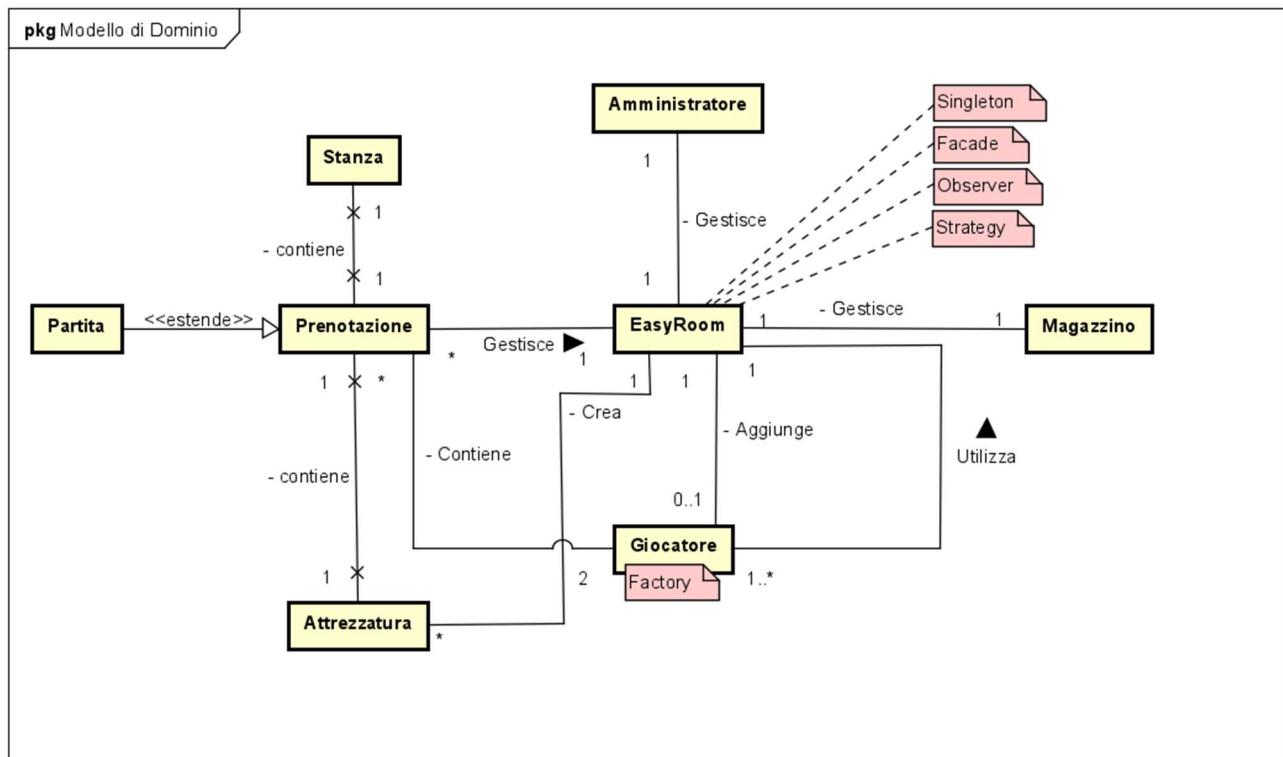
L'analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti. Vengono utilizzati diversi strumenti per fornire tale descrizione: Modello di Dominio, SSD, e Contratti delle operazioni.

Modello di Dominio

Per i casi d'uso scelti sono state identificate le seguenti classi concettuali:

- EasyRoom: rappresenta l'applicazione;
- Giocatore: utente che vuole utilizzare il Sistema per prenotare una stanza, è l'attore primario di questo caso d'uso.
- Prenotazione: entità che rappresenta l'esclusiva sull'utilizzo di una stanza in un intervallo di tempo ben definito.
- Stanza: contiene le informazioni relative ad una escape room.
- Magazzino: contiene le informazioni e tiene traccia delle richieste effettuate.
- Attrezzatura: contiene la quantità di utilities richieste.
- Partita: una estensione della prenotazione che rappresenta il passaggio dal riservare la stanza all'utilizzo vero e proprio della stanza stessa.

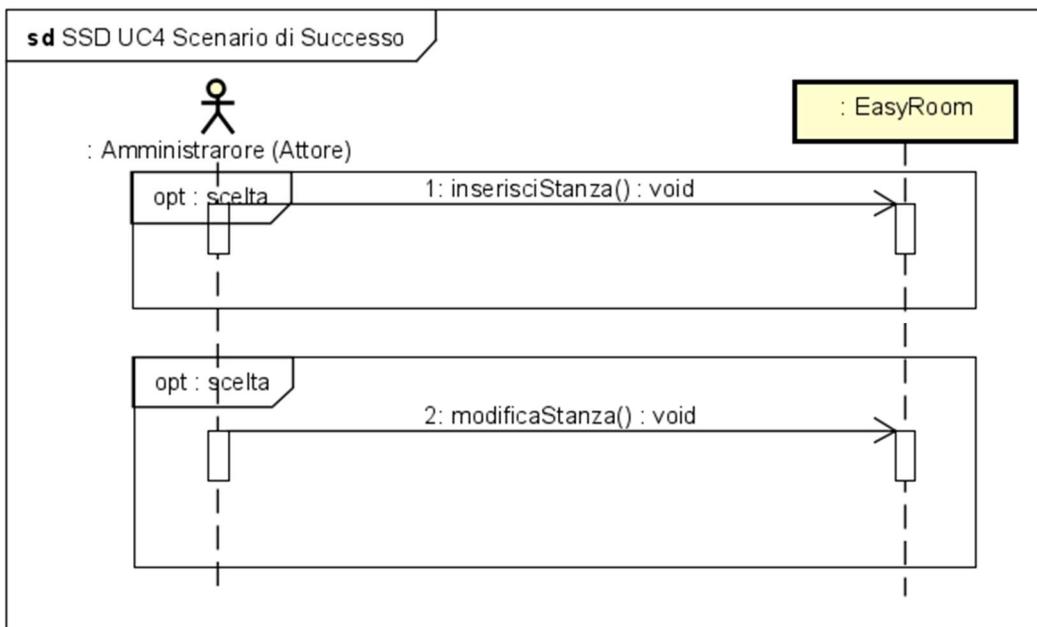
È stato ricavato il seguente Modello di Dominio:



UC4

Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC4:



In questo diagramma, l'Amministratore chiede al sistema di inserire o modificare una prenotazione. Il sistema gestisce queste due operazioni in modo separato tramite i metodi inserisciStanza() e modificaStanza(), che vengono invocati in base alla scelta dell'utente.

Il metodo inserisciStanza() si occupa di aggiungere nel sistema una stanza. Il sistema chiede all'amministratore di inserire il nome della stanza e il costo/h, il numero della stanza (che identifica la stanza stessa in maniera univoca) verrà calcolato in maniera automatica. Una volta creata l'istanza di Stanza, il sistema la aggiungerà in un file di testo (persistenza dei dati) e aggiornerà la lista in memoria.

Il metodo modificaStanza() si occupa di modificare una stanza che è presente nel sistema, dove per modificare si intende avere la possibilità di cambiare il nome e il prezzo/h della stanza. Il sistema chiederà all'Amministratore di inserire il numero della stanza da modificare, se la stanza non dovesse esistere verrà notificato e il sistema tornerà al menù. Una volta ottenuta la stanza verranno chiesti i nuovi valori di nome e prezzo/h, verrà modificata l'istanza senza creare un nuovo oggetto Stanza, successivamente verrà aggiornato il file di testo contenente le stanze (persistenza dei dati).

