1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Corso di Laurea Triennale in Informatica Applicata

Programmazione e modellazione a Oggetti

Relazione del Progetto

Moonlight Hotel

Abatino Maria Rita
Matricola: 293171

Sessione Straordinaria Anno Accademico 2021/2022

Indice

Capitolo 1

Analisi

L'obiettivo del progetto è quello di implementare un'applicazione gestionale che permette di prenotare una camera d'albergo. In particolare, l'applicazione dovrà consentire ai clienti di effettuare una prenotazione scegliendo la camera tra quelle disponibili e di poter usufruire o meno dei servizi offerti dell'hotel in questione.

Funzionalità fornite dal progetto:

- Se sono presenti camere libere, il cliente dovrà inserire i suoi dati (nome, cognome, numero di adulti, numero di bambini e durata del pernottamento).
- Inseriti i dati, i programma verificherà che ci sia almeno una camera in grado di accogliere tutti gli ospiti;
- Se è libera almeno una camera adatta alle necessità del cliente il programma le, o la, mostra al cliente, che dovrà sceglierne una.
  Nell'elenco mostrato al cliente saranno specificati:
  - la tipologia di camera (Standard, Superior o Deluxe);
  - il numero dalla camera;
  - il costo.
- Una volta scelta la camera si aprirà una schermata che mostra tutti i servizi offerti dall'hotel. Il cliente potrà scegliere se usufruirne in totalità, di una parte degli stessi o di non usufruirne affatto. I costi dei servizi di cui il cliente sceglierà di beneficiare verranno aggiunti sulla base della tipologia di camera scelta.
- Uno dei servizi offerti è il servizio di noleggio bici, se il cliente sceglie di usufruirne dovrà decidere per quanti degli adulti della sua prenotazione noleggiarne una.
- Un altro servizio offerto è il servizio spa, se il cliente decide di avvalersi del servizio dovrà scegliere il giorno e il turno (tra quelli disponibili) in cui accedere alla spa.
- Se il cliente ha con sé dei bambini e nel caso in cui ci siano sufficienti posti liberi, potrà servirsi dell'area bimbi senza costi aggiuntivi.
- Successivamente, verrà mostrato un riepilogo della prenotazione del cliente, in cui verrà anche specificato il costo totale. In questa interfaccia il cliente potrà scegliere se confermare o la prenotazione oppure se iniziarne una nuova senza ultimare quella appena effettuata.

1.1 Requisiti

Requisiti funzionali

- L'applicazione permette l'inserimento dei dati personali del cliente, scegliendo il numero di adulti e di bambini tra quelli preimpostati.
- L'applicazione permette la scelta della camera tra quelle disponibili e in grado di accogliere tutti i componenti della prenotazione. Nel caso in cui non ci sia camere disponibili adatte verrà comunicato al cliente.
- L'applicazione permette al cliente di scegliere se beneficiare o meno dei servizi offerti.
- Nel caso in cui il cliente scelga di avvantaggiarsi del noleggio bici, il programma permetterà al cliente di decidere per quanti dei componenti adulti della prenotazione noleggiare una bicicletta.
- Nel caso in cui il cliente scelga di beneficiare della spa, il programma gli farò scegliere il giorno e il turno in cui accedervi.
- L'applicazione mostrerà al cliente un riepilogo della prenotazione, il quale potrà scegliere se ultimarla o meno. Quest'ultima opzione dà al cliente la possibilità annullare la prenotazione iniziandone una nuova oppure terminare l'esecuzione.

