



UNIVERSITÀ DEGLI STUDI DI PALERMO  
SCUOLA DI SCIENZE E BASI APPLICATE  
FACOLTA' DI INFORMATICA  
CORSO DI METODI AVANZATI PER LA PROGRAMMAZIONE  
A.A. 2019-2020

---

# Book Your Resort

---

**Membri del progetto:**

Andrea di Benedetto 0674265

Maria Assunta Sciortino 0674630

# Sommario

<b>Abstract .....</b>	<b>3</b>
<b>1. Introduzione .....</b>	<b>4</b>
<b>2. Requisiti funzionali .....</b>	<b>7</b>
<b>3. Requisiti non funzionali .....</b>	<b>8</b>
<b>4. Casi d'uso .....</b>	<b>9</b>
<b>5. UML – Diagramma delle classi.....</b>	<b>14</b>
<b>6. Design Pattern .....</b>	<b>15</b>
6.1. Flyweight Pattern .....	15
6.1.1. Factory Pattern .....	18
6.2. Decorator Pattern .....	19
6.2.1. Observer Pattern.....	21
6.3. Strategy Pattern .....	24
<b>7. Guida all'uso e Mock up .....</b>	<b>26</b>
<b>8. Conclusioni .....</b>	<b>35</b>
<b>Referenze.....</b>	<b>35</b>

## **ABSTRACT**

Abbiamo implementato un software che simula la gestione di una prenotazione in un Resort di lusso. Per fare ciò abbiamo sfruttato metodi di programmazione avanzata del linguaggio Java quali i Design Pattern e alcune tecniche di Java avanzato, insieme ad alcuni moduli che abbiamo integrato quali l'invio della mail e la generazione della fattura del cliente a prenotazione avvenuta.

## 1. Introduzione

*Book Your Resort* è un software realizzato per l'esame di *Metodi avanzati per la programmazione* che mira a dimostrare competenze apprese in merito a tecniche avanzate di programmazione come i Design Pattern.

Di seguito l'elenco dei design pattern usati:

- Flyweight
- Factory
- Decorator
- Observer
- Strategy

Inoltre, l'uso di *java.awt* e di *Swing* per la realizzazione dell'interfaccia grafica ci ha permesso di utilizzare implicitamente il *Composite Pattern* di cui *Swing* è composto. Tuttavia, i dettagli di utilizzo di ogni singolo design pattern verranno approfonditi nei paragrafi successivi.

Il presente programma gestisce la prenotazione, da parte di vari utenti, di un soggiorno in un resort di lusso. Esso permette all'utente di scegliere: la durata del soggiorno, la tipologia di villa dove si desidera pernottare e le attività da svolgere.

La durata del soggiorno è da intendersi come numero di giorni di pernottamento; le tipologie di ville disponibili sono due, una con piscina e una senza piscina, la cui scelta influenzerà le attività da poter svolgere.

Chi sceglierà la *villa senza piscina* potrà svolgere le seguenti attività:

Holistic Massage, Pilates, Zumba, Golf, Excursion, Yoga.

Mentre chi preferirà la tipologia di *villa con piscina*, oltre a poter scegliere le attività sopra citate, potrà svolgere anche le attività di: Acquagym, Hydrobike, Aqua Tabata, Woga, Swimming Pool Games.

Sarà permesso all'utente di pagare il soggiorno tramite due metodi di pagamento, ovvero carta di credito e paypal, e di avere tramite e-mail, la fattura del suddetto pagamento.

Questo programma presenterà un'interfaccia iniziale, che richiederà l'inserimento della data di check-in e di check-out.

Successivamente, l'utente potrà selezionare la villa di suo interesse e vedere, come conseguenza della selezione, una griglia che illustrerà quali ville sono state già prenotate e quali risultano ancora disponibili.

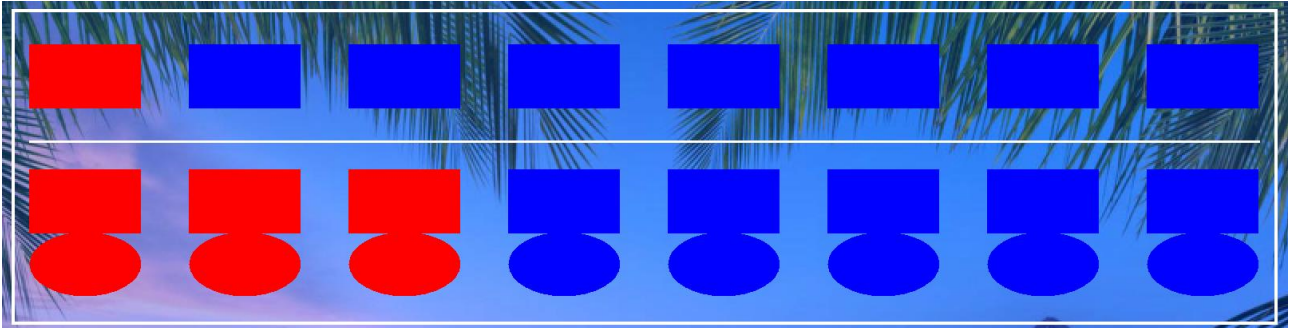


Figura 1: Griglia prenotazioni (colore rosso=villa prenotata, colore blu=villa libera)

L'utente quindi potrà cliccare su *Follow* o *Book*.

Cliccando su *Follow*, riceverà una mail ogni qual volta un altro utente, avrà prenotato una villa della stessa tipologia e avrà anche informazioni circa quante ville di quella tipologia rimangono disponibili; mentre cliccando su *Book*, verrà inviata una mail agli utenti interessati, diminuiranno, inoltre, le ville disponibili di quella tipologia e di conseguenza verrà aggiornata la griglia delle ville. Si potrà quindi procedere sia all'inserimento dei propri dati anagrafici e sia alla scelta del metodo di pagamento. Terminata la fase di pagamento, l'utente riceverà la fattura tramite e-mail, dove verranno mostrati tutti i dettagli del soggiorno.

Durante l'implementazione del software, abbiamo riscontrato delle problematiche legate soprattutto alla gestione e allo studio delle librerie esterne utili per espletare determinate funzionalità.

La libreria *itextpdf* è stata utile per la generazione della fattura, mentre l'invio della stessa è stata gestita dalla libreria *javax.mail*, con la quale abbiamo avuto problemi causati dall'accesso al server, in quanto richiedeva la messa in chiaro della password. I server di Google non hanno permesso l'accesso facendo ricadere la scelta di utilizzo verso i server di Libero.

Importantissima e poco banale è stata la gestione delle date che dovevano risultare disponibili all'utente solo se all'interno del range di date da lui scelto, determinato dal check-in e dal check-out, non vi erano delle ville disponibili inerente alla tipologia di villa scelta. A tal fine abbiamo studiato e utilizzato un algoritmo che permette tutto ciò. Di seguito viene mostrato lo pseudocodice:

$$(\text{start1} < \text{end2} \text{ and } \text{start2} < \text{end1})$$

Il range start2-end2 corrisponde all'intervallo di date di check-in e check-out scelto dall'utente, mentre il range start1-end1 corrisponde all'intervallo di date già occupate. Se tale condizione viene verificata, vuol dire che gli intervalli si sovrappongono.

Prima di quest'ultima problematica, abbiamo dovuto risolvere quella riguardante la sovrapposizione sinistra e destra tra i range di date scelte dall'utente e occupate includendo gli estremi dell'intervallo.

Per una corretta visualizzazione delle date ci siamo serviti della libreria *jCalendar* messa a disposizione dalla community di java.

La gestione delle date che abbiamo considerato si collega direttamente alla corretta visualizzazione sulla griglia delle ville che corrisponde al numero di ville libere e al numero di ville occupate. Di conseguenza ci siamo serviti del *Flyweight Pattern* e della libreria *Graphics* per una corretta visualizzazione della suddetta.

## 2. Requisiti funzionali

Come è già stato introdotto precedentemente, dopo aver avviato il programma l'utente deve inserire le date di check-in e di check-out. In questo pannello vengono effettuate delle operazioni di controllo sulle date inserite dall'utente in cui viene visualizzato un messaggio di popup in base al tipo di errore che l'utente ha commesso. Tali eventuali messaggi di popup vengono scaturiti dal click sul bottone "Next".

Nel pannello successivo l'utente deve inserire la tipologia di villa a cui è interessato sia nel caso in cui voglia diventare semplicemente un follower che nel caso in cui voglia effettuare la prenotazione. Viene mostrato un messaggio di popup nel caso in cui non effettua alcuna selezione.

Dopo aver effettuato le proprie scelte dovrà selezionare le attività che gli verranno mostrate sul pannello delle attività. In questo caso non vi sono obblighi di scelta da parte dell'utente in quanto può anche decidere di non partecipare ad alcuna tra le attività proposte e procedere con la prenotazione.

Il pannello successivo prevede l'inserimento dei propri dati anagrafici necessari per l'immissione dei dati nel sistema. In questo pannello vi è un sommario sulla destra in cui viene mostrato il riepilogo delle informazioni inserite dall'utente. Qualora l'utente desiderasse modificare le proprie informazioni lo potrà effettuare cliccando nuovamente sul bottone "Submit" dopo aver terminato l'immissione delle nuove informazioni.

L'ultimo pannello consente di scegliere la metodologia di pagamento e di inserire le informazioni richieste in base alla scelta effettuata. Il sistema non consente di proseguire al pagamento qualora l'utente non immette tutte le informazioni richieste. A questo punto l'utente dovrà cliccare su "Confirm" per confermare il pagamento ed infine su "Yes". Il sistema elaborerà il pagamento e invierà la mail con la fattura al suddetto utente che ha effettuato la prenotazione e le mail ai follower che hanno scelto di seguire quella tipologia di villa, se presenti.

### 3. Requisiti non funzionali

Nel momento in cui l'utente immette la data di check-in e di check-out il sistema salva momentaneamente le date che sono state inserite. L'utente è libero a quel punto di passare al pannello successivo e poi di ritornare al precedente per effettuare un'eventuale modifica. Ciò significa che l'interazione con i bottoni è determinante per il salvataggio delle informazioni da parte del sistema.

L'utente in base alla tipologia di villa che sceglie consentirà al sistema di poter visualizzare le attività corrispondenti alla scelta nel pannello delle attività. Se clicca sul bottone per tornare indietro (dal pannello delle attività al pannello della scelta del tipo di villa), dovrà cliccare nuovamente sul tipo di villa che è interessato per una corretta visualizzazione delle ville disponibili e di quelle occupate.

L'interazione con il bottone "Submit" nel pannello contenente il form per inserire i propri dati anagrafici è necessaria per l'inserimento momentaneo dei dati all'interno del sistema. Tali dati verranno salvati in maniera effettiva sul file csv relativo ai customers, nel momento in cui l'utente ha concluso l'operazione di pagamento. Se l'utente prova a ritornare indietro dal pannello del pagamento al pannello contenente i dati del cliente, il sistema obbliga l'utente a cliccare nuovamente sul pulsante "Submit" per prevedere un'eventuale modifica da parte di quest'ultimo e, quindi, per la reimmissione momentanea dei dati nel sistema.