Contratto delle operazioni

Di seguito viene indicato il Contratto delle operazioni per l'UC4:

Operazione:	inserisciStanza()
Riferimenti:	Caso d'uso UC4: Inserimento/Modifica di una stanza.
Pre-condizioni:	-
Post-condizioni:	<ol style="list-style-type: none">8. È stata creata l'istanza nuovaStanza di Stanza.9. Gli attributi numeroStanza, nome e prezzo sono stati inseriti correttamente con valori validi che rispettano le regole di dominio. nuovaStanza è stato associato a EasyRoom. Viene

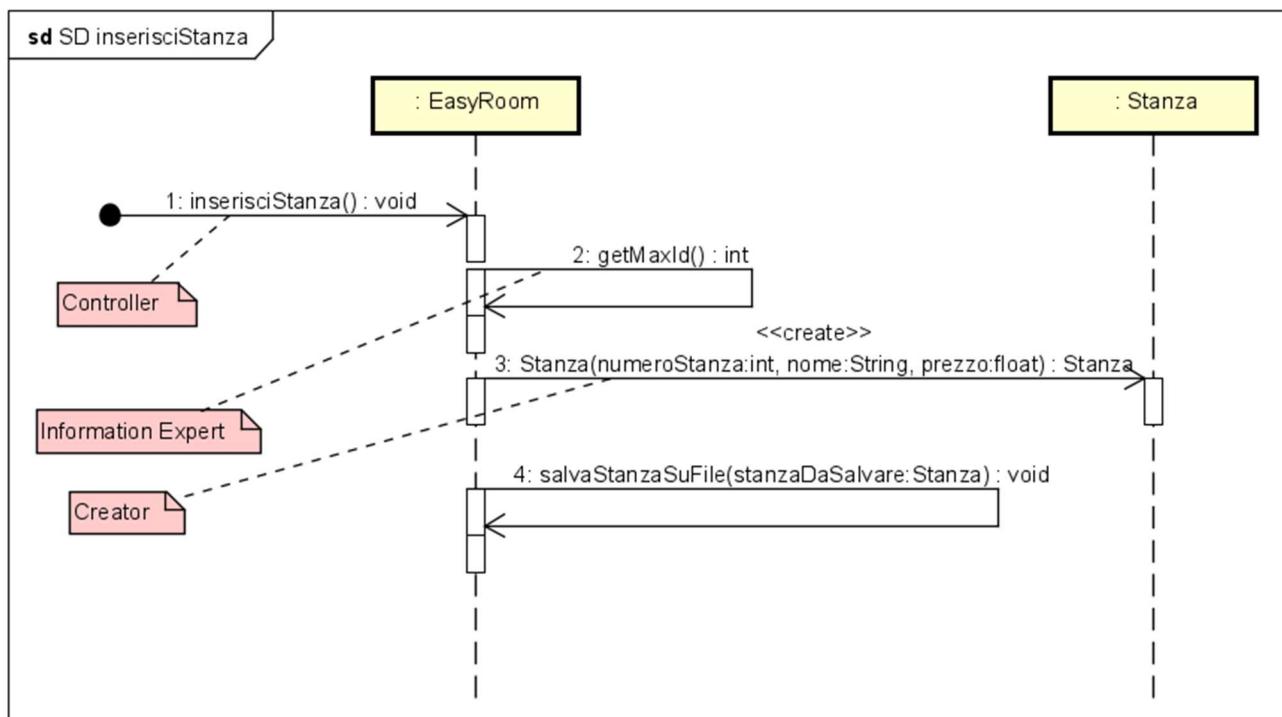
	inserito in una lista mantenuta in memoria in maniera persistente (file di testo).
--	--

Operazione:	modificaStanza()
Riferimenti:	Caso d'uso UC4: Inserimento/Modifica di una stanza.
Pre-condizioni:	-
Post-condizioni:	<p>1. È stata modificata l'istanza stanzaDaModificare di Stanza.</p> <p>2. Gli attributi nome e prezzo sono stati aggiornati correttamente con valori validi che rispettano le regole di dominio.</p> <p>stanzaDaModificare è stato associato a EasyRoom. Vengono aggiornati la lista mantenuta in memoria e i dati persistenti (file di testo).</p>

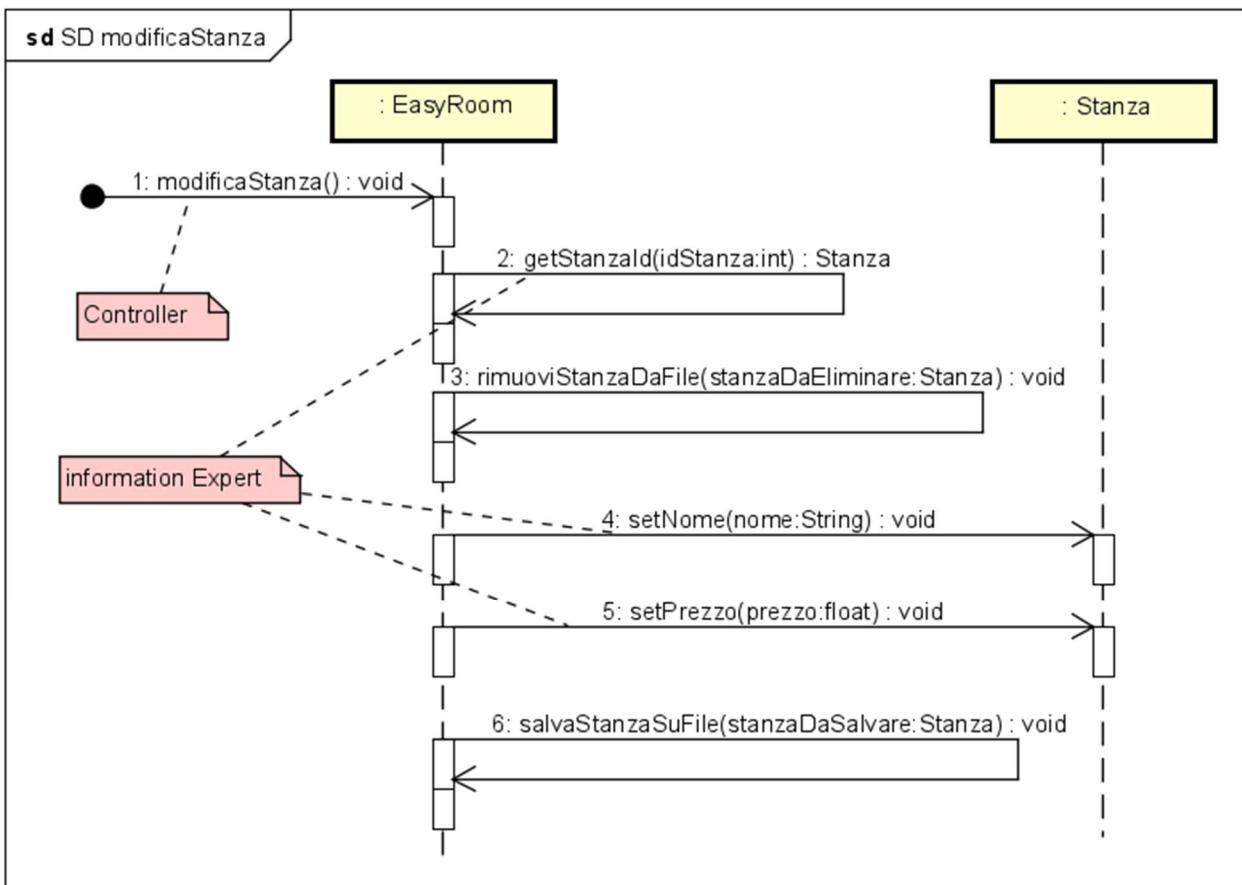
Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