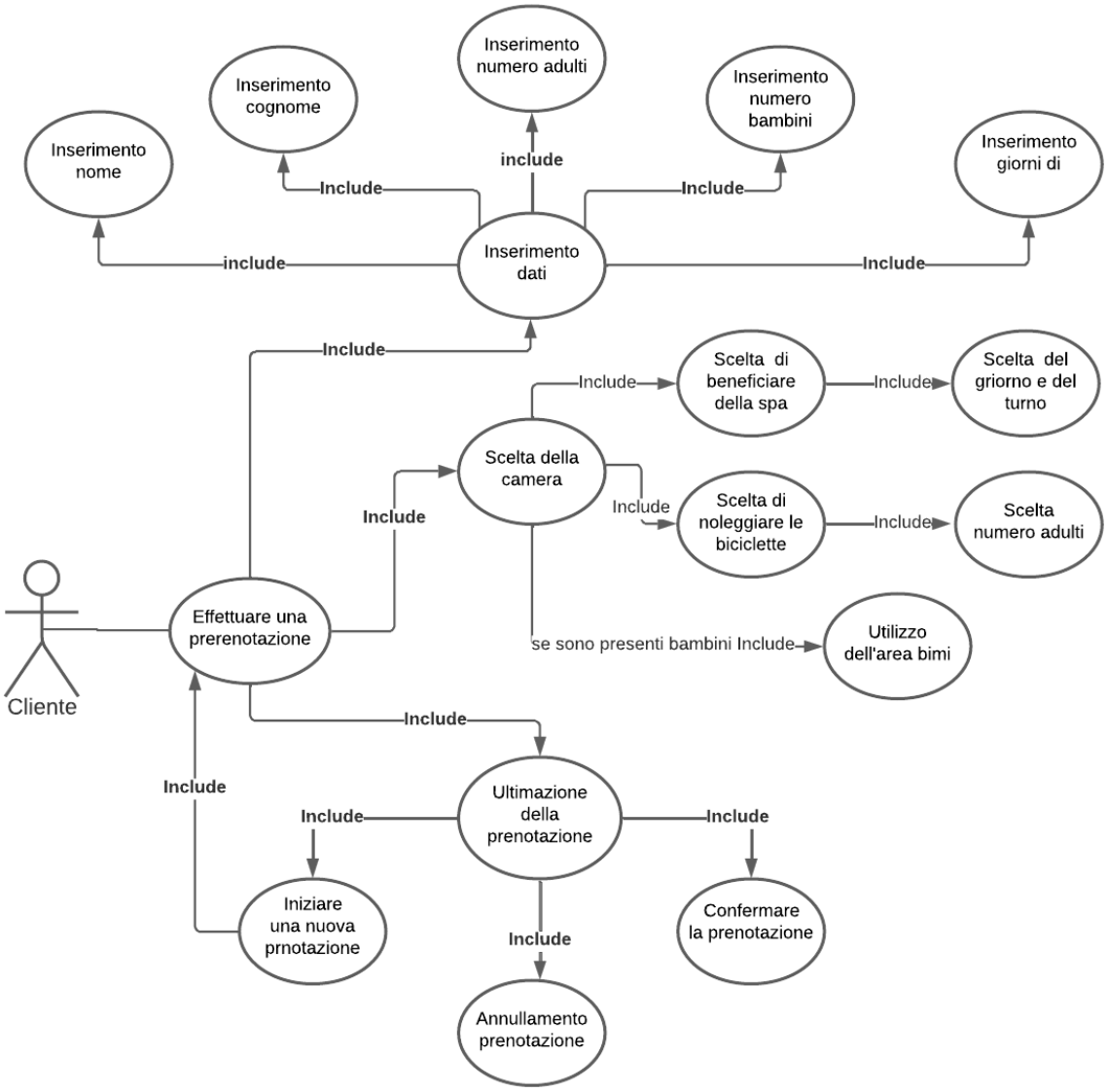
Requisiti non funzionali

- L'applicazione deve essere intuitiva e facile da usare per l'utente.
- L'applicazione permette al cliente di effettuare delle prenotazioni che soddisfano le loro esigenze.
- L'applicazione permette di ricominciare una nuova prenotazione senza confermare quella precedente, senza inficiare sulle prestazioni.

Casi d'uso

Di seguito vengono riportati i casi d'uso che permettono di individuare le interazioni degli utenti con il programma.

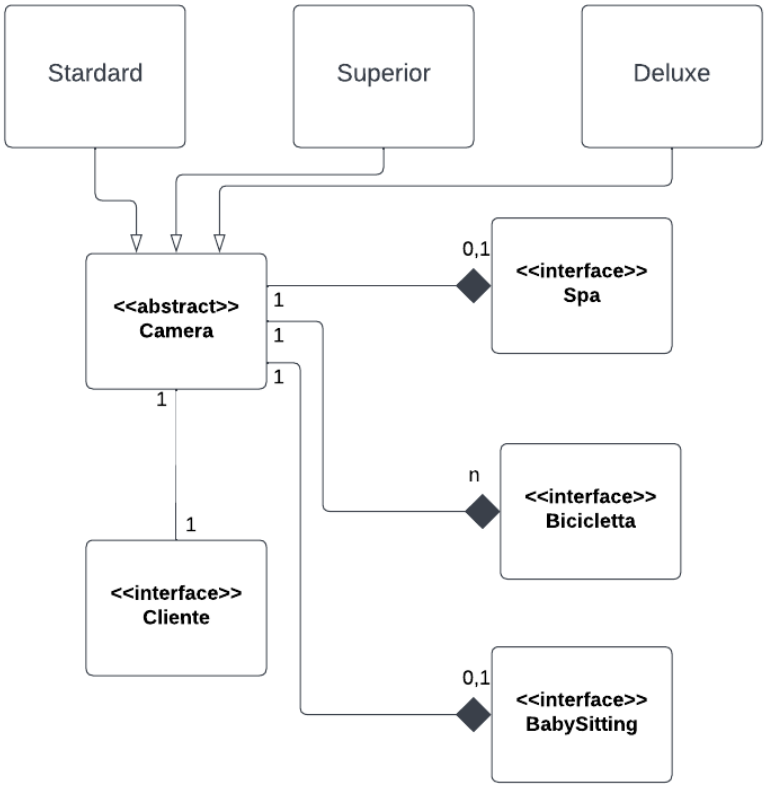
L'unico attore è il cliente, ed il caso d'uso mostrato mostra le opzioni per effettuare una prenotazione. Nella figura alcune opzioni minori sono integrate in altre.