Dopo aver completato la procedura di pagamento il sistema salva le informazioni dell'utente nel file csv che simula il database (eliminando la sua e-mail come follower) e blocca l'interazione con i bottoni di tale pannello in quanto, l'utente ha effettuato tutte le operazioni richieste e può proseguire con la chiusura del programma.



## 4. Casi d'uso

Nome del Caso d'uso	Check-in Check-out
Attori partecipanti	Utente
Condizione di entrata	L'utente avvia il programma.
Flusso di eventi	<p>1) L'utente aprendo il programma avrà accesso alla schermata in cui può inserire la data di check-in e di check-out</p> <p>2) Una volta che l'utente ha selezionato le date corrispondenti può passare alla schermata successiva tramite il bottone "Next"</p>
Condizione di Uscita	L'utente ha cliccato su "Next" e passa alla schermata successiva

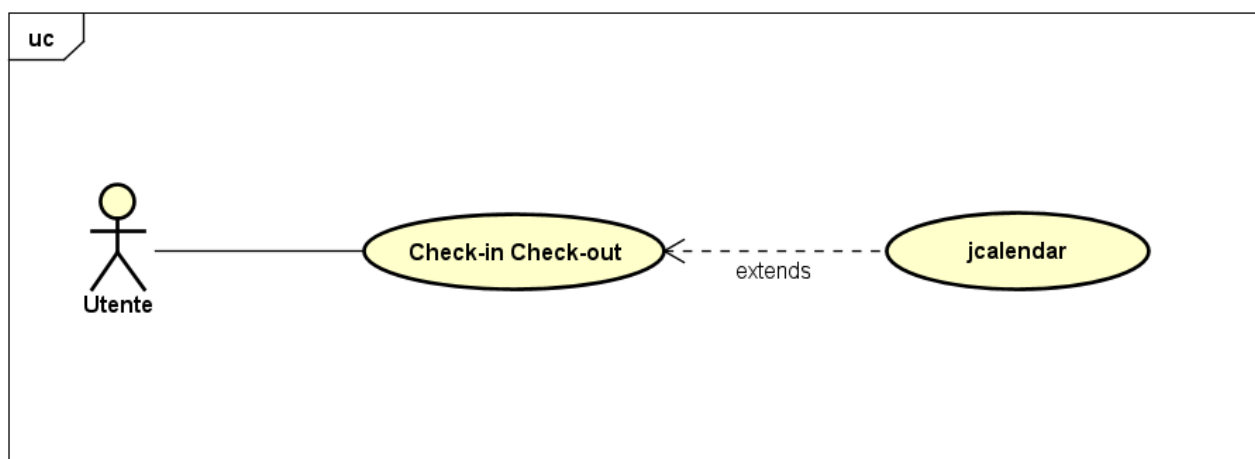
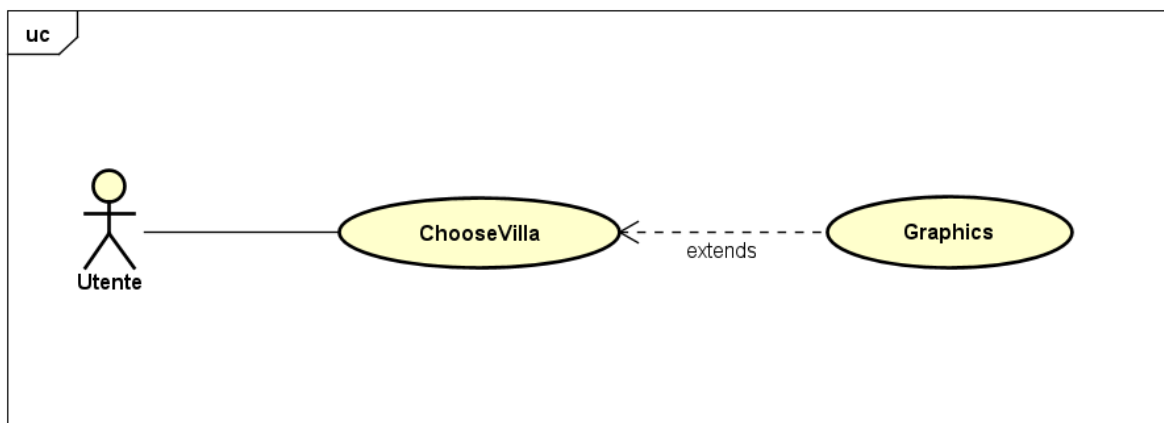


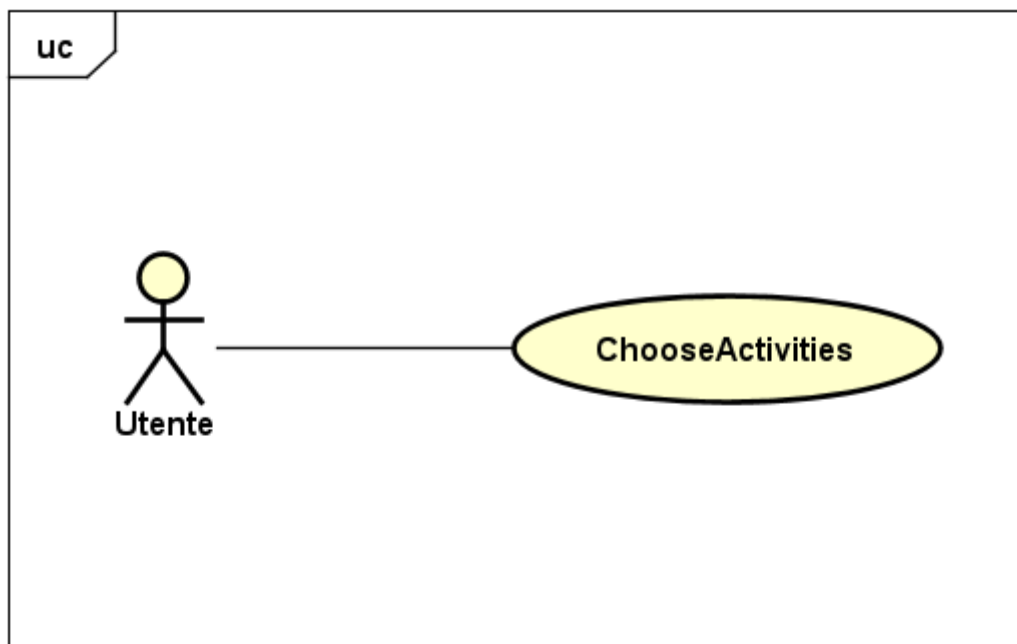
Figura 2: Diagramma del caso d'uso Check-in Check-out.

Nome del Caso d'uso	Choose Villa
Attori partecipanti	Utente
Condizione di entrata	L'utente ha cliccato sul bottone "Next" dal pannello precedente per passare al pannello di scelta della tipologia di villa
Flusso di eventi	<p>1) L'utente sceglie la tipologia di Villa interessata.</p> <p>2) Nella griglia verranno mostrate le ville libere (in blu) e quelle occupate (in rosso)</p> <p>3) L'utente può scegliere di cliccare sul bottone "Follow" per seguire la tipologia di villa inserendo la propria e-mail</p> <p>4) Una volta che l'utente ha effettuato le proprie scelte può passare alla schermata successiva tramite il bottone "Next"</p>
Condizione di Uscita	L'utente può cliccare su "Next" e passare alla schermata successiva per procedere alla prenotazione, oppure, può chiudere il programma e attendere le mail in caso di avvenuta prenotazione di una villa della tipologia che ha deciso di seguire



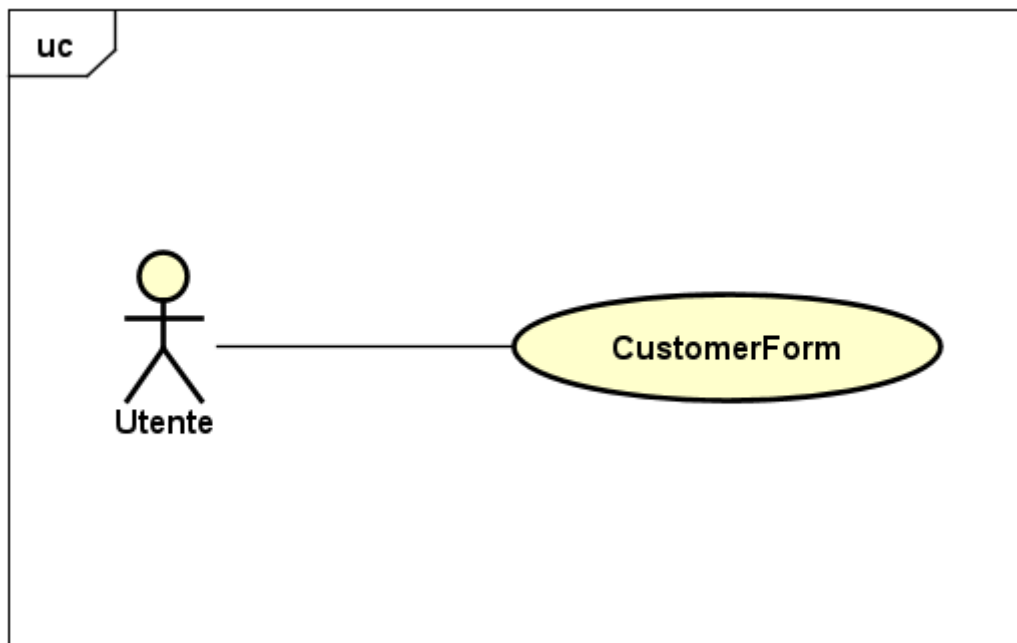
*Figura 3: Diagramma del caso d'uso Choose Villa.*

Nome del Caso d'uso	Choose Activities
Attori partecipanti	Utente
Condizione di entrata	L'utente ha cliccato sul bottone "Next" dal pannello precedente per passare al pannello di scelta delle attività
Flusso di eventi	<p>1) L'utente può scegliere le attività utilizzando gli appositi checkbox</p> <p>2) Una volta che l'utente ha effettuato le proprie scelte può passare alla schermata successiva tramite il bottone "Next"</p>
Condizione di Uscita	L'utente ha cliccato su "Next" e passa alla schermata successiva