inserisciStanza() : void



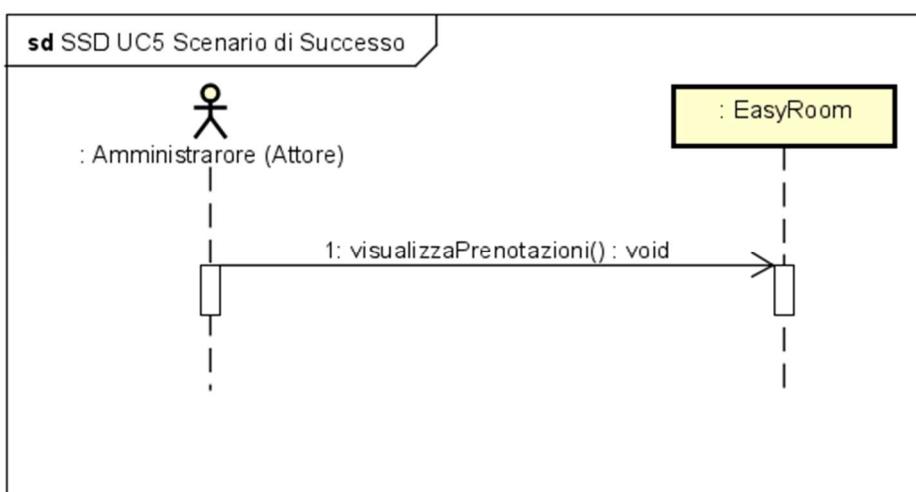
modificaStanza() : void



UC5

Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC5:



Il metodo `visualizzaPrenotazioni()` chiede all'amministratore se vuole visualizzare le prenotazioni di tutte le stanze o di una singola stanza, nell'ultimo caso chiede l'inserimento del numero della stanza. A schermo verranno stampare le prenotazioni scelte.

Contratto delle operazioni

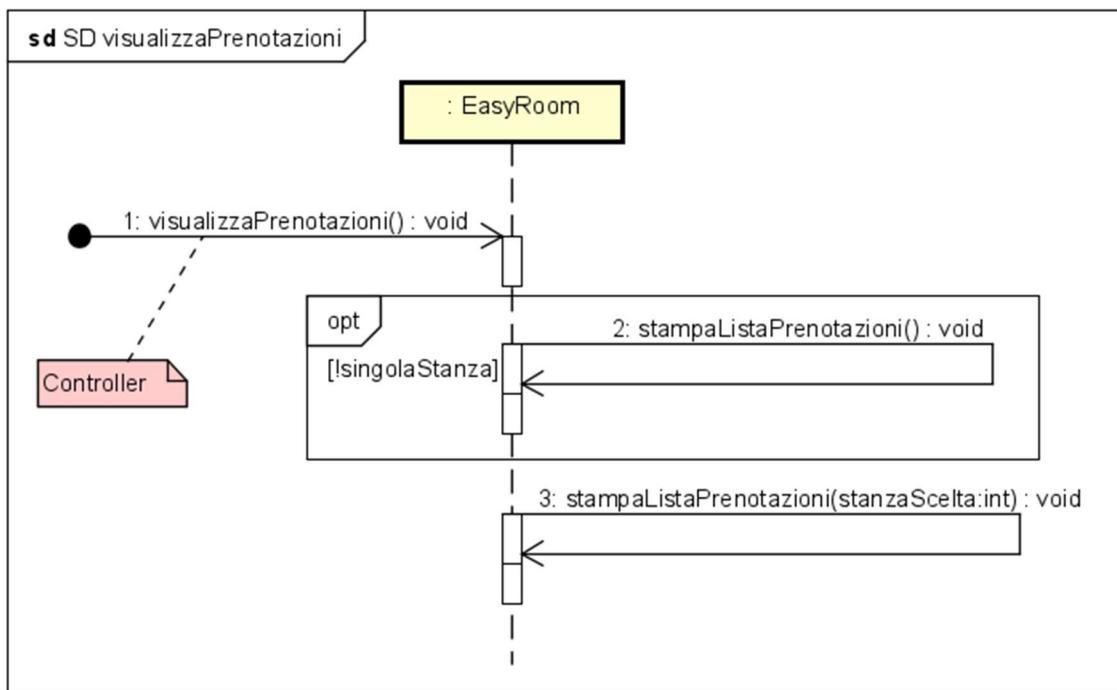
Di seguito viene indicato il Contratto delle operazioni per l'UC5:

Operazione:	visualizzaPrenotazioni()
Riferimenti:	Caso d'uso UC5: Visualizzazione prenotazioni.
Pre-condizioni:	Le prenotazioni devono essere caricate nel sistema.
Post-condizioni:	Vengono stampate a schermo le prenotazioni della stanza scelta o di tutte le stanze.

Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

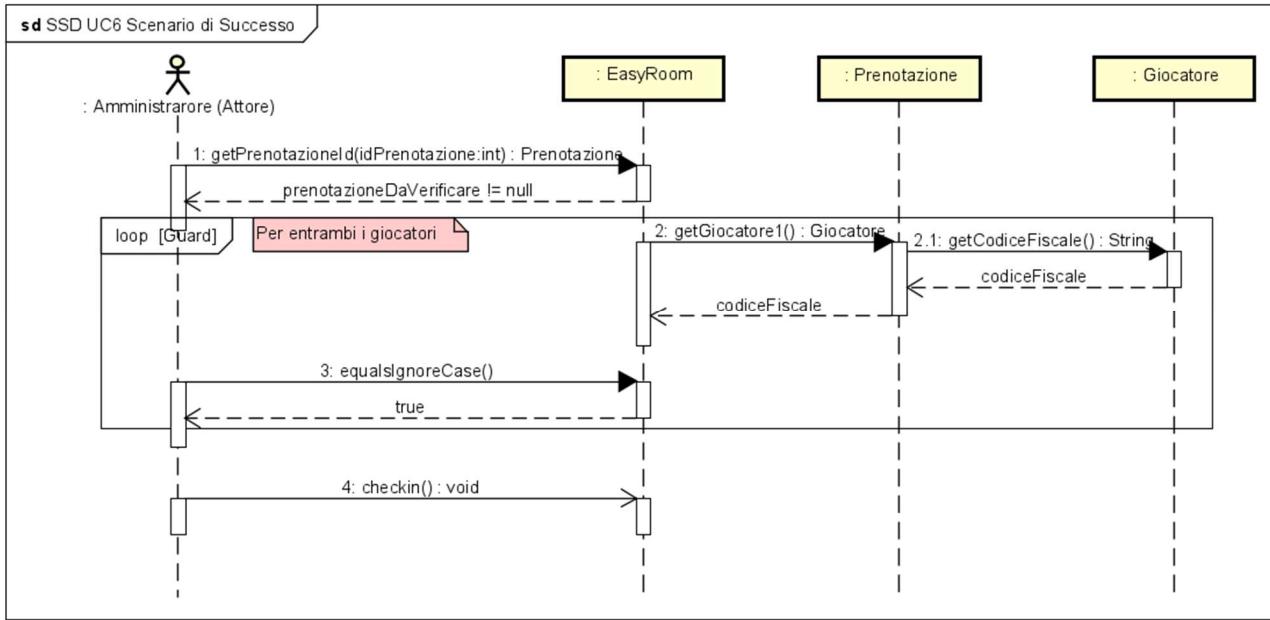
visualizzaPrenotazioni() : void



UC6

Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC6:



Il metodo `checkin()` permette all'amministratore di eseguire il check-in di una prenotazione. Il sistema chiede all'Amministratore di inserire l'ID della prenotazione che si vuole confermare, verificando che quest'ultima sia presente in memoria. Una volta recuperata la prenotazione, il sistema chiede all'Amministratore di inserire i codici fiscali dei due giocatori, se coincidono allora creerà una istanza di Partita, settando come orario di check-in (e temporaneamente anche orario di check-out visto che non è stato ancora eseguito) l'orario corrente, e la memorizzerà nel sistema in maniera persistente.