1.2 Modello del dominio

Gli elementi principali del problema sono:

- Cliente, il quale effettua una prenotazione.
- Camera, che può essere di diverse tipologie.
- Camera Standard, tipologia di camera che non ha incluso nessun servizio.
- Camera Superior, tipologia di camera che ha incluso il noleggio bici.
- Camera Deluxe, tipologia di camera che ha incluso il noleggio bici e l'utilizzo della spa.
- Bicicletta, entità che può essere noleggiata dal cliente.
- Spa, entità di cui il cliente può scegliere se beneficiare o meno.
- Area bimbi, entità di cui il cliente può servirsi in ogni momento al di là della tipologia di camera scelta.


Stardard
Superior
Deluxe
<<abstract>>
Camera
0,1
<<interface>>
Spa
1
1
1
1
n
<<interface>>
Bicicletta
1
<<interface>>
Cliente
0,1
<<interface>>
BabySitting

Capitolo 2

Design

2.1 Architettura

L'applicazione è stata implementata utilizzando il pattern architetturale Model-View-Controller (MVC). Tale pattern è uno dei più diffusi nella programmazione ad oggetti, in quanto permette di dividere gli aspetti logici del programma da quelli interattivi. In particolare, i componenti principali sono:

- Model: si occupa della gestione dei dati;
- View: si occupa dell'interazione dell'utente;
- Controller: sulla base dell'interazione con l'utente modifica lo stato degli altri componenti.

2.1.1 View

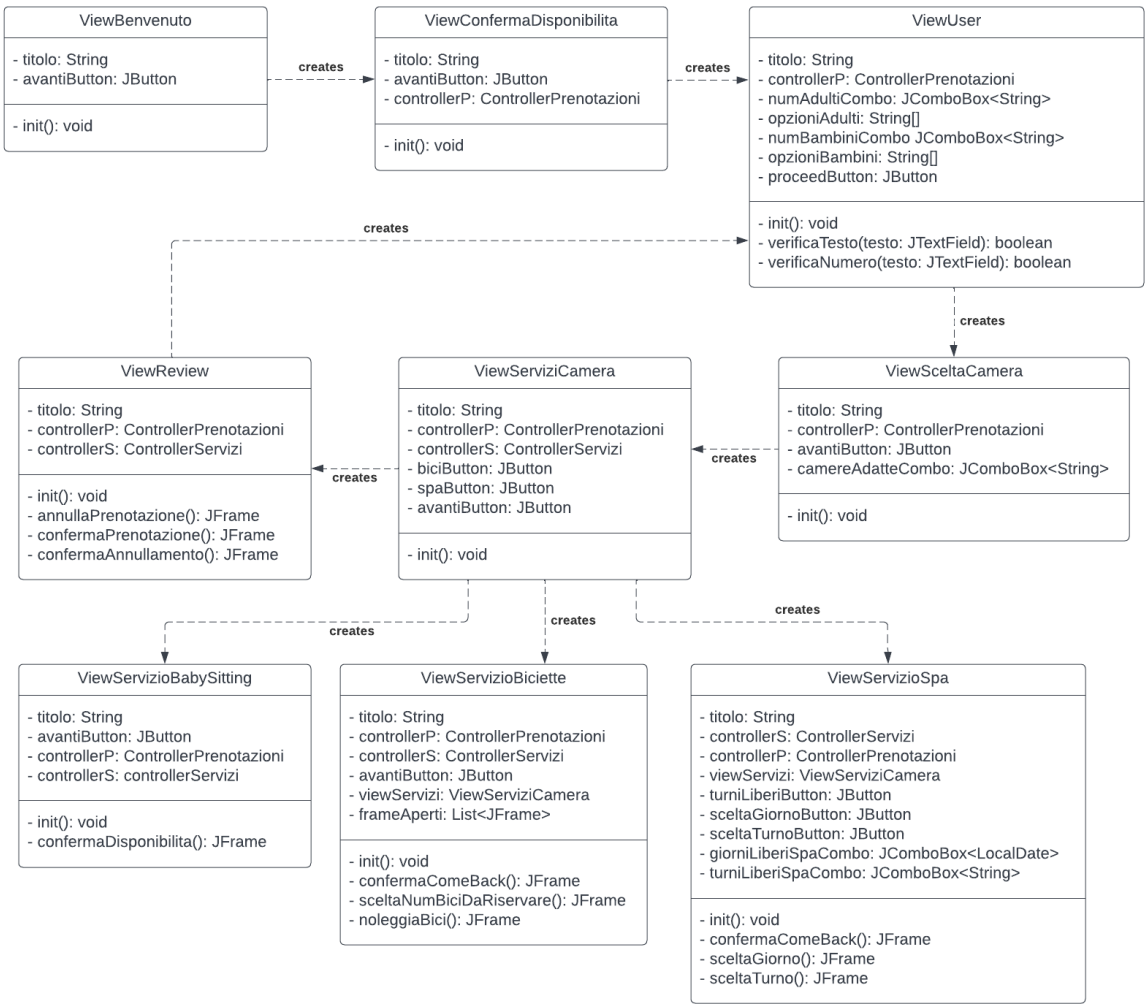
Il view si occupa dell'interazione con l'utente e della visualizzazione dei dati contenuti nel model. Nel programma realizzato l'interazione con l'utente avviene tramite interfaccia grafica, realizzata tramite la libreria Java Swing.