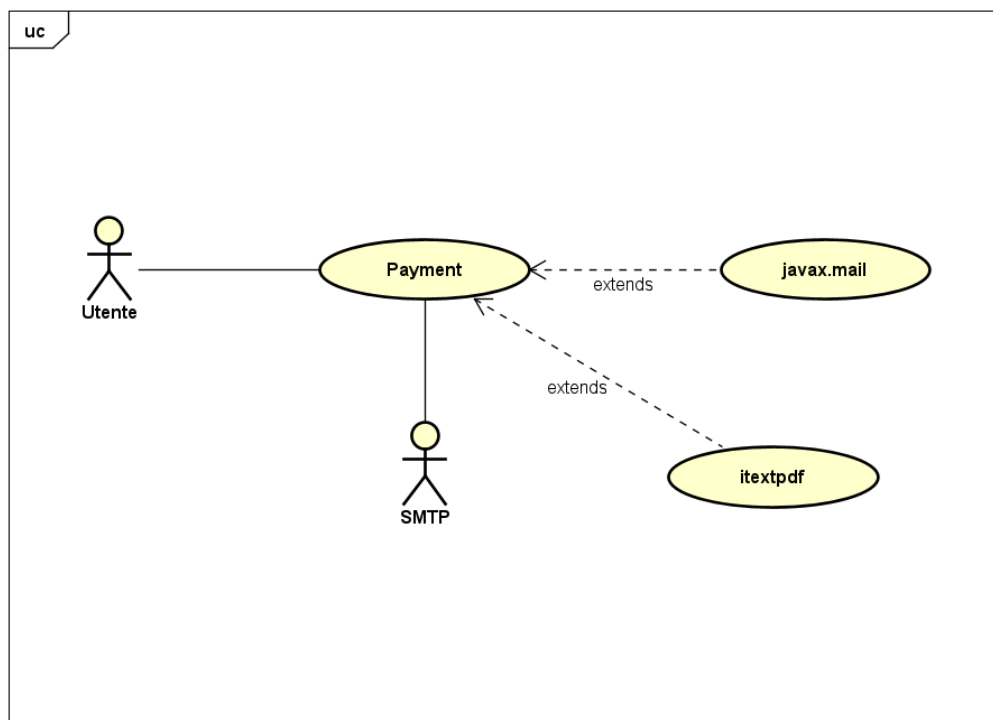
*Figura 4: Diagramma del caso d'uso Choose Activities.*

Nome del Caso d'uso	Customer Form
Attori partecipanti	Utente
Condizione di entrata	L'utente ha cliccato sul bottone "Next" dal pannello precedente per passare al pannello di compilazione dei dati del customer
Flusso di eventi	<p>1) L'utente compila il form con i propri dati anagrafici</p> <p>2) L'utente accetta i termini e condizioni e clicca su "Submit" per immettere i propri dati nel sistema</p> <p>3) L'utente può passare alla schermata successiva cliccando sul bottone "Book"</p>
Condizione di Uscita	L'utente ha cliccato su "Book" e passa alla schermata successiva



*Figura 5: Diagramma del caso d'uso Customer Form.*

Nome del Caso d'uso	Payment
Attori partecipanti	Utente, server SMTP
Condizione di entrata	L'utente ha cliccato sul bottone "Book" dal pannello precedente per passare al pannello di compilazione dei dati di pagamento
Flusso di eventi	<ol style="list-style-type: none"> <li>1) L'utente sceglie la tipologia di pagamento</li> <li>2) L'utente compila i dati richiesti dalla tipologia di pagamento che ha scelto</li> <li>3) L'utente clicca su "Confirm" e successivamente su "Yes" per confermare il pagamento</li> <li>4) Se il pagamento è andato a buon fine, il server SMTP invierà la mail all'utente contenente la fattura in allegato</li> </ol>
Condizione di Uscita	L'utente chiude il programma



*Figura 6: Diagramma del caso d'uso Payment.*

## 5. UML – Diagramma delle classi

La Figura 7 mostra il diagramma delle classi del software Book Your Resort. Tale diagramma, allo scopo di dare una maggiore chiarezza di utilizzo dei pattern, è stato privato di tutte quelle classi che implementano le GUI.

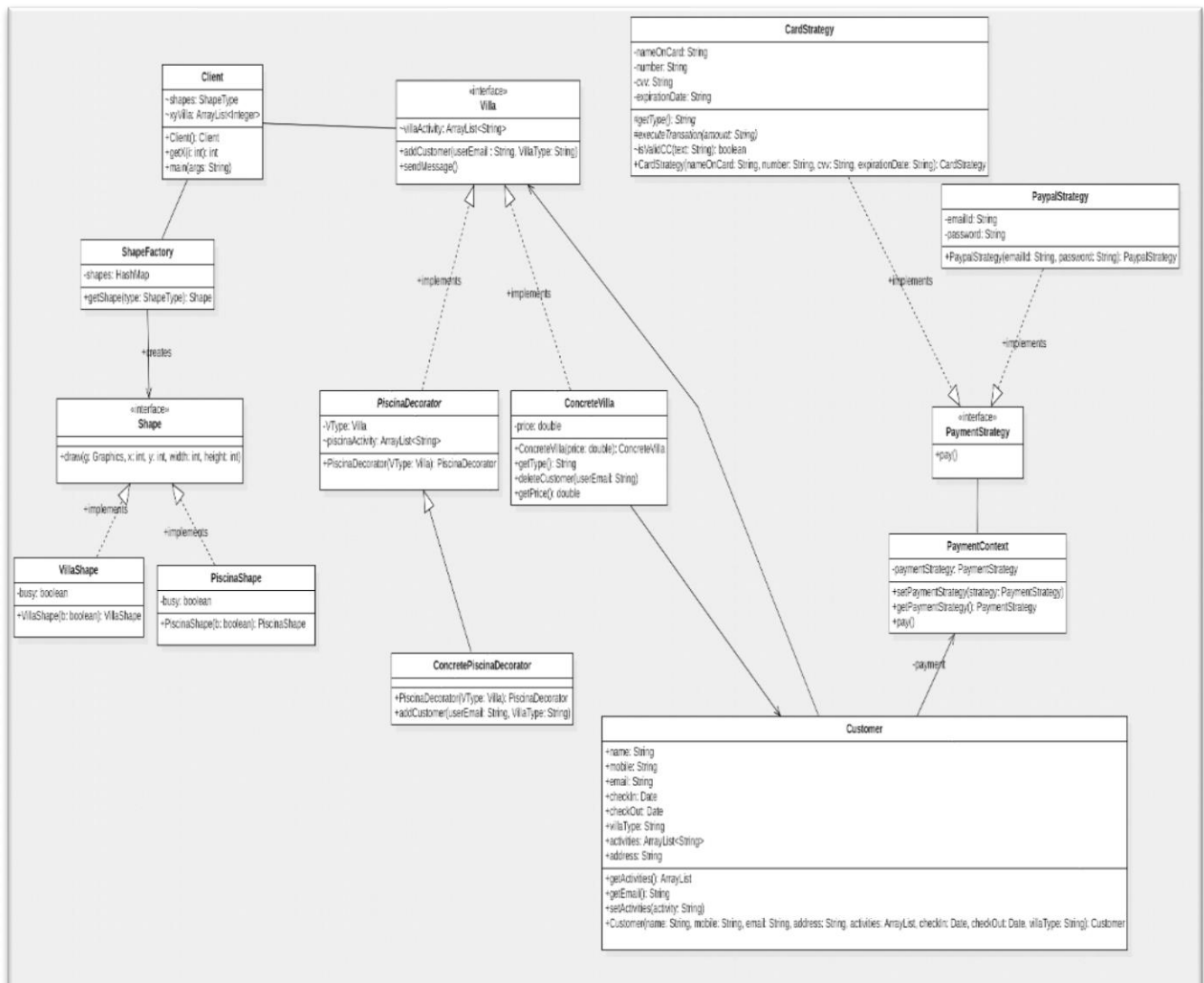


Figura 7: Diagramma delle classi complessivo del software Book Your Resort.

## 6. Design Pattern

L'utilizzo dei Design Pattern si è rivelato utile durante la fase di progettazione in quanto hanno portato alla suddivisione in moduli della soluzione da noi adottata rispettando i principi SOLID.

Tali Design Pattern verranno descritti in maniera dettagliata nei paragrafi successivi.

### 6.1. Flyweight Pattern

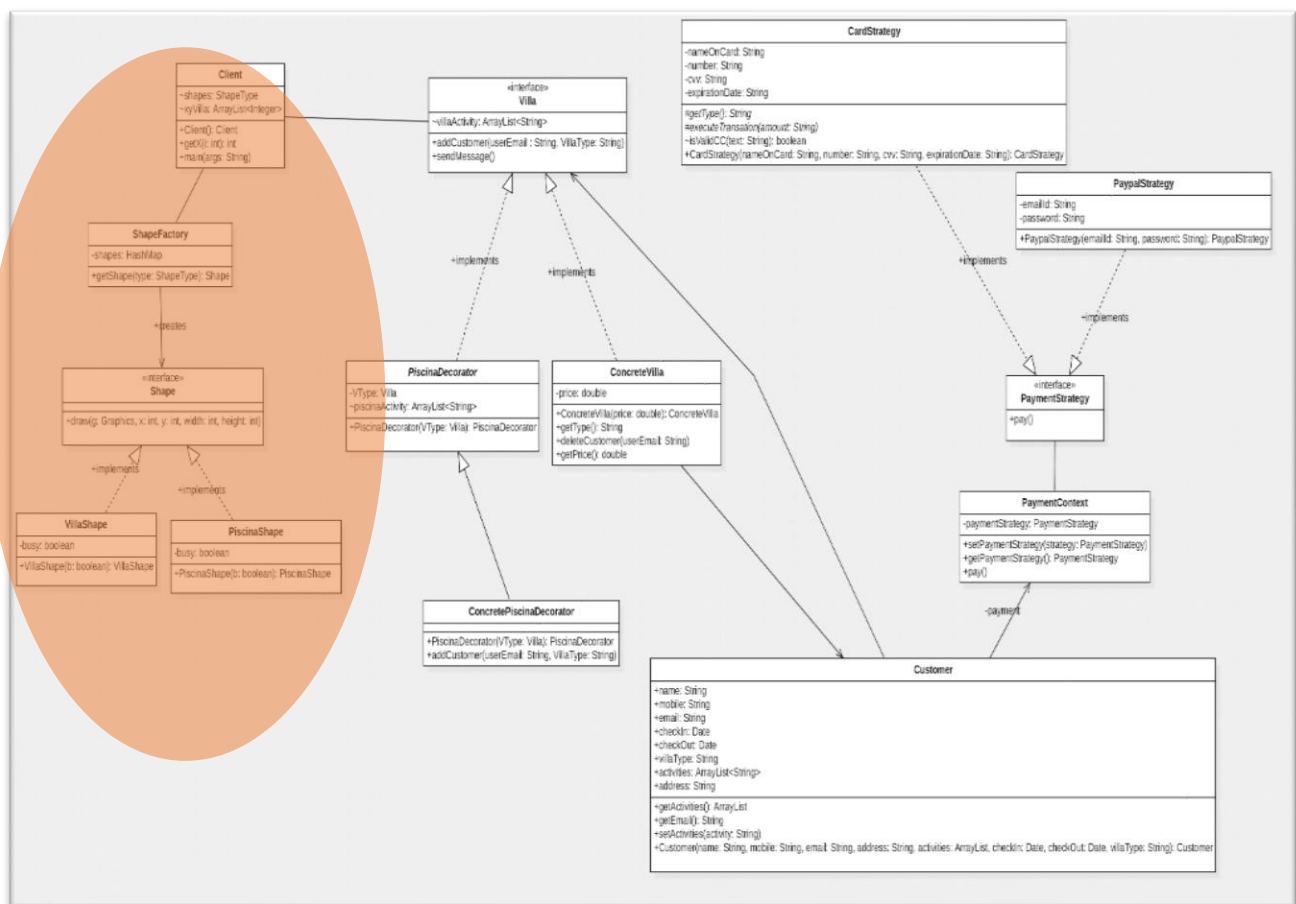


Figura 8: Diagramma delle classi complessivo del software Book Your Resort.

