



# Università degli Studi di Salerno

*Corso di Ingegneria del Software*

---

---





# Università degli Studi di Salerno

*Corso di Ingegneria del Software*

---

## Quick Service

Requirements Analysis Document

The logo consists of the words "Quick Service" in a stylized green font. The letters are slightly overlapping and have a hand-drawn, informal appearance. To the right of the text is a black stick figure walking towards the right. The figure has a blue dot for a head and a blue line for a tail or a backpack strap.

*Data <25/03/2008>*

**Prefazione:**

Questo documento contiene i requisiti del Quick Service. E' destinato ai designers e ai clienti del progetto.

**Target:**

Clienti, Sviluppatori

**MEMBRI DEL QUICK SERVICE:****Team Members:**

*Vicidomini Vincenzo*

*Mercurio Antonio*

*Iacoletti Alessandro*

*Pacifico Marta*

*Pizza Luca Ernesto*

# *Indice degli argomenti*

---

## **1. Introduzione**

- 1.1 Gli obiettivi del sistema
- 1.2 Scopo del sistema
- 1.3 Obiettivi e criteri di successo del progetto
- 1.4 Riferimenti

1.5 Visione generale

## **2 Sistema corrente**

## **3 Sistema proposto**

3.1 Visione generale

3.2 Requisiti Funzionali

Servizio Ristorante

Servizio Bar

Servizio Attività extra

Servizio Camere

3.3 Requisiti non funzionali

3.3.1 Usabilità

3.3.2 Affidabilità

3.3.3 Performance

3.3.4 Supportabilità

3.3.5 Implementazione

3.3.6 Interfaccia

3.3.7 Packaging

3.3.8 Legali

3.4 Vincoli

### 3.5 Modelli di sistema

#### 3.5.1 Scenari

Servizio Ristorante

Servizio Camere

Servizio Bar

Servizio Attività extra

#### 3.5.2 Modello dei casi d'uso

##### 3.5.2.1 Modello Use Case Attività extra

Attori

Organizzazione del modello

Gestione prenotazioni

Nuova prenotazione

Modifica prenotazione

Annullamento prenotazione

Visualizza prenotazione

Resoconto prenotazione

Gestione Attività

Visualizza attività

Valida

Inserimento Attività

Modifica Attività

Cancellazione Attività

##### 3.5.2.2 Modello dei Casi d'uso Servizio Bar

Attori

Organizzazione del modello

Gestione Ordinazioni

Visualizza Ordinazione

Nuova Ordinazione

Modifica Ordinazione

Disdetta Ordinazione

Visualizza Saldo

Gestione Bar

Visualizza Listino

Inserimento prodotto

Modifica Prodotto

Elimina Prodotto

Valida

### 3.5.2.3 Modello dei Casi d'uso Servizio Ristorante

Attori

Organizzazione del modello

Gestione Prenotazioni

Prenotazioni Menù

Modifica Prenotazioni

Annulamento Prenotazioni

Visualizza Prenotazioni

Visualizza Saldo

Gestione Menù

Visualizza Menù

Valida Utente

Inserimento menu

Modifica menu

Consultazione prenotazioni

#### 3.5.2.4 Modello dei Casi d'uso Servizio Camere

Attori

Organizzazione del modello

Gestione Camere

Visualizza Camera

Valida Camera

Assegna Camera

Modifica Camera

Libera Camera

#### 3.5.3 Modello ad oggetti

##### 3.5.3.1 Livello di presentazione

Addetto Bar

Addetto Ristorante

Gestore

Cliente

##### 3.5.3.2 Livello di applicazione

Package Gestore

Sessione AttivitàGestoreImp

SessioneImp

Package Addetto Bar

Sessione Bar

Sessione Bar Gestore

Package Addetto Ristorante

Main Ristorante

Sessione Ristorante

Sessione RistoranteImp

Package Cliente

Sessione Cliente

Sessione

Sessione ClienteImp

### 3.5.3.3 Livello dati

Cliente

Camera

Prenotazione

Menù

PortateMenu

DettaglioPrenotazioneAttivitàExtra

DettaglioPrenotazioneRistorante

DettaglioPrenotazioneBar

ProdottoBar

Categoria

Utente

AttivitàExtra

### 3.5.4 Modello dinamico

Diagramma di attività

Bar: Nuova Ordinazione