In base alle necessità dell'applicazione sono state create diverse view ognuna per uno step della prenotazione. In particolare:

- `ViewBenvenuto`: fornisce al cliente un primo benvenuto dando le informazioni riguardanti le camere e i relativi costi, sia per adulti che per bambini.
- `ViewConfermaDisponibilita`: comunica al cliente che nell'hotel sono presenti camere libere, ma che prima di mostrargliele è necessario acquisire qualche informazione per verificare che rispecchino i suoi bisogni.
- `ViewReview`: fornisce al cliente un riepilogo della prenotazione effettuata e diverse opzioni:
  - Confermare la prenotazione;
  - Annullare la prenotazione;
  - Ricominciare perdendo tutti i dati della prenotazione appena effettuata.
- `ViewSceltaCamera`: mostra al cliente tutte le camere disponibili che soddisfano le sue esigenze e gli fornisce la possibilità di scegliere la camera che più gli aggrada.
- `ViewServiziCamera`: fornisce al cliente di beneficiare o meno dei servizi offerti dall'hotel indipendentemente dalla tipologia di camera scelta.
- `ViewServizioBabySitting`: fornisce al cliente che effettua una prenotazione con bambini la possibilità di utilizzare l'area bimbi indipendentemente dalla tipologia di camera

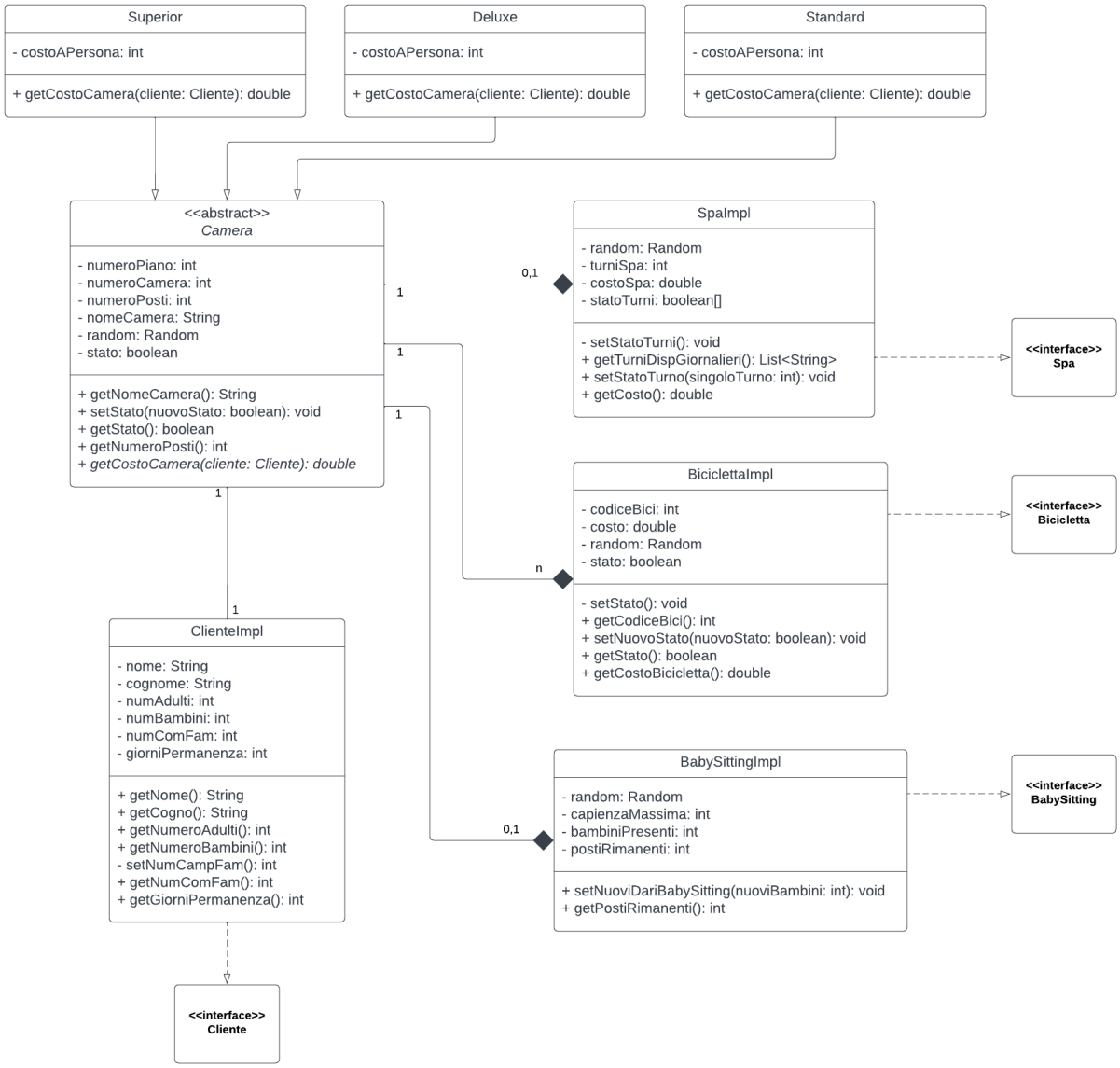
scelta. Solo nel caso in cui non ci siano sufficienti posti liberi nell'area bimbi il cliente non potrà utilizzare il servizio.