Il Flyweight Pattern si tratta di un pattern strutturale basato su oggetti che viene utilizzato per ottimizzare l'utilizzo delle risorse ed evitare la presenza di oggetti duplicati.

Tale pattern è composto dai seguenti partecipanti:

- **Flyweight:** dichiara una interfaccia attraverso la quale le classi concrete possono definire lo stato estrinseco;
- **ConcreteFlyweight:** implementano l'interfaccia Flyweight e definiscono gli eventuali stati intrinseci;
- **FlyweightFactory:** crea e gestisce oggetti Flyweight. Rende disponibile al Client gli oggetti precedentemente creati oppure se non esiste ne crea uno nuovo e lo mantiene in cache.
- **Client:** mantiene una referencia agli oggetti Flyweight e definisce lo stato estrinseco degli oggetti da creare.

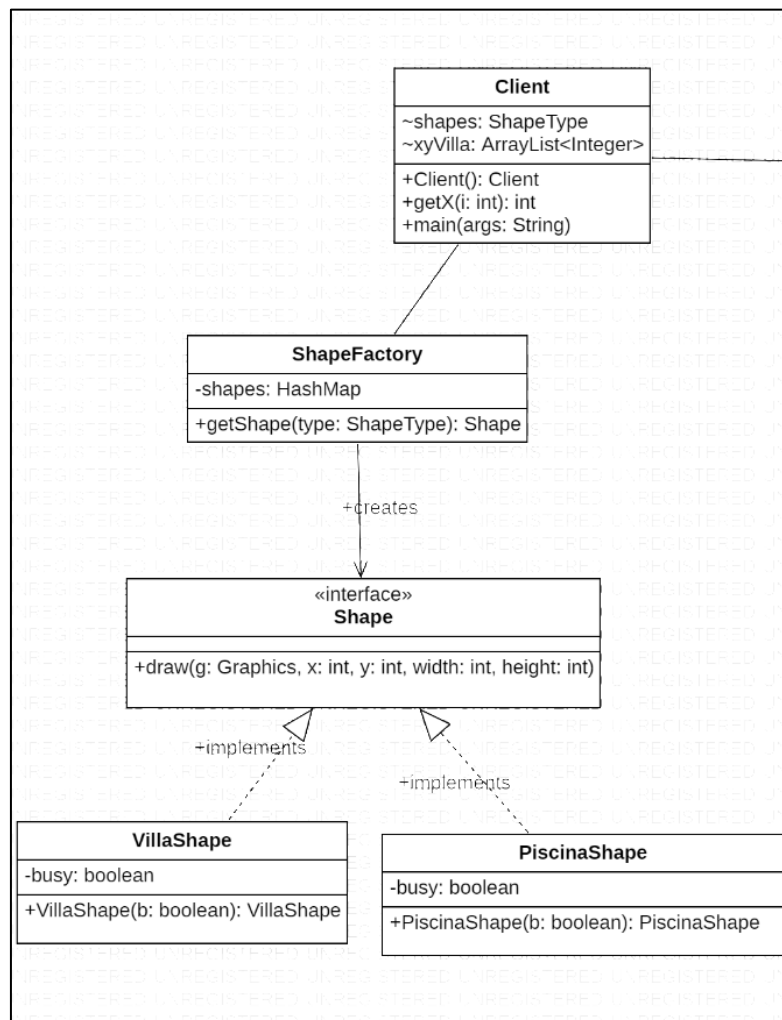


Figura 9: Implementazione del Flyweight Pattern.



Il Flyweight Pattern è il primo Design Pattern utilizzato all'avvio del programma. Tale Pattern è stato implementato insieme all'utilizzo della libreria *Graphics* per la visualizzazione, sulla griglia, delle ville libere e occupate per ogni tipologia di villa.

Nella nostra classe *Client* vengono definite le forme geometriche da disegnare sulla griglia (Rettangoli e Ovali) e il loro rispettivo posizionamento.

La classe *ShapeFactory* utilizza l'HashMap per gestire le forme geometriche. Se quest'ultima non esiste, viene creata e inserita nell'HashMap, in modo da poter essere richiamata quando necessario.

A questo punto non rimane altro che l'interfaccia *Shape* che corrisponde alla generalizzazione della forma geometrica con le rispettive concretizzazioni *VillaShape* e *PiscinaShape*. Tali classi implementano il metodo *draw()* definito nell'interfaccia (figura 10):

```
/**
 * Shape interface. This interface is used to generalize a shape.
 *
 */
public interface Shape {

    /**
     * This method draw a shape based on its implementation using x,y coordinates.
     *
     * @param g a Graphics object of the Graphics class.
     * @param x x coordinate of pool shape.
     * @param y y coordinate of pool shape.
     * @param width width of pool shape.
     * @param height height of pool shape.
     */
    public void draw(Graphics g, int x, int y, int width, int height);
}
```

*Figura 10: Definizione del metodo draw() nell'interfaccia.*

Ogni concretizzazione comanda alla factory di disegnare la rispettiva forma geometrica che le è stata assegnata.

### 6.1.1. Factory Pattern

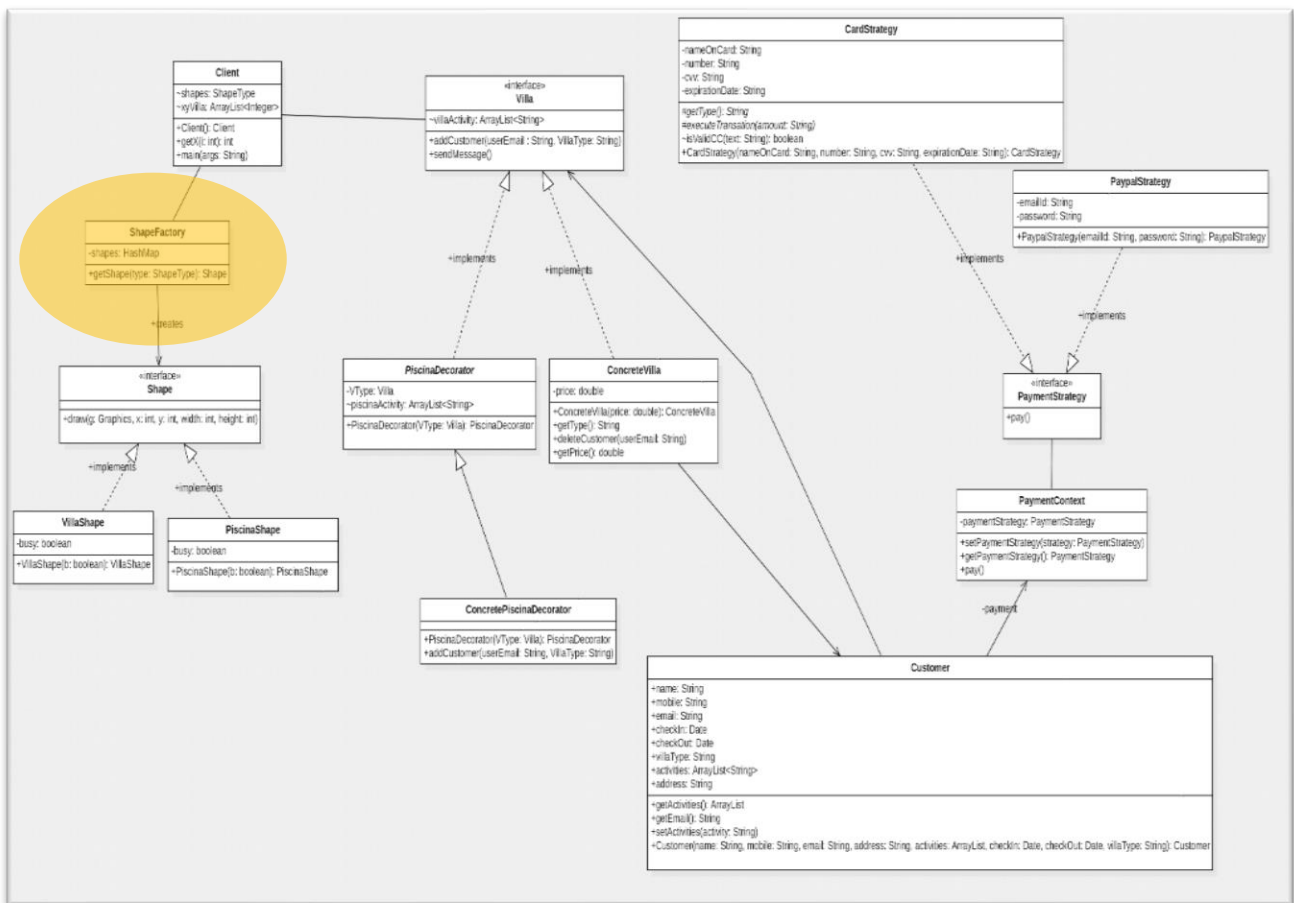


Figura 11: Diagramma delle classi complessivo del software Book Your Resort.

La Factory Pattern si tratta di un pattern creazionale che definisce un'interfaccia per creare oggetti, ma lascia alle sottoclassi la decisione del tipo di classe da istanziare. Tale pattern è composto dai seguenti partecipanti:

- **Creator:** dichiara la Factory che avrà il compito di ritornare l'oggetto appropriato
- **ConcreteCreator:** effettua l'override del metodo della Factory al fine di ritornare l'implementazione dell'oggetto
- **Product:** definisce l'interfaccia dell'oggetto che deve essere creato dalla Factory
- **ConcreteProduct:** implementa l'oggetto in base ai metodi definiti dall'interfaccia Product.

La Factory Pattern nel nostro caso corrisponde alla classe ShapeFactory utilizzata nel Flyweight Pattern, infatti, essa è intrinseca in quest'ultimo Pattern e il suo compito è quello di creare le forme geometriche se non già presenti, aggiungerle nell'HashMap, e ritornarle in maniera tale da poter essere visualizzate sulla griglia.

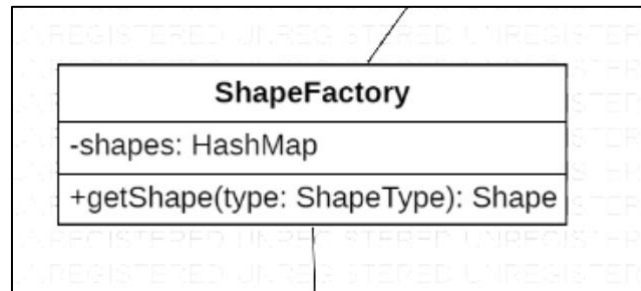


Figura 12: Implementazione della Factory Pattern.