Ristorante: Inserimento Menu

Gestione Camera: Assegna Camera

Attività extra: Cancellazione Attività

Bar: Modifica Ordinazione

Bar: Inserimento Prodotto

Gestione Camere: Libera Camera

Ristorante:Prenotazione Menu

Attività Extra:Nuova Prenotazione

Sequenza di diagramma

Gestione Camera: Assegna Camera

Ristorante: Prenotazione Menu

Bar: Nuova Ordinazione

Attività Extra:Nuova Prenotazione

Ristorante: Inserimento Menu

Bar: Inserimento Prodotto

Attività Extra: Inserimento Attività

### 3.5.5 Interfaccia Utente – Navigational Paths And Screen Mockups

Servizio Attività Extra – Interfaccia Gestore

Servizio Attività Extra – Interfaccia Cliente

Servizio Bar – Interfaccia Gestore

Servizio Bar – Interfaccia Cliente

Servizio Ristorante – Interfaccia Gestore

Servizio Ristorante – Interfaccia Cliente

Servizio Camere – Interfaccia Cliente

## 4. Appendice A : Indice delle Figure

## 5. Glossario

# **1.0 INTRODUZIONE**

## **1.1 GLI OBIETTIVI DEL SISTEMA**

Gli obiettivi principali del sistema sono quelli di realizzare un software per la gestione del servizio in camera presso una struttura. Per poter realizzare ciò, abbiamo quindi bisogno di alcuni requisiti che ci aiutano a sviluppare al meglio l'intero sistema.

- Il software deve essere molto flessibile;
- Si deve poter integrare in applicazioni più vaste;
- Deve essere compatibile con qualsiasi piattaforma.

Il progetto QUICK TOUCH SERVICE si adatta per le strutture che vogliono consentire una gestione facile, comoda e completa dei servizi volti alla clientela. Con un semplice click del mouse, o meglio ancora con un semplice pressione del monitor (touch screen), e con la rappresentazione grafica dei prodotti l'utente potrà:

1. Effettuare le ordinazioni al bar;
2. Decidere il menu per il pranzo e/o per la cena;
3. Partecipare alle attività dell'albergo (es. escursioni);
4. Visualizzare il proprio saldo presso la struttura.
5. E così via ...

Tutte le fasi saranno gestite da un computer che svolgerà tutto il lavoro, dallo smistamento delle ordinazioni fino al calcolo del conto finale da saldare da parte del cliente.

Punti fermi del software da implementare saranno:

- Interfaccia di facile comprensione da parte dell'utenza. Bisogna considerare la completa inesperienza di alcune persone verso l'utilizzo di strumenti tecnologici all'avanguardia.
- Possibilità di adattarsi ai singoli problemi di una struttura. Il software che realizzeremo è implementato per un albergo, ma deve essere facilmente adattabile ad altre situazioni.
- Possibilità di modificare una o più funzioni in maniera semplice. Bisogna realizzare funzioni chiare e ben documentate per poter in futuro modificare, dove serve, i piccoli inconvenienti che vengono fuori dalla fase di Testing.

Il software non solo permetterà un aumento della comodità e della semplicità da parte del cliente, ma anche da parte dell'albergatore. Infatti QUICK TOUCH SERVICE, mette a disposizione del

gestore della struttura tutti quei servizi atti ad una facile, comoda e veloce gestione del proprio lavoro. Tali servizi riguarderanno:

- Avere la possibilità gestire i conti in ogni istante.
- Saranno presenti numerose stampe che permetteranno di avere immediatamente: situazioni, liste di adesioni (per le escursioni), estratti conti etc..
- Possibilità di organizzare la cucina in maniera preventiva a seconda della scelta dei menu effettuata dai clienti
- Possibilità di gestire meglio il servizio di rifornimento della dispensa.
- L'aumento dei comfort a disposizione della clientela.