Contratto delle operazioni

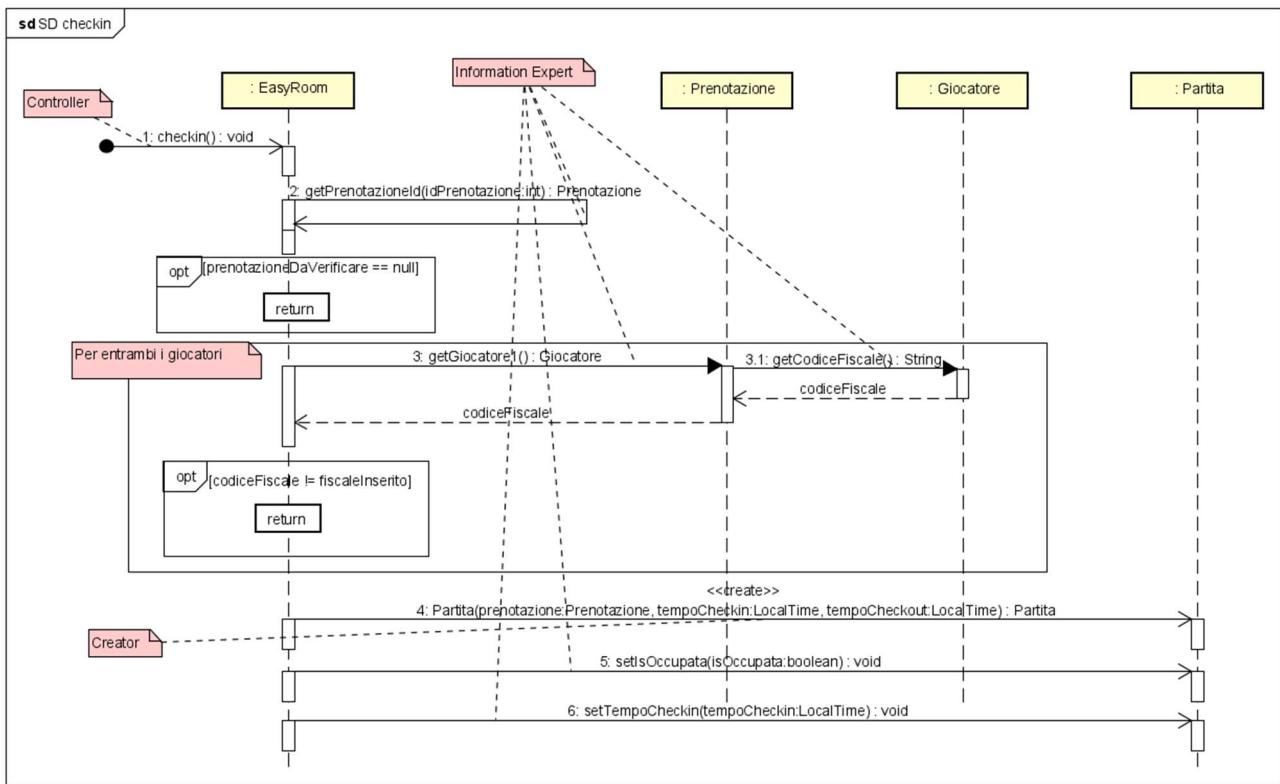
Di seguito viene indicato il Contratto delle operazioni per l'UC6:

Operazione:	<code>checkin()</code>
Riferimenti:	Caso d'uso UC6: Check-in stanza.
Pre-condizioni:	-
Post-condizioni:	Verrà creata una istanza di Partita caratterizzata dai dati della prenotazione associata con informazioni aggiuntive sull'orario di check-in e un valore booleano che indica che la stanza è attualmente occupata.

Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

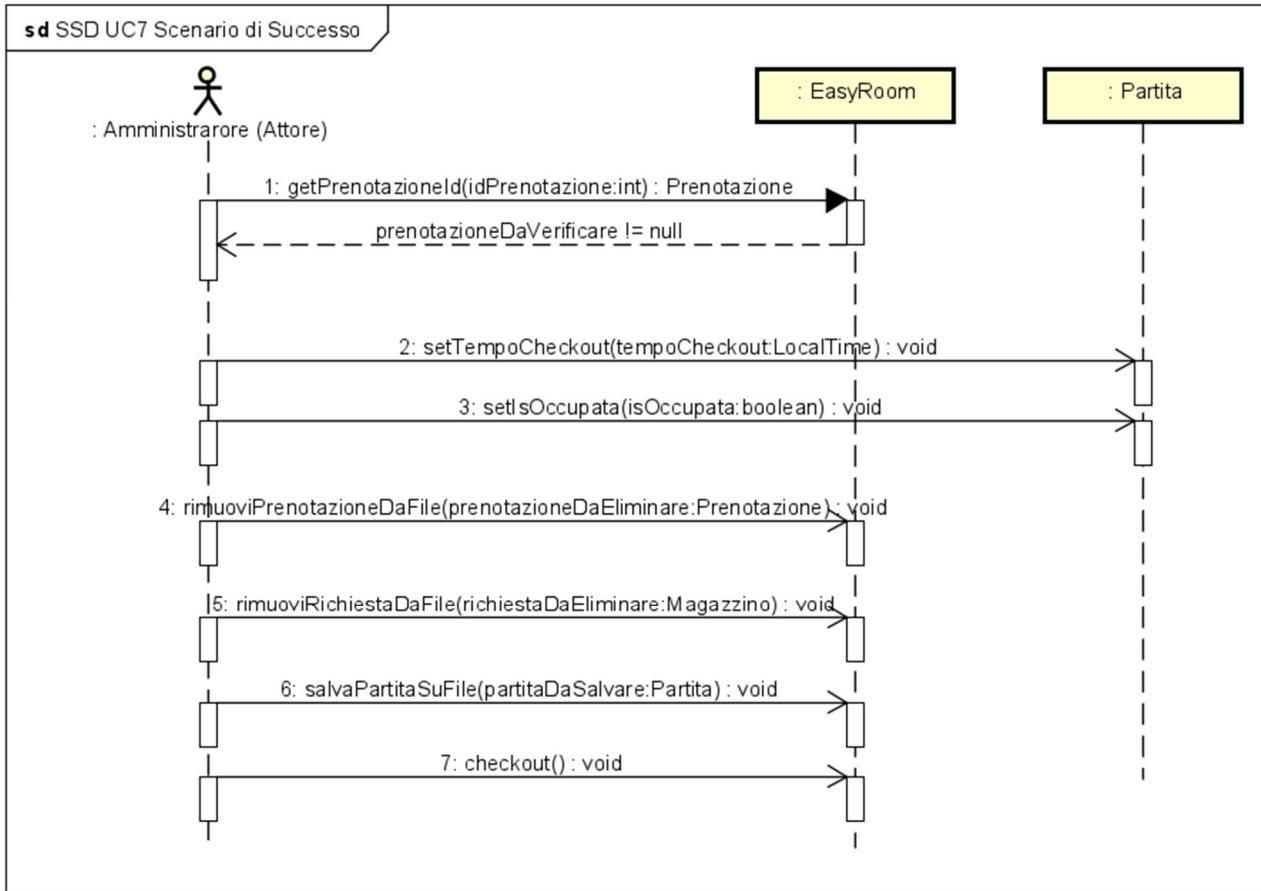
`checkin() : void`



UC7

Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC7:



Il metodo `checkout()` permette all'amministratore di eseguire il check-out di una prenotazione. Il sistema chiede all'Amministratore di inserire l'ID della prenotazione che si vuole confermare, verificando che quest'ultima sia presente in memoria. Una volta recuperata la prenotazione, il sistema setterà l'orario di check-out con l'orario corrente, setterà il valore `isOccupata` a false per indicare che la stanza è stata liberata. Il sistema si occupa di eliminare dalla memoria in maniera persistente la prenotazione ormai obsoleta e aggiunge in memoria in maniera persistente l'istanza di `Partita`, creata in fase di check-in, che fa riferimento alla partita terminata.

Contratto delle operazioni

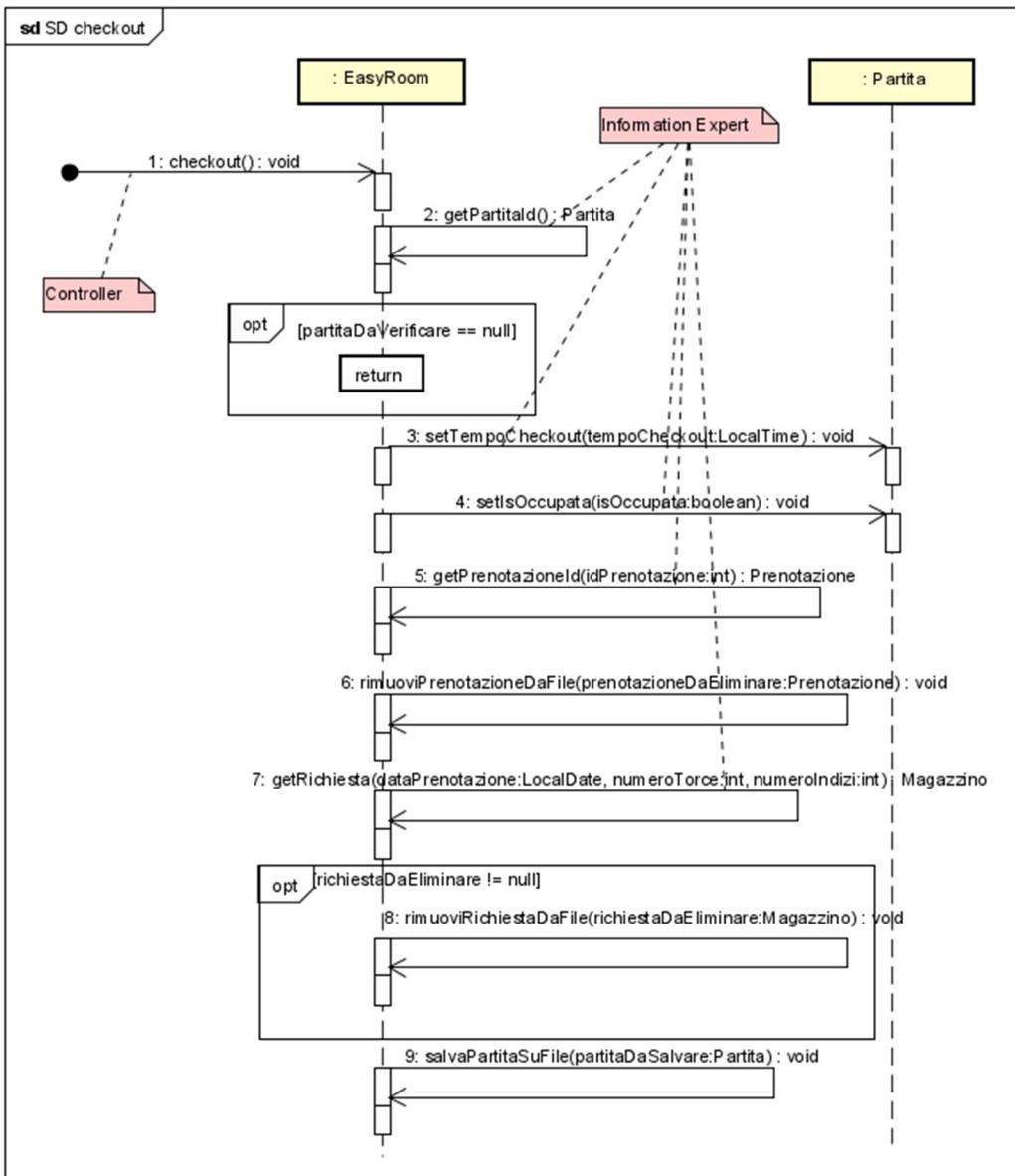
Di seguito viene indicato il Contratto delle operazioni per l'UC7:

Operazione:	<code>checkout()</code>
Riferimenti:	Caso d'uso UC7: Check-out stanza.
Pre-condizioni:	-
Post-condizioni:	Verrà memorizzata in maniera persistente l'istanza di <code>Partita</code> caratterizzata dai dati della prenotazione associata con informazioni aggiuntive sull'orario di check-out e un valore booleano che indica che la stanza è attualmente libera.

Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

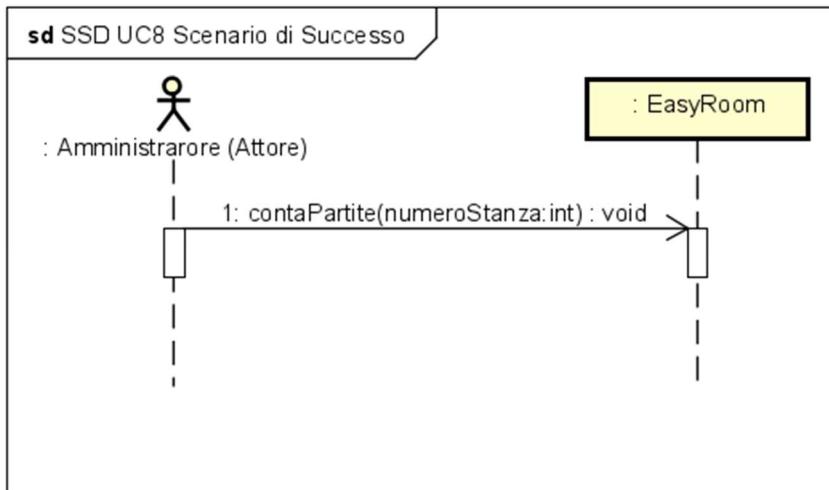
`checkout() : void`



UC8

Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC8:



Il metodo `contaPartite(numeroStanza : int)` permette all'amministratore di contare il numero di partite che sono state giocate in una determinata stanza. Il sistema chiede all'Amministratore di inserire il numero della stanza di cui vuole conoscere le informazioni. Il sistema in base ai dati presenti in memoria aggiornerà un contatore che verrà successivamente stampato a schermo.