## 6.2. Decorator Pattern

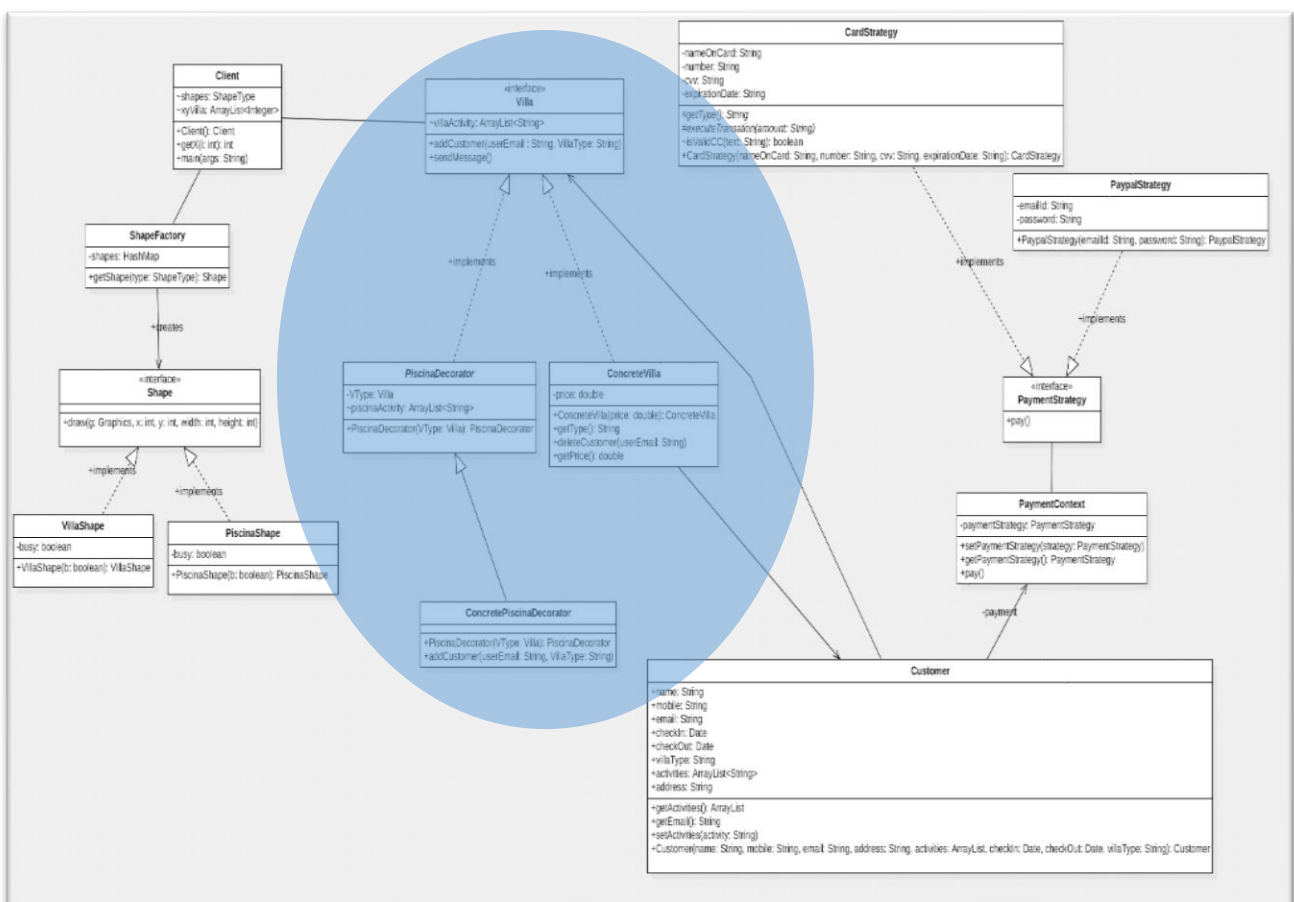


Figura 13: Diagramma delle classi complessivo del software Book Your Resort.

Il *Decorator Pattern* è molto utile e utilizzato dai programmatori più esperti, i quali lo utilizzano al posto della ereditarietà in situazioni in cui è necessario aggiungere/modificare a runtime il comportamento di un oggetto senza scomodare l'ereditarietà. Esso, strutturalmente, prevede quattro elementi principali:

- **Component:** rappresenta l'interfaccia dell'oggetto che dovrà essere decorato dinamicamente;
- **ConcreteComponent:** rappresenta l'oggetto a cui andranno aggiunte le nuove funzionalità;
- **Decorator:** rappresenta l'interfaccia tra il Component e i ConcreteDecorator, possiede un riferimento al Component e un'interfaccia a esso conforme;
- **ConcreteDecorator:** rappresentano gli oggetti che aggiungono le nuove funzionalità ai ConcreteComponent.

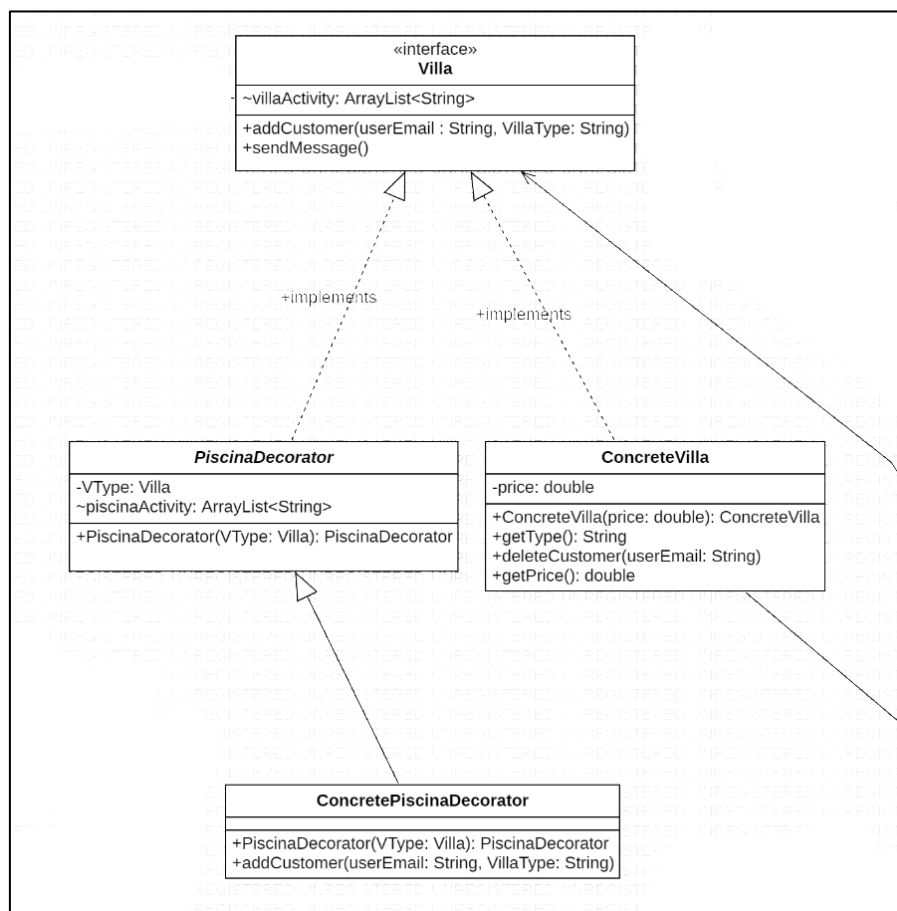


Figura 14: Implementazione del Decorator Pattern.

La Figura 14 mostra la nostra implementazione del Decorator Pattern. L'oggetto che dovrà essere decorato dinamicamente è denominato *ConcreteVilla*, il quale rappresenta la concretizzazione dell'interfaccia *Villa*. Mentre gli oggetti che aggiungono nuove funzionalità ai *ConcreteVilla*, sono rappresentati dalla classe *ConcretePiscinaDecorator*. Quest'ultima, quindi, riguarda l'aggiunzione della piscina ad un oggetto di tipo villa nel momento in cui l'utente avrà selezionato tale preferenza.

### 6.2.1. Observer Pattern

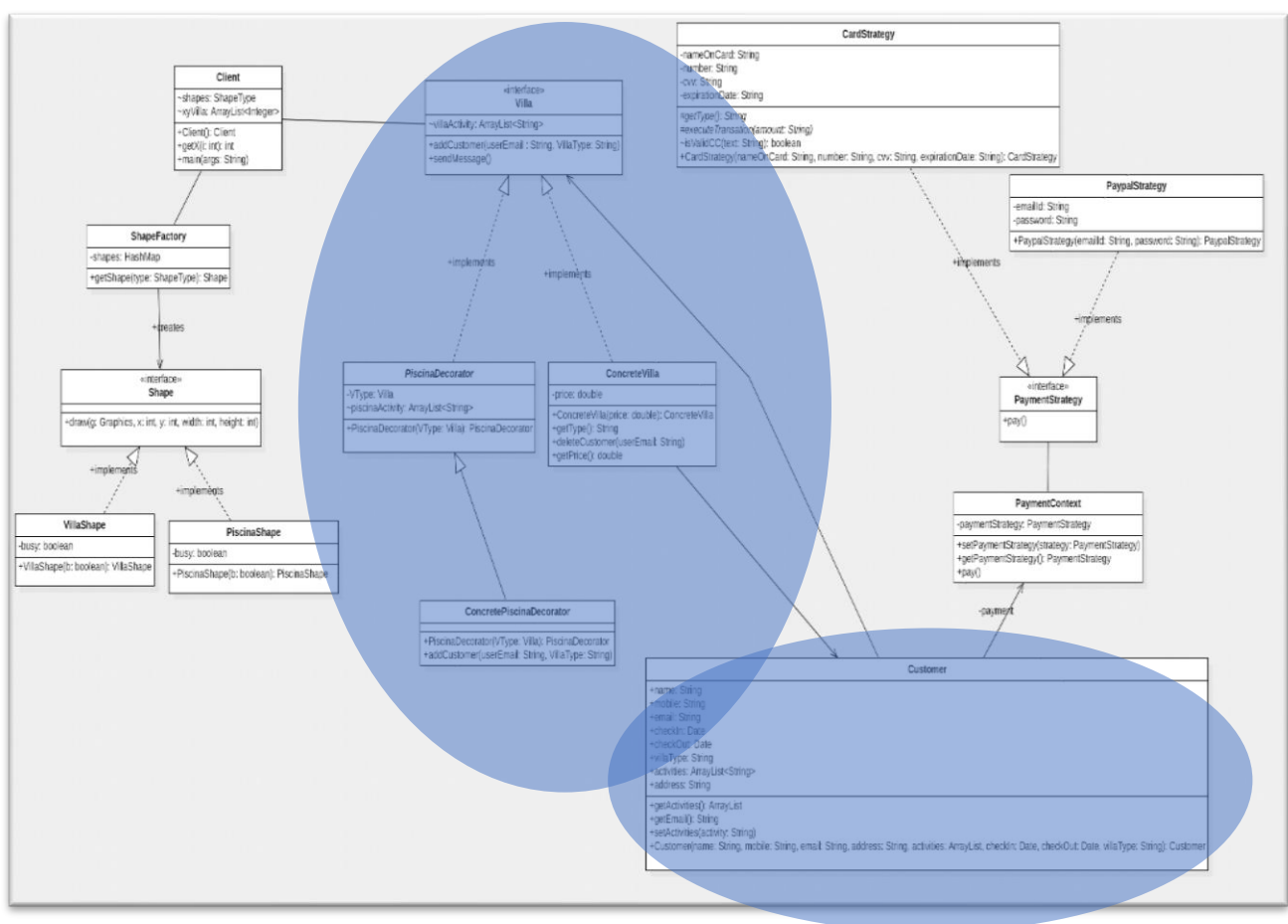


Figura 15: Diagramma delle classi complessivo del software Book Your Resort.