## 1.2 SCOPO DEL SISTEMA

Lo scopo principale del sistema è quello di mirare alla creazione di un software che faciliti la gestione di strutture e infrastrutture strettamente connesse con le attività turistiche. Quindi l'area di interesse comprende:

- Villaggi vacanza;
- Hotel di qualità;
- Pensioni maggiormente accessibili;

Ma tali strutture sono talvolta frequentate da chi per motivi di lavoro è costretto a spostarsi regolarmente (oppure no) in varie località geografiche.

L'intenzione attualmente più prossima è quella di impegnarsi innanzitutto a migliorare la gestione interna delle strutture(Hotel, Alberghi) migliorando soprattutto i modi di interazione con il cliente e agevolare il più possibile un soggiorno personalizzato e quindi qualitativamente migliore.

## 1.3 OBIETTIVI E CRITERI DI SUCCESSO

Il software che intendiamo proporre parte dall'idea che migliorare la qualità dei servizi offerti favorisce un turismo soddisfacente e sempre crescente innescando un meccanismo che ha per tornaconto l'esigenza di un software sempre più ampio e indispensabile.

Il cuore del nostro prodotto affonda le radici su due osservazioni:

- Migliorare la gestione delle prenotazione di camere e appartamenti siti in villaggi;
- Offrire servizi sempre nuovi e più agevoli da sfruttare, per favorire il consenso del pubblico.

Quindi una buona organizzazione interna è necessaria affinché il cliente rimanga soddisfatto della propria scelta. Bisogna quindi curare qualsiasi cosa riguardi il soggiorno del cliente, dal momento in cui arriva nella struttura fino al momento della partenza.

Per cui è necessario un buon software che risponda alle esigenze del cliente e aiuti l'albergatore a rispondere ad elargire servizi mirati in modo pronto ed efficace.

## **1.4 RIFERIMENTI**

- System Design Document (SDD)
- Object software Engineering – Using UML, Patterns and Java

## **1.5 VISIONE GENERALE**

- Sezione 1: contiene un'introduzione e una descrizione dello scopo e dei principali obiettivi del progetto QUICK TOUCH SERVICE
- Sezione 2: Descrive, se esiste, lo stato di un sistema corrente con le sue funzionalità e i problemi attinenti ad esso
- Sezione 3: Descrive il sistema da realizzare, i requisiti funzionali e non funzionali; riporta inoltre i modelli del sistema e un prototipo dell'interfaccia utente che verrà realizzata.
- Sezione 4: Contiene un glossario nel quale vengano riassunti e specificati alcuni termini presenti nel documento al fine di rendere chiaro il significato di ogni sezione

## **2.0 SISTEMA CORRENTE**

Attualmente di software del genere ne esistono di svariati, ed in ognuno di essi si trattano i seguenti problemi:

1. gestione dell'assegnamento delle camere
2. gestione anagrafiche
3. gestione listini con stagionalità, supplementi a camera e a persona
4. disponibilità di periodo
5. disponibilità giornaliera
6. storico soggiorni e situazione sospesi direttamente dall'anagrafiche

ma in nessuno di essi sono trattati i seguenti problemi:

1. gestione del “servizio in camera” con possibilità di ordinare al bar o al ristorante dalla propria camera
2. gestione delle partecipazioni alle attività della struttura

Su questi ultimi due punti si basa l'idea di progetto e quindi si cercheranno di aumentare e di migliorare nello stesso tempo i servizi offerti, dalle strutture quali alberghi e villaggi, ai propri clienti.