- `ViewServizioBiciclette`: fornisce al cliente tutte le informazioni riguardanti il servizio e la possibilità di scegliere per quanti adulti sella sua prenotazione noleggiare le biciclette. Il cliente potrà tornare indietro e non usufruire del servizio in ogni momento.
- `ViewServizioSpa`: fornisce al cliente tutte le informazioni riguardanti il servizio e la possibilità di scegliere il giorno e il turno (tra quelli disponibili) in cui beneficiare della spa. Il cliente potrà tornare indietro e non usufruire del servizio in ogni momento.
- `ViewUser`: acquisisce le informazioni personali del cliente come: nome, cognome, numero di adulti, numero di bambini e durata del pernottamento.



2.1.2 Model

Il model si occupa di fornire al controller una rappresentazione corrente dello stato interno dell'applicazione e dell'accesso ai dati. Le classi del model sono state definite e implementate sulla base dell'analisi di dominio.

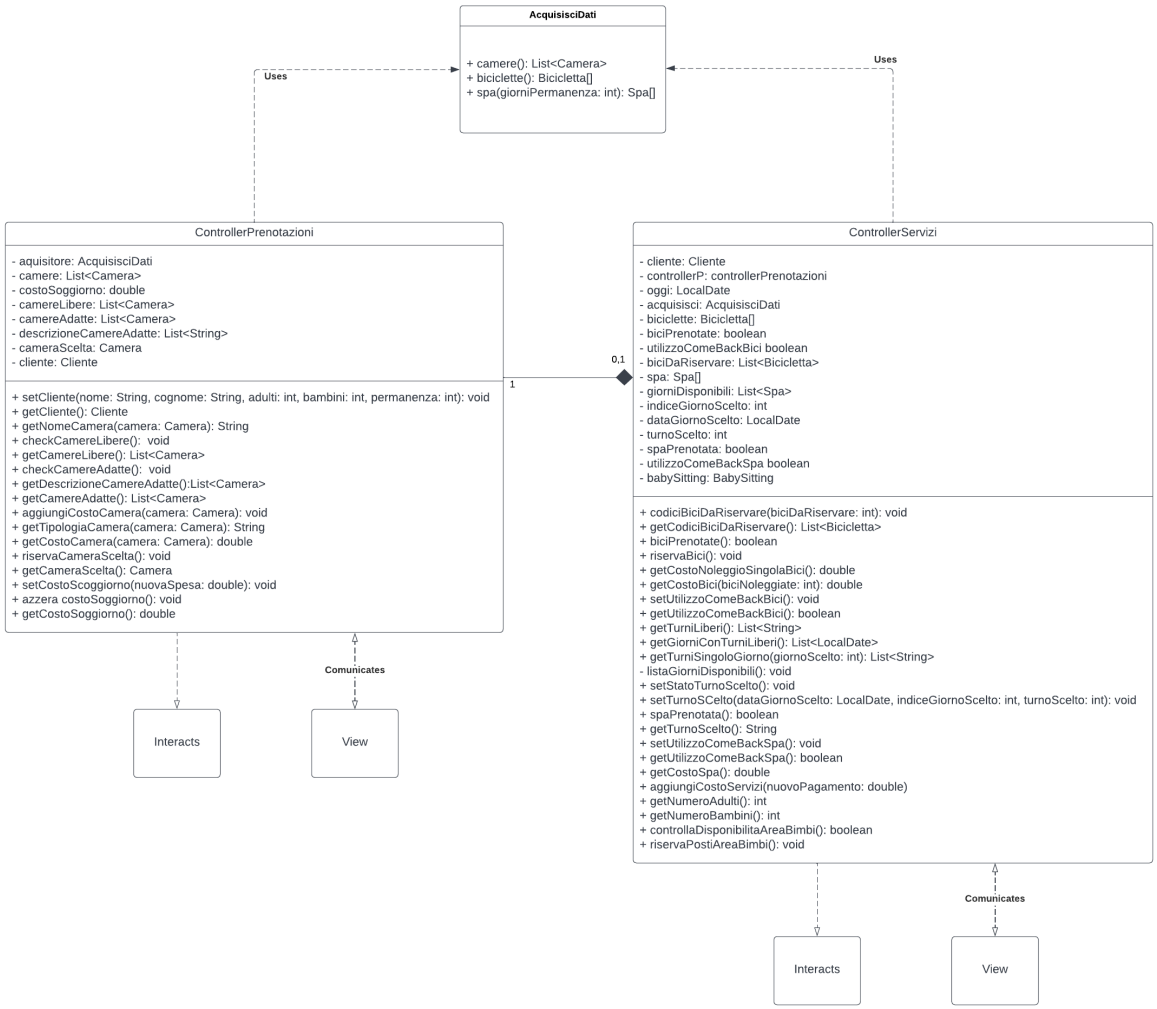


2.1.3 Controller

Il controller si occupa di acquisire le informazioni dall'utente tramite le view e a seconda dei dati raccolti cambiare lo stato del model.

In base alle necessità del programma implementato sono state create più classi di gestione:

- `ControllerPrenotazioni`: ha il compito di gestire le camere e di inizializzare il cliente.
- `ControllerServizi`: ha il compito di gestire i servizi offerti dall'hotel.



2.1.4 Punti d'accesso

I punti di accesso al componenti del pattern MVC sono:

- Model: le interfacce del modello e la classe astratta `Camera` con le relative estensioni `Superior`, `Standard e Deluxe` sono nel package `moonlightHotel.model`, mentre le classi del modello sono contenute nel package `moonlightHotel.model.impl`.
- View: le classi del view sono contenute nel `package moonlightHotel.view`.
- Controller: le classi del controller sì trovano nel package `moonlightHotel.controller`.

2.2 Design dettagliato

Pulizia del main, dichiarazione e inizializzazione dei controller

L'intero programma ha origine dall'invocazione della `ViewBenvenuto` presente nel main. Questa scelta è stata presa con l'obiettivo di rendere il main il più pulito possibile. In particolare, ogni view darà origine ad una view sulla base delle decisioni prese dal cliente.