L' *Observer Pattern* trova applicazione nei casi in cui diversi oggetti, denominati come *Observers*, devono conoscere lo stato di un oggetto, il *Subject*, in modo tale che se esso cambia il suo stato interno, ciascuno degli oggetti dipendenti viene notificato e aggiornato automaticamente.

Il Subject dovrà fornire operazioni per l'aggiunta e cancellazione di Observer, ma anche operazioni di notifica.

Come riscontrabile dalla Figura 15 e dalla Figura 16, la nostra implementazione dell'Observer Pattern si fonde con quella del Decorator Pattern. L' oggetto che viene "osservato" è la *Villa* e gli oggetti che "osservano" i cambiamenti di quest'ultimo sono gli oggetti della classe *Customer*, che hanno deciso di seguire tramite il pulsante "Follow" una delle due tipologie di ville messe a disposizione (con o senza piscina).

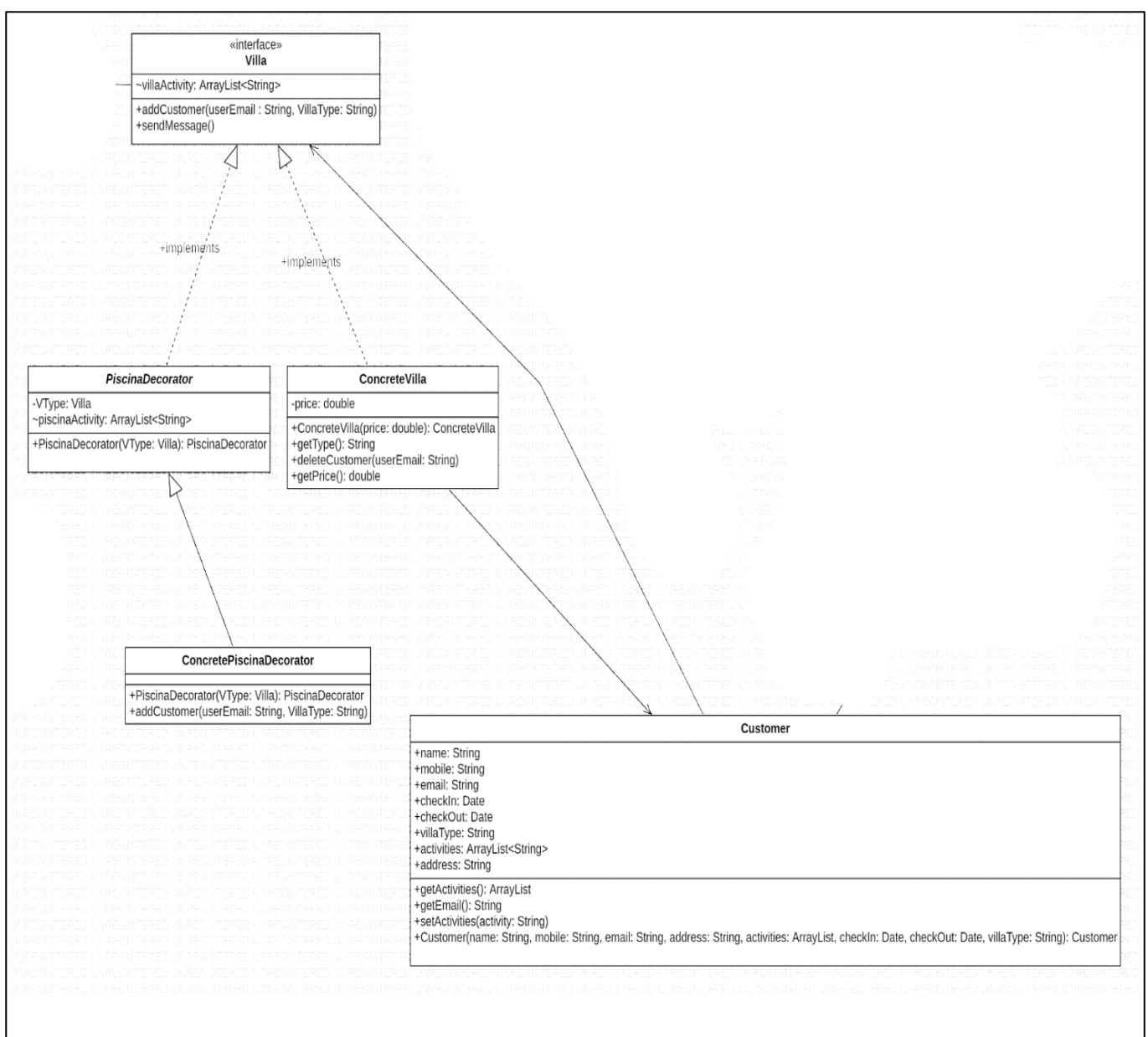


Figura 16: Implementazione del Observer Pattern.

L'interfaccia *Villa* possiede i seguenti metodi di aggiunta di un Customer (follower) e di invio della notifica, che corrisponde all'invio di una e-mail reale.

```
/**
 * This method add the customer.
 * @param userEmail string that represents the customer's email
 * @param VillaType string that represents the type of villa chosen by the customer
 */
public void addCustomer(String userEmail, String VillaType);

/**
 * This method sends email to all followers of the villa type.
 *
 */
public void sendMessage();
```

Figura 17: Definizione del metodo *addCustomer()* e *sendMessage()* nell'interfaccia *Villa*.

Tali metodi verranno implementati nella classe *ConcreteVilla*, concretizzazione dell'interfaccia, insieme al metodo di cancellazione di un Customer (follower).

```
/**
 * This method delete the customer.
 * @param userEmail string that represents the customer's email
 */
public void deleteCustomer(String userEmail) {
    CsvUtilities.deleteEmailToCSVFollowers(userEmail);
}
```

Figura 18: Definizione del metodo *deleteCustomer()* nella classe *ConcreteVilla*.

### 6.3. Strategy Pattern

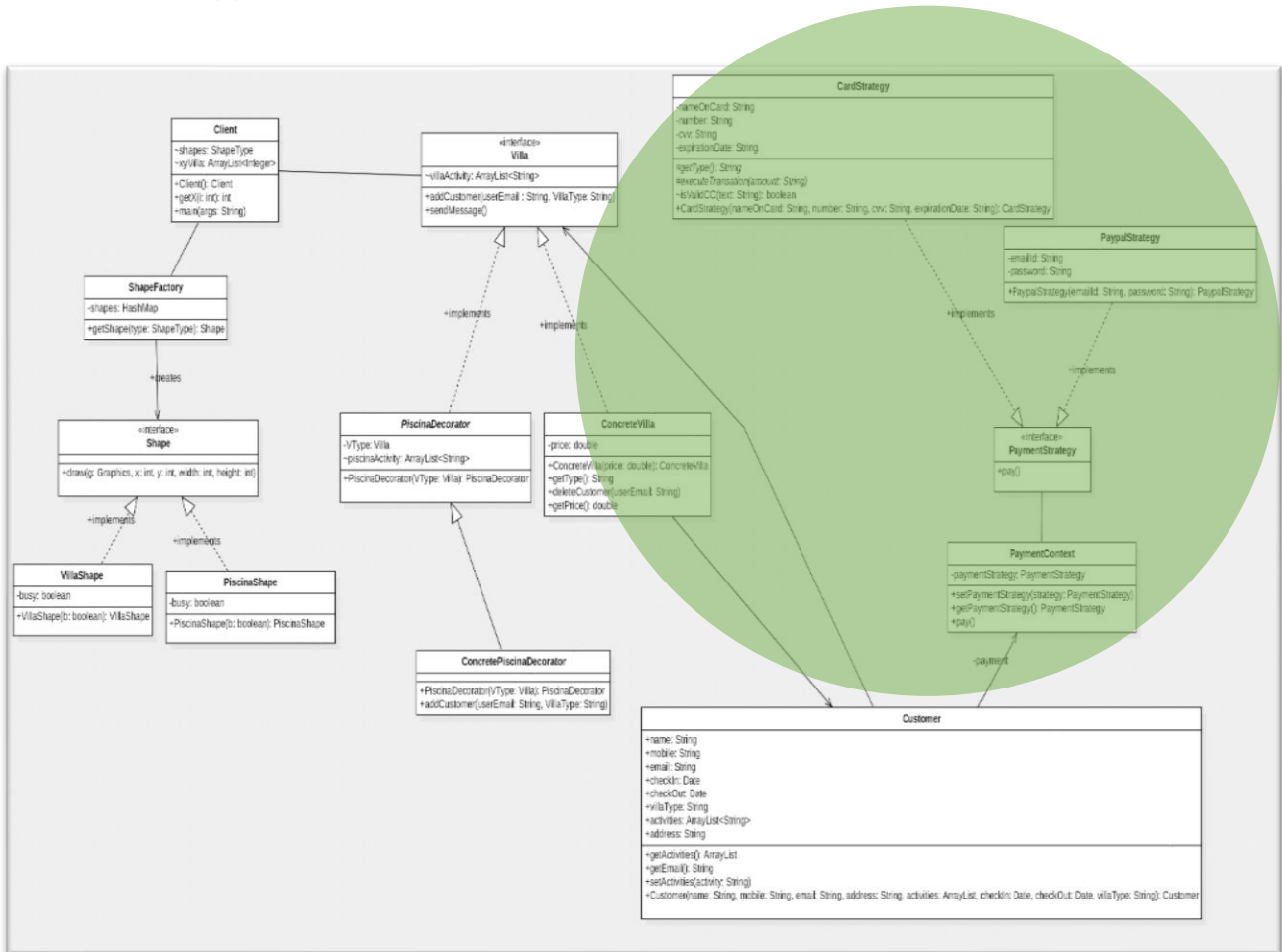


Figura 19: Diagramma delle classi complessivo del software Book Your Resort.

Lo Strategy Pattern è un pattern comportamentale che consente di isolare un algoritmo al di fuori di una classe, per far sì che quest'ultima possa variare dinamicamente il suo comportamento, rendendo così gli algoritmi intercambiabili a runtime. Tale pattern è composto dai seguenti partecipanti:

- **Strategy:** dichiara l'interfaccia della nostra classe di algoritmi. Interfaccia che viene utilizzata da Context per invocare un algoritmo concreto
- **ConcreteStrategy:** sono le concretizzazioni degli algoritmi, ovvero implementano uno specifico algoritmo che espone l'interfaccia Strategy
- **Context:** classe di contesto che invoca la ConcreteStrategy (sotto richiesta dei suoi client).