Contratto delle operazioni

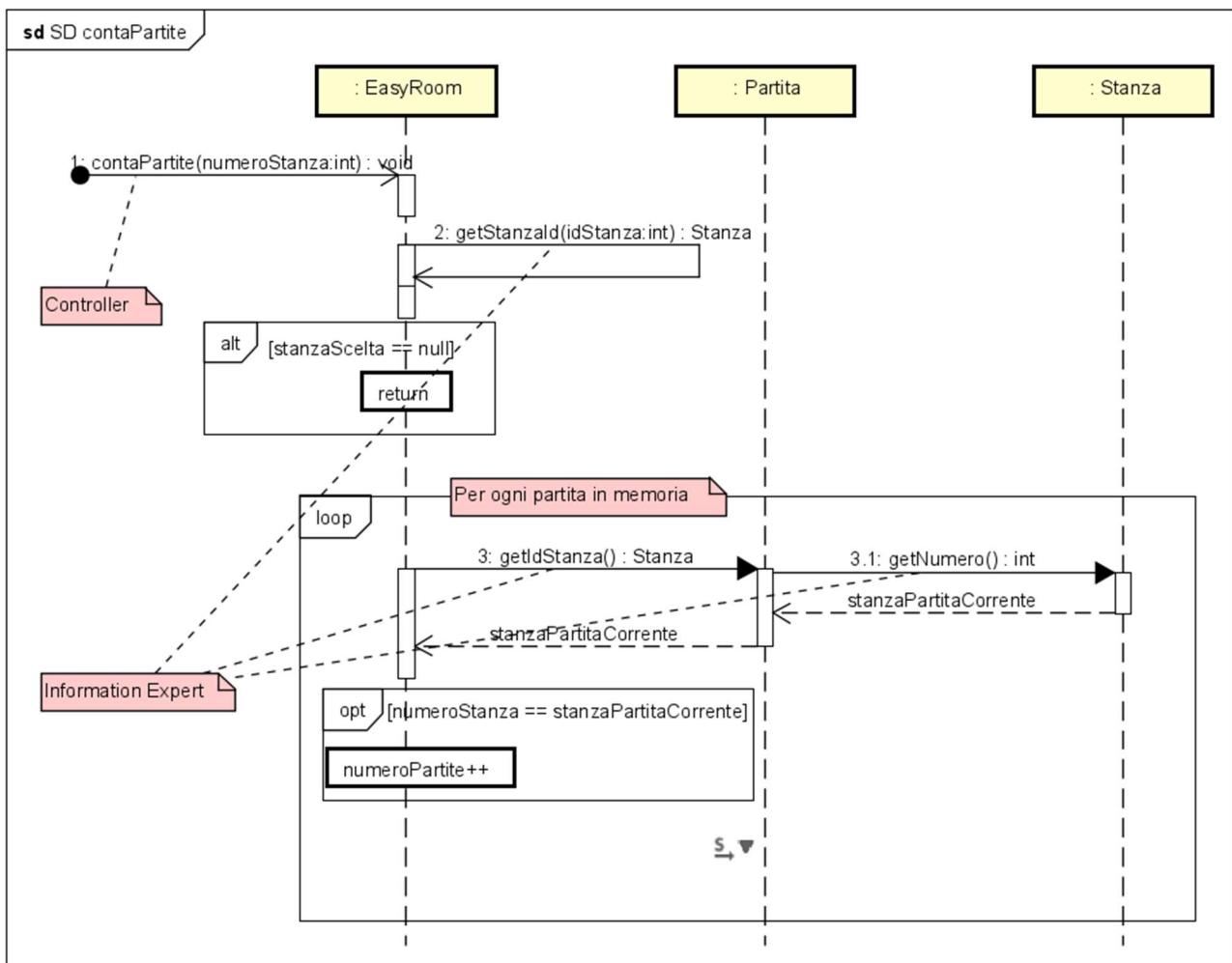
Di seguito viene indicato il Contratto delle operazioni per l'UC7:

Operazione:	<code>contaPartite(numeroStanza : int)</code>
Riferimenti:	Caso d'uso UC8: Conteggio partite in una determinata stanza.
Pre-condizioni:	-
Post-condizioni:	Verrà stampato a schermo il numero di partite giocate in una determinata stanza.

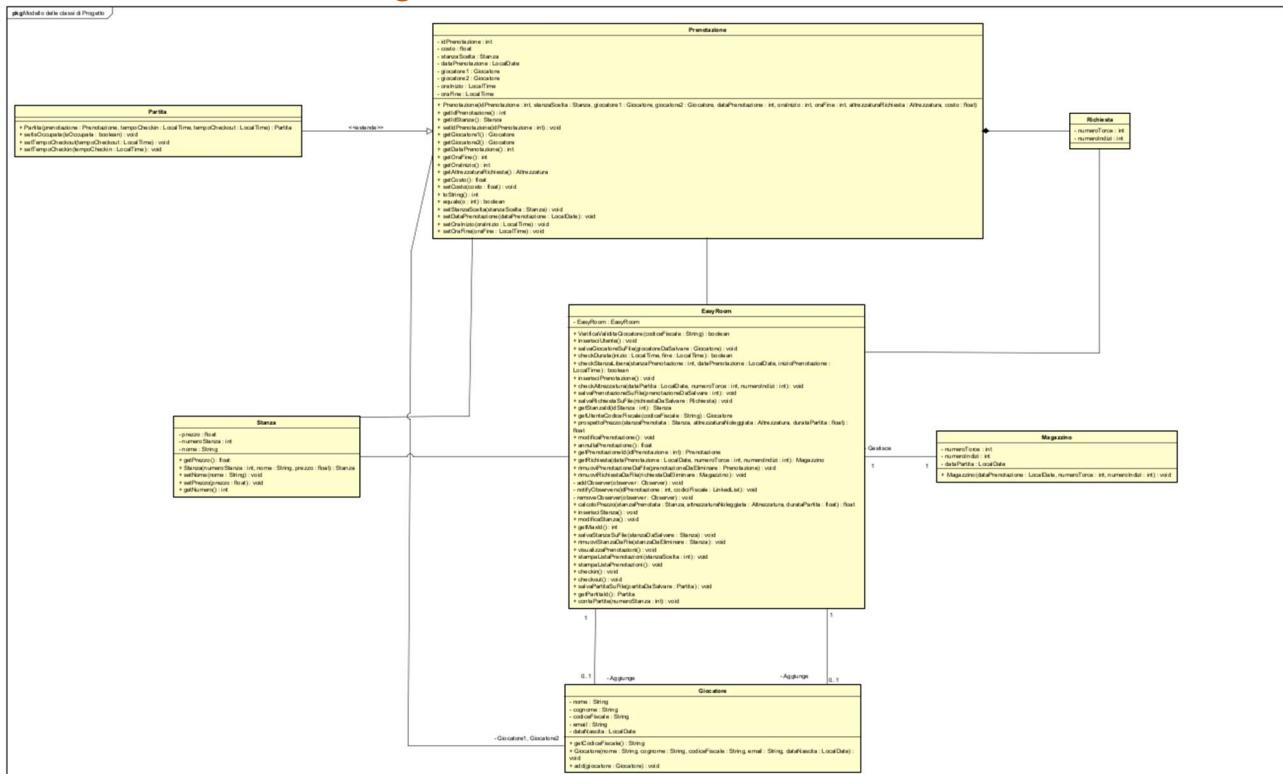
Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

`contaPartite(numeroStanza : int) : void`



Modello delle classi di Progetto



Elaborazione – Iterazione 5

L'obiettivo dell'Iterazione 5 è implementare gli scenari di successo e tutte le estensioni individuate nel caso d'uso UC9: "Gestione del magazzino".