All'interno della view di benvenuto viene dichiarato il `ControllerPrenotazioni` e inizializzato con l'elenco di tutte le camere presenti nell'hotel. Successivamente, il controller, effettuerà una ricerca per individuare le camere libere.

Il `ControllerServizi`, che si occupa della gestione dei servizi, viene dichiarato e inizializzato dopo la scelta della camera da parte del cliente. Il controller dei servizi viene istanziato con il controller delle prenotazioni, in modo che in ogni momento possa ottenere la camera scelta dal cliente e il cliente stesso.

Possibilità di scelta

Nel realizzare il programma ho scelto di dare al cliente il massimo delle possibilità di movimento, nel dettaglio nella `viewServiziCamera` vengono mostrati tre pulsanti:

- Il primo corrisponde al pulsante guida al noleggio delle biciclette. Inizialmente vengono mostrate le informazioni del servizio e il cliente potrà scegliere il numero di bici che vuole noleggiare.
  Fino a quando il cliente non confermerà il numero di bici scelto potrà tornare alla `viewServiziCamera` senza che il programma risenta della percorso effettuato dal cliente. L'unico vincolo che viene imposto è la non possibilità di poter prenotare le biciclette una volta che si è già tornati indietro, in ogni caso il programma comunica al cliente ogni opzione non reversibile.
- Il secondo corrisponde al pulsante guida alla prenotazione di un turno nella spa. Inizialmente vengono mostrate le informazioni del servizio e il cliente potrà scegliere il giorno in cui prenotare il turno (nel caso in cui cambi idea potrà tornare indietro e cambiare giorno). Successivamente gli verrà mostrata un'ulteriore schermata in cui potrà scegliere il turno da prenotare.
  Fino a quando il cliente non confermerà il turno scelto potrà tornare alla `viewServiziCamera` senza che il programma risenta della percorso effettuato dal cliente. L'unico vincolo che viene imposto è la non possibilità di poter prenotare le biciclette una volta che si è già tornati indietro, in ogni caso il programma comunica al cliente ogni opzione non reversibile.
- Il terzo corrisponde al pulsante per andare avanti, il quale verrà utilizzato dal cliente solo nel caso in cui non voglia usufruire di almeno un servizio.