Utilizziamo lo Strategy Pattern per gestire i metodi di pagamento che l'utente può utilizzare per effettuare il pagamento.

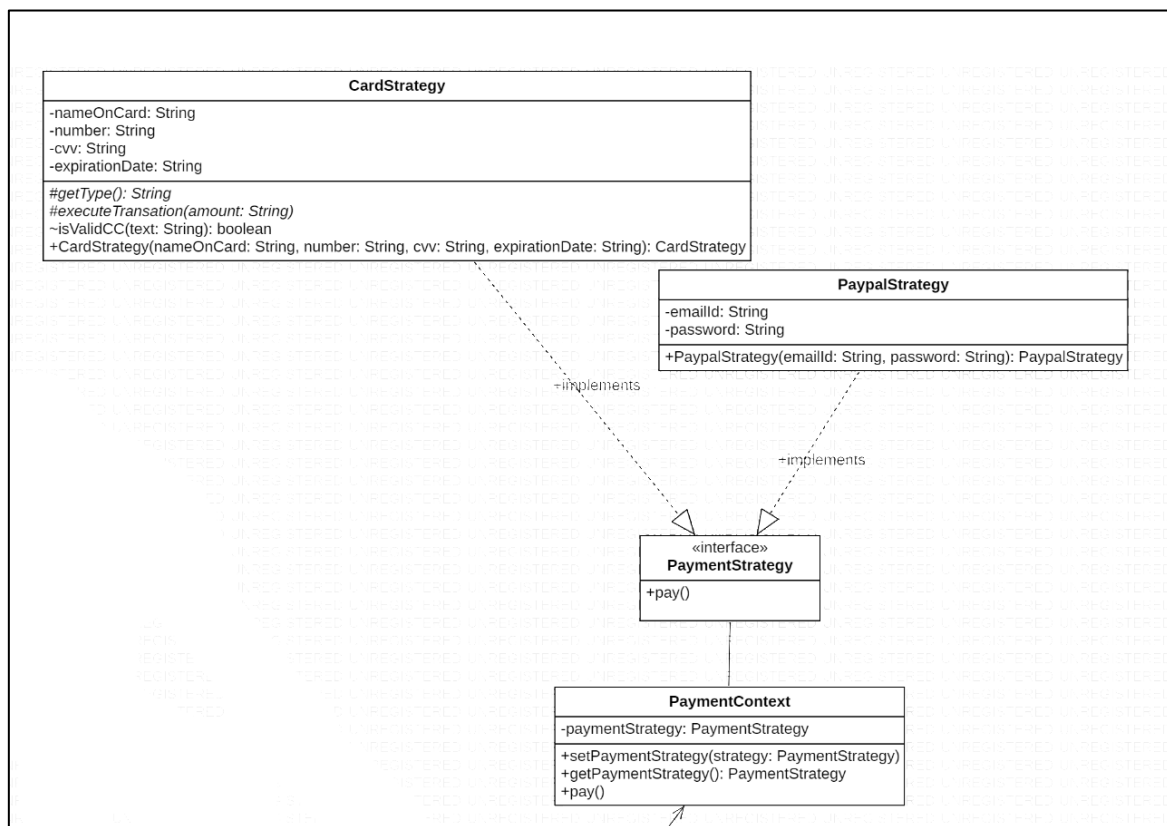


Figura 20: Implementazione dello Strategy Pattern.

La classe *PaymentContext* utilizza l'interfaccia *PaymentStrategy* per gestire il pagamento tramite Paypal o tramite Carta di Credito. Quindi non fa altro che settare la strategia di pagamento scelta dall'utente invocando la concretizzazione dell'interfaccia corrispondente a quella determinata strategia. L'interfaccia *PaymentStrategy* corrisponde alla generalizzazione della strategia di pagamento con le rispettive concretizzazioni *PaypalStrategy* e *CardStrategy*. Tali classi implementano il metodo *pay* definito nell'interfaccia (figura 21):

```

/**
 * Declaration of the pay method.
 *
 */
public void pay();

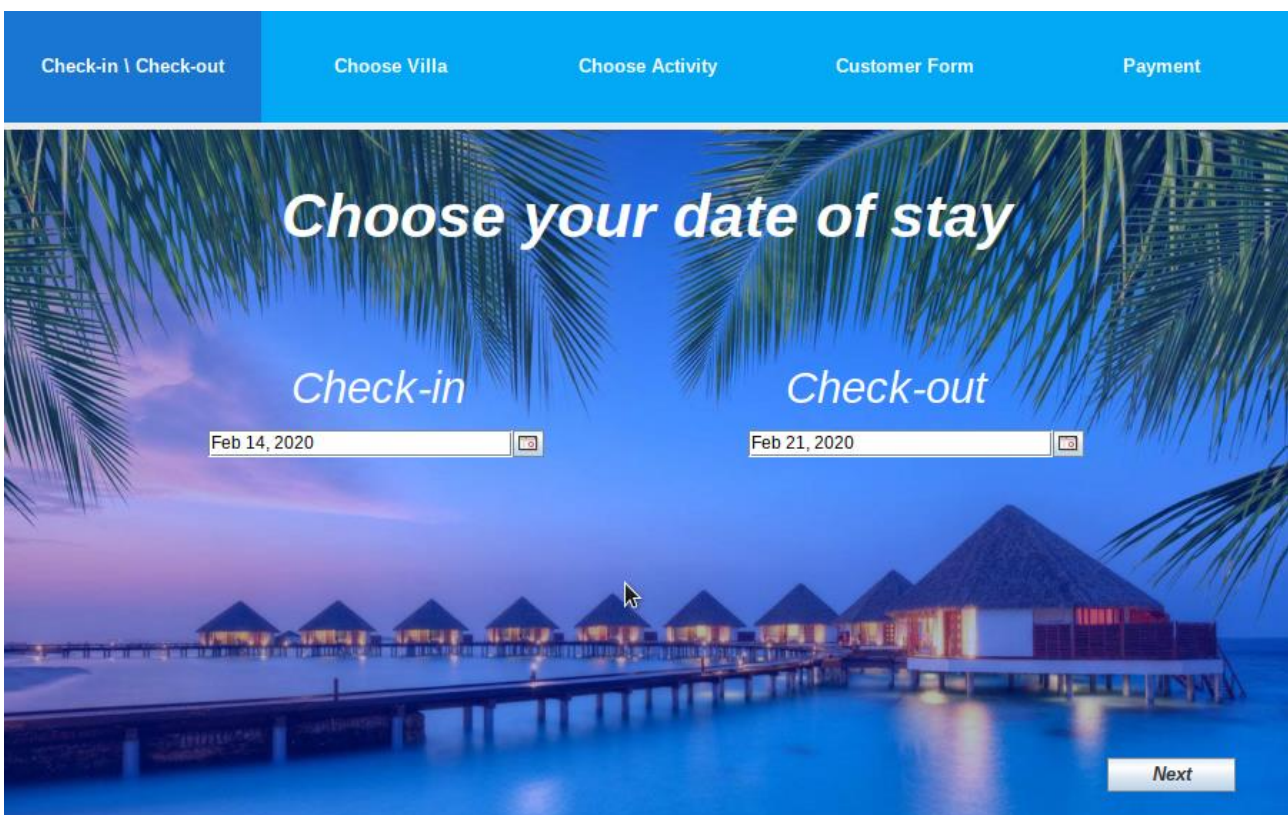
```

Figura 21: Definizione del metodo *pay()* nell'interfaccia *PaymentStrategy*.

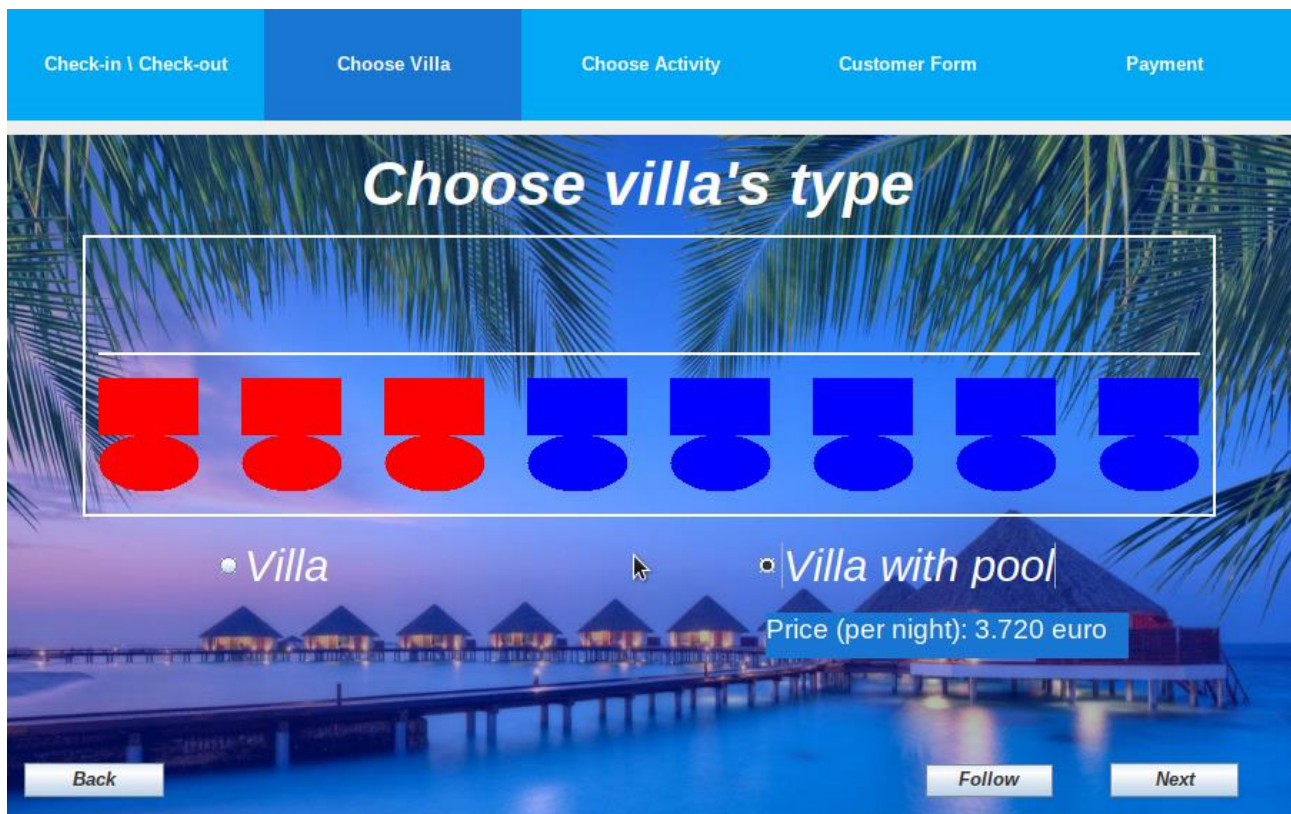
## 7. Guida all'uso e Mock up

Il software utilizza il font Sans-Serif ed è stato inserito in tutti i pannelli in maniera tale da mantenere una coerenza grafica e la portabilità. Il software è stato testato sia su Windows 10 che su Elementary OS (distribuzione basata su Ubuntu).