Analisi Orientata agli Oggetti

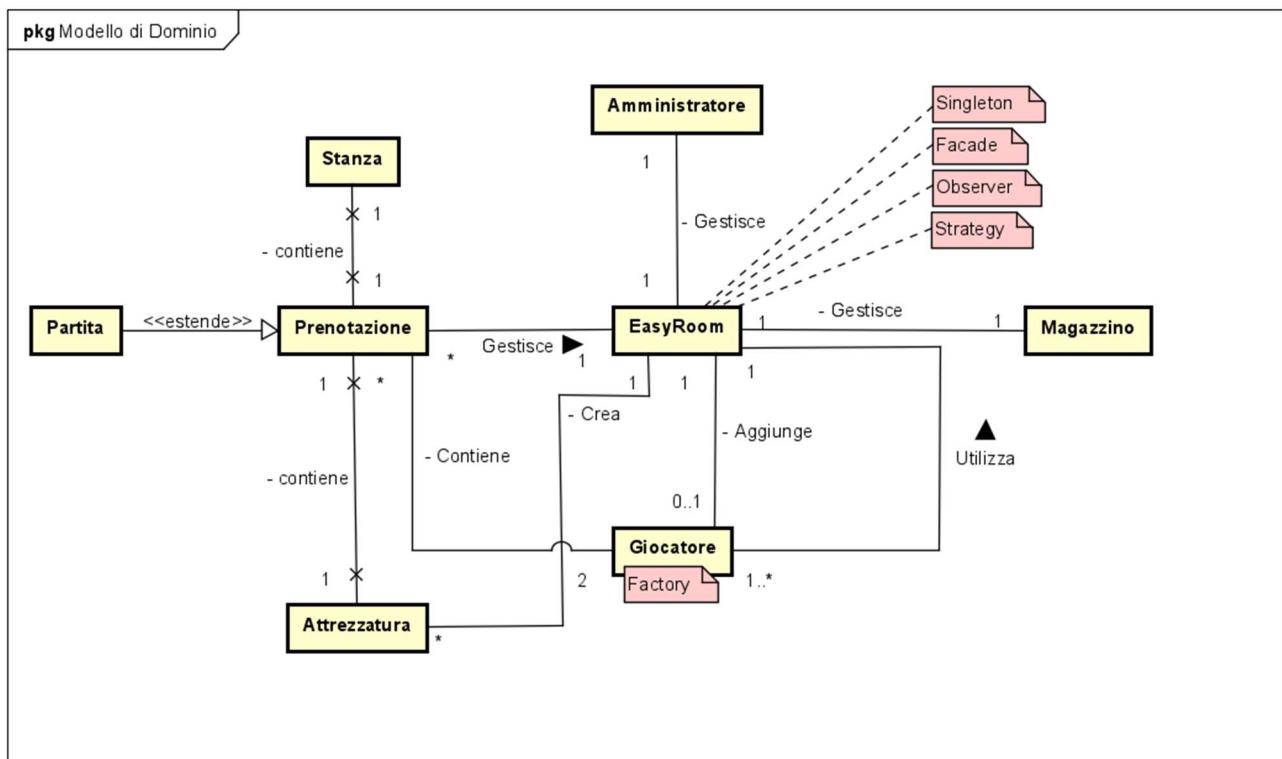
L'analisi orientata agli oggetti si basa sulla creazione di una descrizione del dominio da un punto di vista ad oggetti. Vengono utilizzati diversi strumenti per fornire tale descrizione: Modello di Dominio, SSD, e Contratti delle operazioni.

Modello di Dominio

Per i casi d'uso scelti sono state identificate le seguenti classi concettuali:

- EasyRoom: rappresenta l'applicazione;
- Giocatore: utente che vuole utilizzare il Sistema per prenotare una stanza, è l'attore primario di questo caso d'uso.
- Prenotazione: entità che rappresenta l'esclusiva sull'utilizzo di una stanza in un intervallo di tempo ben definito.
- Stanza: contiene le informazioni relative ad una escape room.
- Magazzino: contiene le informazioni e tiene traccia delle richieste effettuate.
- Attrezzatura: contiene la quantità di utilities richieste.
- Partita: una estensione della prenotazione che rappresenta il passaggio dal riservare la stanza all'utilizzo vero e proprio della stanza stessa.

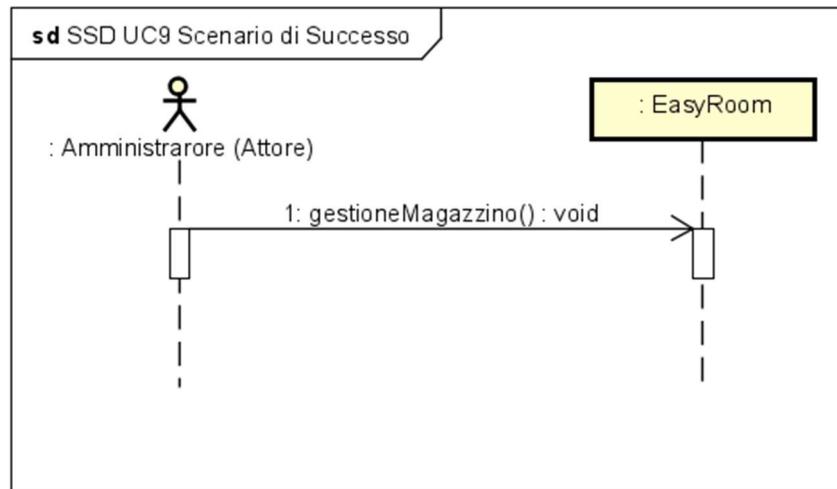
È stato ricavato il seguente Modello di Dominio:



UC9

Diagramma di sequenza di sistema

Diagramma di Sequenza di Sistema (SSD) per lo scenario del caso d'uso UC9:



In questo diagramma, il sistema permette all'Amministratore di gestire il magazzino. Il sistema permette di modificare il valore massimo di torce e indizi che possono essere affittati giornalmente, di modificare il costo di una singola attrezzatura. Il sistema permette di visualizzare per un dato giorno quanta attrezzatura sarà disponibile sulla base delle prenotazioni caricate in memoria.

Contratto delle operazioni

Di seguito viene indicato il Contratto delle operazioni per l'UC9:

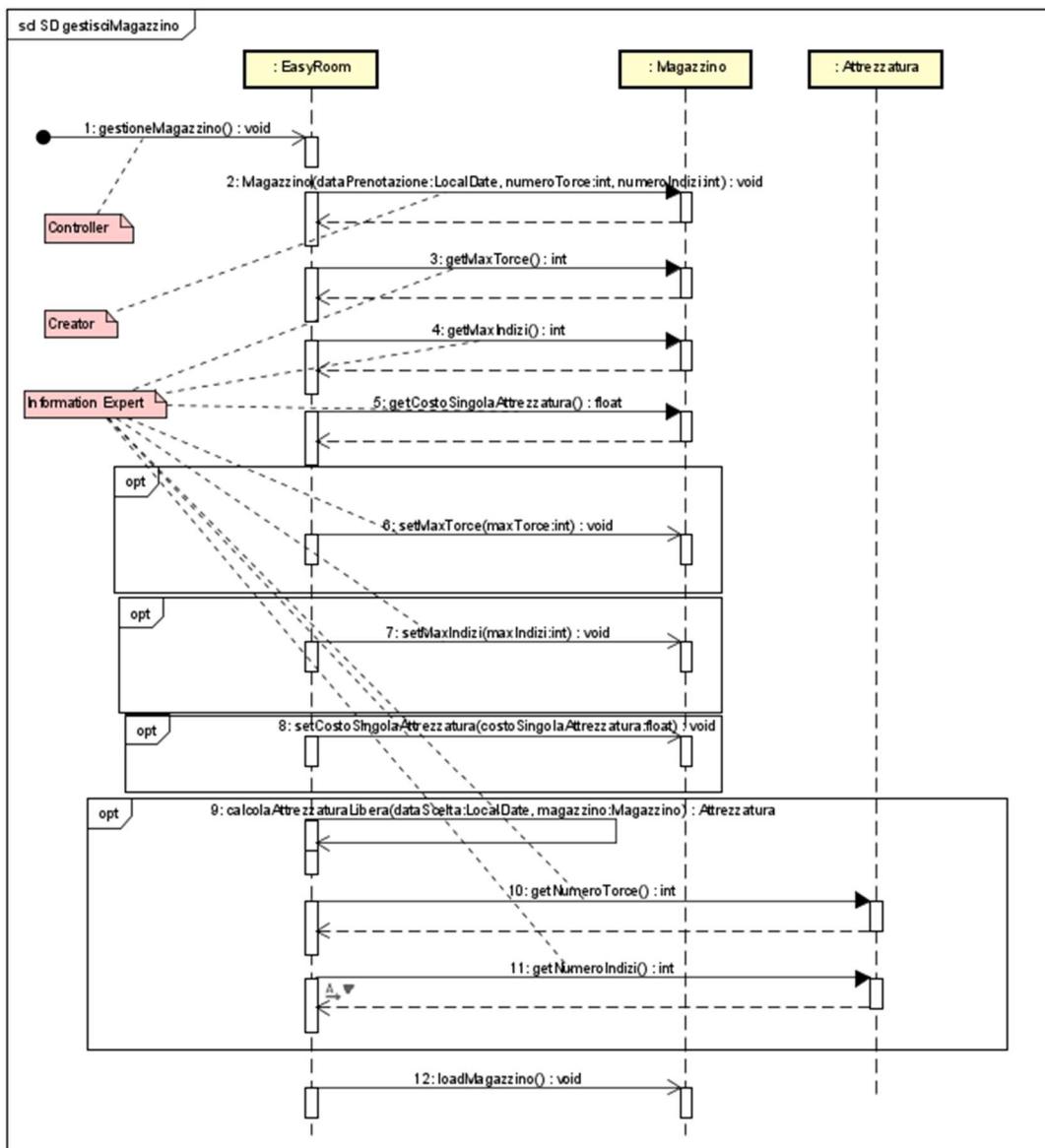
Operazione:	gestisciMagazzino()
Riferimenti:	Caso d'uso UC9: Gestione magazzino.
Pre-condizioni:	Devono essere caricate in memoria le informazioni relative al numero massimo di torce e indizi, oltre al costo della singola attrezzatura.
Post-condizioni:	I valori maxTorce, maxIndizi e costoSingolaAttrezzatura sono aggiornati correttamente e memorizzati in memoria in maniera persistente.

Operazione:	calcolaAttrezzaturaLibera(giornoPartita : Localdate, magazzino : Magazzino)
Riferimenti:	Caso d'uso UC9: Gestione magazzino.
Pre-condizioni:	Devono essere caricate in memoria le informazioni relative al numero massimo di torce e indizi.
Post-condizioni:	3. È stata creata l'istanza attrezzaturaDisponibile di Attrezzatura. 4. Gli attributi numeroTorce e numeroIndizi sono stati calcolati correttamente. attrezzaturaDisponibile è stato associato a EasyRoom.

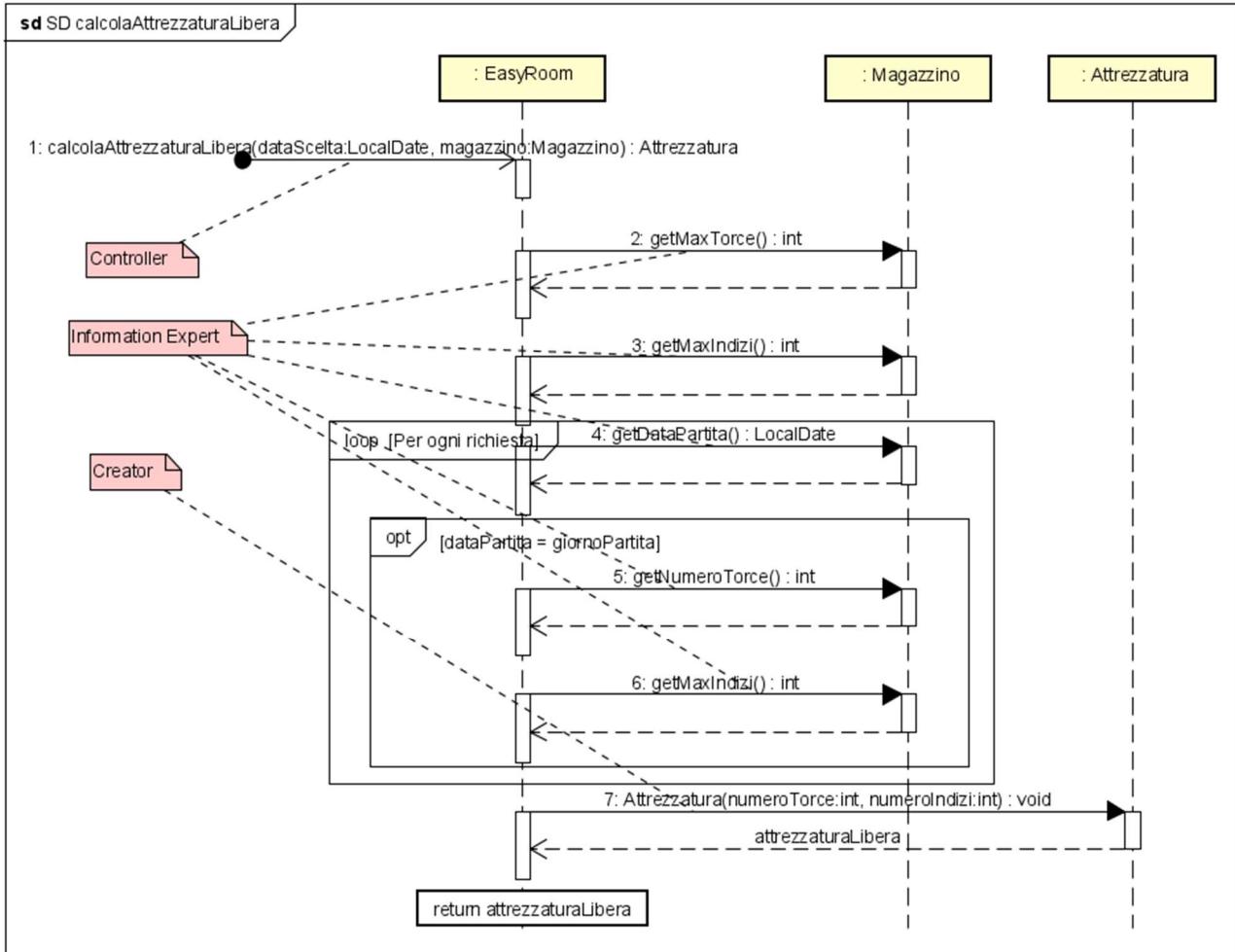
Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

gestioneMagazzino() : void



calcolaAttrezzaturaLibera(giornoPartita : Localdate, magazzino : Magazzino)



Modello delle classi di Progetto