Nel caso in cui il cliente scelga di usufruire di entrambi i servizi oppure di solo uno e dell'altro scelga di torna indietro senza confermare oppure di tornare indietro senza confermare per entrambi i servizi verrà in automatico mostrata la view successiva che sarà `viewServizioBabuSitting`, nel caso in cui nella prenotazione ci siano bambini o `viewReview` nel caso contrario.

Di seguito viene mostrata la l'implementazione di tutti i controlli all'interno della `viewServizioSpa`:

```
if(controllerS.getUtilizzoComeBackBici() && controllerS.getUtilizzoComeBackSpa()) {
    if(controllerS.getNumeroBambini()>0) {
        new ViewServizioBabySitting(controllerS, controllerP);
    } else {
        new ViewReview(controllerP, controllerS);
    }
} else if (controllerS.getUtilizzoComeBackSpa() && controllerS.biciPrenotate()){
    if(controllerS.getNumeroBambini()>0) {
        new ViewServizioBabySitting(controllerS, controllerP);
    } else {
        new ViewReview(controllerP, controllerS);
    }
} else {
    viewServizi.setVisible(true);
}
```

Di seguito viene mostrata la l'implementazione di tutti i controlli all'interno della `viewServizioBiciclette`:

```
if(controllerS.getUtilizzoComeBackBici() && controllerS.getUtilizzoComeBackSpa()) {
    if(controllerS.getNumeroBambini()>0) {
        new ViewServizioBabySitting(controllerS, controllerP);
    } else {
        new ViewReview(controllerP, controllerS);
    }
} else if (controllerS.getUtilizzoComeBackBici() && controllerS.spaPrenotata()){
    if(controllerS.getNumeroBambini()>0) {
        new ViewServizioBabySitting(controllerS, controllerP);
    } else {
        new ViewReview(controllerP, controllerS);
    }
} else {
    viewServizi.setVisible(true);
}
```

Sempre con lo scopo di dare al cliente la maggior possibilità di movimento nella `viewReview` oltre ad un riepilogo dettagliato della prenotazione sono presenti tre pulsanti:

- Quello più a sinistra consente al cliente di annullare la prenotazione effettuata.
- Quello più a destra consente al cliente di confermare la prenotazione effettuata. Solo dopo aver confermato la prenotazione la camera scelta e gli eventuali servizi verranno effettivamente riservati.
- Quello centrale consente al cliente, che magari non è soddisfatto delle prenotazione effettuata, di cominciarne una nuova portandolo ad una nuova `viewUser`. In ogni caso le camere libere sono sempre quelle a meno che non venga iniziata una nuova esecuzione.

  Inserire questo pulsante ha evidenziato la necessità di ripulire delle liste usate precedentemente tramite il metodo clear() al fine di eliminare elementi ridondanti nel menù a tendina presente nella `viewSceltaCamera`. Il metodo viene utilizzato

all'interno del Controller delle prenotazioni, ed in particolare dove viene riportato di seguito:

```
// metodo che verifica se ci sono camere libere che rienstrano nel budget
public void checkCamereAdatte(){

    String descrizione;

    // svuoto la lista delle camere adatte a quella delle descrizioni
    this.camereAdatte.clear();
    this.descrizioneCamereAdatte.clear();

    // cerco le camere
    for(int i = 0; i < this.camereLibere.size(); i++) {
        if( (this.camereLibere.get(i).getNumeroPosti() >= this.cliente.getNumCompFam()) ) {
            this.camereAdatte.add(this.camereLibere.get(i));
            descrizione = this.getTipologiaCamera(this.camereLibere.get(i))
                    +" "+this.camereLibere.get(i).getNomeCamera()
                    +" a "
                    +this.camereLibere.get(i).getCostoCamera(this.cliente)
                    +"€";
            this.descrizioneCamereAdatte.add(descrizione);
        } // fine if
    } // fine for

}
// fine metodo
```

Inoltre, è stato necessario aggiungere un metodo che azzerasse il costo del soggiorno precedente Questo perché a differenza della camera o degli eventuali servizi che vengono riservati solo dopo aver confermato la prenotazione, il costo viene calcolato man mano. L'aggiunta di questo semplice metodo è stata evidente dopo la fase di testing che mostrava costi troppo elevati.

JComboBox nelle View

Nella `viewUser` per definire il numero di adulti e il numero di bambini ho utilizzato delle JComboBox, in modo da poter rendere prenotabile una camera che riesca ad accogliere tutti i membri della prenotazione.

Anche nella `viewServizioBiciclette` ho utilizzato una JComboBox per far scegliere al cliente quante biciclette noleggiare. Le opzioni presenti nella JComboBox sono definite sulla base del numero di adulti presenti.