All'avvio del programma, il primo pannello che viene visualizzato è quello che consente di inserire la data di check-in e di check-out del soggiorno.



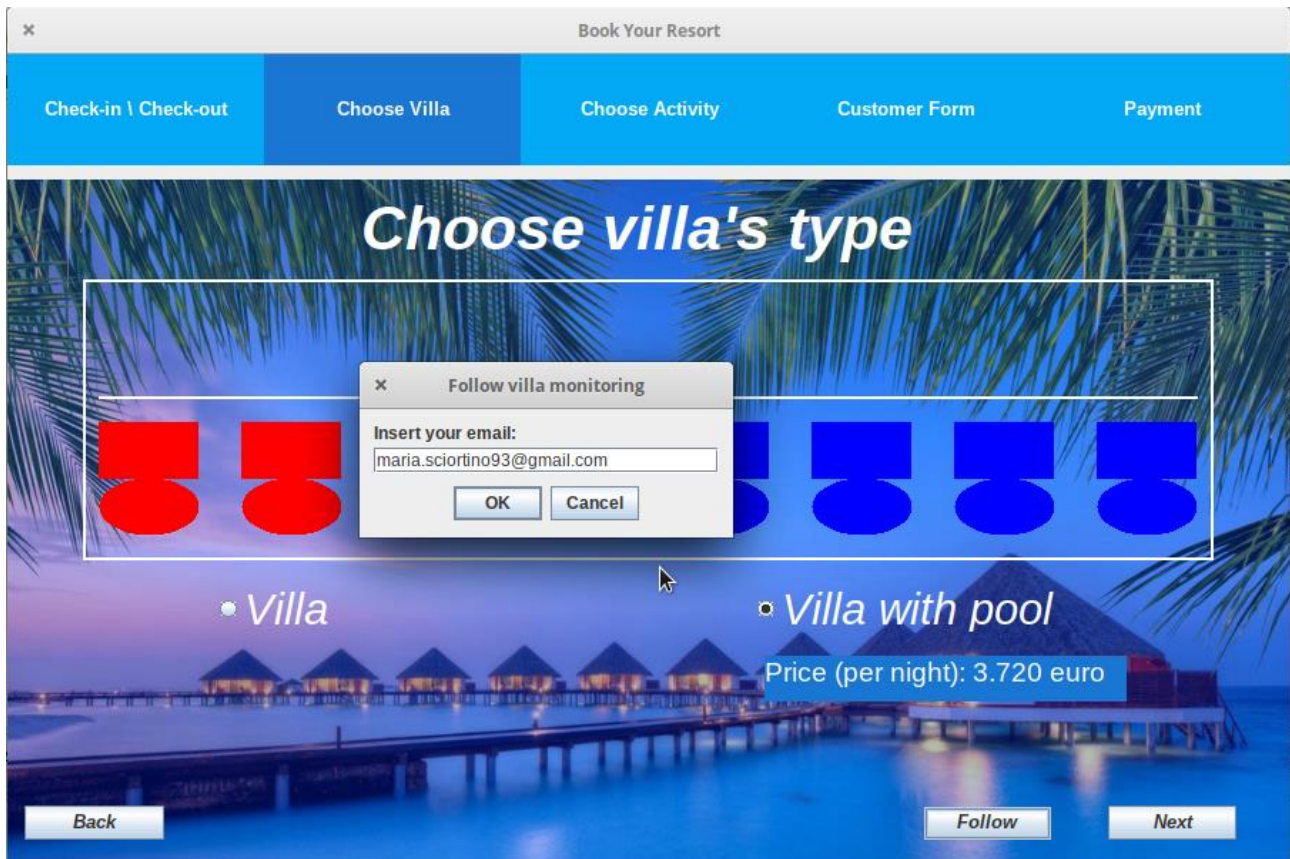
Una volta effettuata la scelta, cliccando sul bottone “Next”, si passa al pannello successivo dove la griglia delle ville non viene visualizzata in quanto bisogna selezionare il tipo di villa per continuare. Successivamente, vengono mostrate sulla griglia le ville della tipologia scelta dove con il colore Rosso si intendono le ville occupate mentre con il colore blue quelle libere (figura a seguire).



Qualora le ville fossero tutte di colore rosso, il sistema non permette di procedere alla prenotazione, in quanto le ville sono state tutte occupate per il range di date selezionato, dunque, bisogna tornare indietro nel pannello precedente per cambiare il range per poi selezionare nuovamente il tipo di villa e procedere con la prenotazione.

In alternativa è possibile cliccare sul bottone "Follow" per seguire la tipologia di villa selezionata (figura a seguire) inserendo la propria e-mail e premendo su "Ok".





Si riceverà una mail nel momento in cui un utente effettua una prenotazione per la tipologia di villa selezionata.



Il pannello successivo consente di scegliere le attività. Verranno mostrate determinate attività in base alla tipologia di villa scelta. Nella figura a seguire, abbiamo deciso di mostrare la visualizzazione derivata dalla selezione della villa con piscina. In particolare, chi sceglie questa tipologia di villa può selezionare anche le attività che possono essere scelte da chi preferisce soggiornare in una villa senza piscina.

Check-in \ Check-out   Choose Villa   **Choose Activity**   Customer Form   Payment

## Choose the activities

☒ Holistic Massage   ☒ Pilates   ☒ Zumba   ☒ Golf   ☐ Excursion   ☐ Yoga

☐ Woga   ☐ AcquaGym   ☒ Hydrobike   ☒ Aqua Tabata   ☐ Swimming Pool Games

Back   Next

Nota: le attività in basso possono essere scelte solamente da chi ha scelto la villa con piscina.

Il pannello successivo consente di inserire i propri dati anagrafici. Una volta inseriti, bisognerà accettare i termini e condizioni e cliccare sul bottone “Submit” per immettere i dati nel sistema. È possibile utilizzare il bottone “Reset” per immettere nuovamente i dati o modificarli direttamente, ricordandosi infine di cliccare su “Submit”.

**Customer Form**

**Name**

**Email**

**Mobile**

**Gender** ☒ Male ☐ Female

**DOB**

**Address**

☒ Accept Terms And Conditions.

Registration Successfully..

**Summary**  
(Please verify your information)

Name : Andrea Di Benedetto  
Email : andreadibenedetto92@gmail.com  
Mobile : 3209883991  
Gender : Male  
DOB : 6/Jun/1993  
Address : via roma, 24



L'ultimo pannello è quello che consente la scelta del metodo di pagamento. Bisognerà selezionare il metodo di pagamento preferito e immettere tutti i dati richiesti dal form per procedere alla prenotazione. Nel caso in cui si scelga di utilizzare la carta di credito, è possibile utilizzare il bottone "Verify" per verificare se il numero della carta immessa è valido o meno, per prevenire eventuali blocchi sul pagamento.

**Choose Payment Method**

☒ Paypal

☐ Credit Card

Email

Card Number

5346746910427534

Verify

Valid number!

Expiration Date

May 2021

CVV

755

Name On Card

Andrea Di Benedetto

Back Confirm

Una volta cliccato su "Confirm" avremo l'ultima chance per apportare eventuali modifiche prima della prenotazione effettiva (figura a seguire).

Cliccando su “Yes” parte il pagamento con il metodo che abbiamo selezionato e viene restituito un popup che informa se è andato a buon fine.

In caso di pagamento effettuato con successo verrà inviata una mail all’email che abbiamo specificato nei nostri dati anagrafici con allegata la fattura contenente tutte le informazioni che abbiamo immesso e il totale pagato (figure seguenti). Attendere qualche secondo il popup di conferma dell’invio della fattura.

NOTA: Effettuando vari test con le nostre e-mail abbiamo riscontrato che si possono avere ritardi nell’invio della mail da parte del server di Libero (max 10 min).



Book Your Resort

Check-in | Check-out

Choose Villa

Choose Activity

Customer Form

Payment

# Choose Payment Method

Paypal

Credit Card

Email

Password

Card Number

56910427534

Verify

Valid number!

Expiration Date

May

2021

CVV

755

Name On Card

Andrea Di Benedetto

Back

Confirm

Message

We sent the invoice to your email!

OK

Booking of Villa

Posta in arrivo

bookyourresort@libero.it

a me

inglese

italiano

Traduci messaggio

Dear Andrea Di Benedetto,

Thank you for your reservation in Book Your Resort.  
Find attached to this mail a copy of your invoice.

Best regards,  
Book Your Resort Staff

Book Your Resort

Invoice No. 14510001

Invoice No. 14510001

Invoice Date 2020/05/24

Name

Andrea Di Benedetto

Email

andrea.benedetto@gmail.com

Mobile

320401091

Address

Marina, 24

PDF

fattura-Andrea Di B...



**Book Your Resort**  
H.Sonary, 2nd Floor, Boduthakurufaanu Magu  
Malé, Maldives - 20026

Account No.	ABC0001
Invoice No.	123456
Invoice Date	2020/02/08

**Name**

Andrea Di Benedetto

**Email**

andreadibenedetto92@gmail.com

**Mobile**

3209883991

**Address**

via roma, 24

**Activities**

Pilates - Zumba - Golf - Hydrobike - Aqua Tabata

**Check-in**

2020-02-14

**Check-out**

2020-02-21

**Villa Type**

Villa with pool

**Price per night (in euro)**

3720.0

**Ext Price**

26040.0

## 8. Conclusioni

Il software Book Your Resort ci ha concesso di vedere nella pratica quanto l'utilizzo dei Design Pattern ci possa agevolare nella progettazione e nello sviluppo di un software. Inoltre, l'implementazione del meccanismo di following di una villa, con conseguente notifica tramite e-mail, ci ha permesso di testare un meccanismo di marketing, spesso usato dalle aziende, che gioca sulla psicologia del cliente. Infatti, l'utente venendo a conoscenza che un altro utente ha prenotato una villa di suo interesse e vedendosi diminuire le ville a disposizione, è portato in maniera implicita ad affrettarsi nell'effettuare la propria prenotazione. Tuttavia, in futuro, sarà possibile migliorare tale software implementando:

- un bottone unfollow per chi non fosse più interessato a seguire una tipologia di villa.
- il metodo `executeTransaction()` per concretizzare il pagamento dell'utente
- un database attraverso un RDBMS.

## REFERENZE

Tool e librerie per lo sviluppo con link:

- <https://www.eclipse.org/>
- <http://astah.net/editions/community>
- <http://staruml.io/>
- <https://toedter.com/jcalendar/>
- <https://itextpdf.com/en>
- <https://javaee.github.io/javamail/>
- <https://www.libero.it/>
- <https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>