# 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:

Immagine che contiene testo, diagramma, Piano, linea

Descrizione generata automaticamente

# UC4

### Diagramma di sequenza di sistema

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

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

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:** | 1. È stata creata l’istanza nuovaStanza di Stanza. 2. 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:** | 1. È stata modificata l’istanza stanzaDaModificare di Stanza. 2. Gli attributi nome e prezzo sono stati aggiornati correttamente con valori validi che rispettano le regole di dominio.   stanzaDaModificare è stato associato a EasyRoom. Vengono aggiornati la lista mantenuta in memoria e i dati persistenti (file di testo). |

## Progettazione

Di seguito sono presenti i Diagrammi di Sequenza:

inserisciStanza() : void

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

modificaStanza() : void

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

# UC5

### Diagramma di sequenza di sistema

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

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

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

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

# UC6

### Diagramma di sequenza di sistema

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

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

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:** | checkin() |
| **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

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

# UC7

### Diagramma di sequenza di sistema

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

Immagine che contiene testo, schermata, Parallelo, linea

Descrizione generata automaticamente

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:** | checkout() |
| **Riferimenti:** | Caso d’uso UC7: Check-out stanza. |
| **Pre-condizioni:** | - |
| **Post-condizioni:** | Verrà memorizzata in maniera persistente l’istanza di Partita 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

Immagine che contiene testo, diagramma, Parallelo, numero

Descrizione generata automaticamente

# UC8

### Diagramma di sequenza di sistema

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

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

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:** | contaPartite(numeroStanza : int) |
| **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

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

## Modello delle classi di Progetto

Immagine che contiene testo, schermata, Rettangolo, Parallelo

Descrizione generata automaticamente