Anche nella `viewServizioSpa` ho utilizzato una JComboBox per far scegliere al cliente il giorno in cui prenotare il turno nella spa. Le opzioni presenti nella JComboBox sono date definite sulla base della durata del pernottamento del cliente. Una volta scelto il giorno verrà mostrata un'altra schermata che dà al cliente la possibilità di scegliere il turno nel giorno specificato. Persino in questo caso ho utilizzato una JComboBox i cui elementi sono definiti sulla base dei turni disponibili nel giorno scelto.

Utilizzo della classe Random

Il programma è stato implementando in modo da simulare l'ultimazione di al più una prenotazione, pertanto, per simulare un hotel con già diverse prenotazioni ho scelto di utilizzare la classe `Random` per inizializzare lo stato di camere, biciclette, turni nella spa e posti liberi nell'area bimbi.

Nel particolare:

- Lo stato di tutte le camere viene definito con l'utilizzo della classe `Random`;
- Lo stato delle biciclette viene definito tramite l'utilizzo della classe `Random` fino alla 17° bicicletta. Questa scelta è stata presa sulla base della necessità di avere delle biciclette libere nel caso in cui venga prenotata una camera con 5 ospiti adulti e servano sufficienti biciclette.
- Lo stato dei turni della spa viene definito tramite l'utilizzo della classe `Random`;
- Lo stato dei posti liberi nell'area bimbi viene definito in parte tramite l'utilizzo della classe `Random`.

Problematiche spa

Come detto sopra lo stato dei turni della spa viene definito tramite l'utilizzo della classe `Random`, questo potrebbe causare giorni in cui non sono presenti turni disponibili. I giorni in questione non devono essere inseriti come opzione durante la scelta del giorno.

Inizialmente, quando si verificava un caso in cui un giorno non aveva turni liberi e veniva scelto un giorno successivo ad esso si verificava un eccezione ArrayList index out of bounds exception.

Successivamente per risolvere il problema ho deciso di creare un metodo nel ControllerServizi che inizializzasse una lista di Spa che contenesse solamente giorni con turni disponibili. Utilizzando questa lista sono riuscita a trovare in ogni caso i turni corretti senza trovare eccezioni.

```
// metodo privato per ottenere una lista dei giorni con i turni liberi della spa
private void listaGiorniDisponibili(){

    for(int i = 0; i < this.cliente.getGiorniPermanenza(); i++) {
        if(this.spa[i].getTurniDispGiornalieri().size() > 0) {
            this.giorniDisponibili.add(this.spa[i]);
        }
    }
}
// fine metodo
```

Posizionamento AcquisisciDati

Un'altra scelta importante di implementazione è stata quella di inserire nel `package moonlightHotel.controller` anche la classe `AcquisisciDati` che non è una classe di gestione a differenza delle `ControllerServizi` e `ControllerPrenotazioni` che, come già specificato, sono rispettivamente i gestori dei servizi e delle camere. La scelta di questo posizionamento della classe `AcquisisciDati` è dovuto al fatto che non rientra negli elementi essenziali del modello e che è una classe che permette solamente di inizializzare nei controller le camere, le biciclette e la spa in maniera "nascosta" e quindi più "pulita".

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Il testing automatizzato è stato effettuato tramite l'utilizzo di JUnit 5, in modo tale da poter verificare il comportamento di diversi metodi in contesti differenti e di confrontare tale comportamento con ciò che ci aspetta in quel contesto.

I test venivano scritti in maniera parallela al codice, così da poter rilevare nell'immediato eventuali errori o problematiche e da poterli risolvere subito.

3.2 Metodologia di lavoro

L'intero lavoro è stato svolto in autonomia. Il primo step è stato quello di definire la specifica adatta, cercando di valutare a priori quali potessero essere le difficoltà che avrei affrontato e se potesse rispettare i requisiti di progetto adatto.

Una volta trovata la specifica che più mi soddisfaceva e che è stata approvata dalla docente, ho affrontato un'attenta analisi per definizione la struttura dell'applicazione tramite grafici UML.

In particolare, ho studiato i casi d'uso in modo da poter evidenziare tutte le interazioni tra utente e programma. Successivamente, tramite lo studio del dominio dell'applicazione ho individuato le entità principali e le relazioni intercorrenti tra loro.

Aver evidenziato le interazioni tra utente e programma ha facilitato l'individuazione delle interfacce che avrei dovuto sviluppare per poter ottenere un programma facilmente comprensibile e facilmente utilizzabile.

Ogni volta che nasceva una nuova problematica è stata sempre subito affrontata e mai lasciata perdere, così da evitare di dimenticarli o eventuali peggioramenti del problema.

Avendo svolto tutto da sola alcune parti hanno richiesto diverso tempo e riflessione, ma ho sempre cercato di progredire in maniera logica evitando di dimenticare delle parti, magari anche piccole.

3.3 Note di sviluppo

Nel programma sono stati utilizzati i seguenti aspetti avanzati di implementazione:

- Espressioni lambda, nella creazione di eventListener per tutti i JButton presenti in tutte le view.