## 3.0 SISTEMA PROPOSTO

### 3.1 Visione generale

L'overview del sistema ci mostra un'architettura gerarchica di tipo client-server. In tale tipo di rete, più computer (client) accedono a servizi e risorse distribuite da un computer (server) dedicato a svolgere particolari compiti:

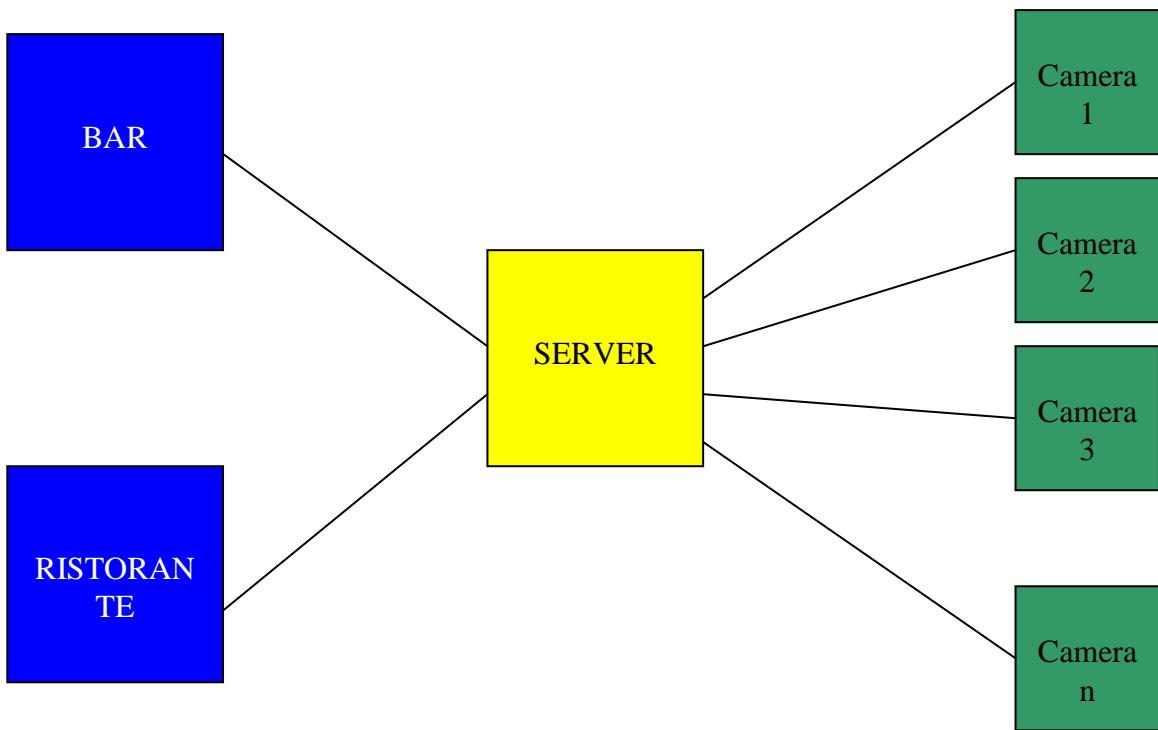
- Amministrazione
- Condivisione file
- Condivisione stampanti
- Condivisione applicativi
- Fornitura di servizi

Il computer addetto alla gestione del sistema, sarà il server e risiederà all'interno di un'area riservata della rete locale (LAN). Quindi il server servirà ad amministrare tutte le attività svolte nell'intero sistema (assegnazioni, aggiornamenti, cancellazioni dei prodotti, aggiornamento della disponibilità, verifica di ordini e pagamenti). Inoltre, il server, grazie ad una stampante presso il bar ed il ristorante, è usato anche per smistare le ordinazioni che arrivano dalle camere dei clienti.

Invece a tutte le altre postazioni (client) sarà possibile effettuare operazioni quali:

1. Ordinazioni al bar.
2. Ordinazioni al ristorante.
3. Prenotazioni escursioni.
4. Visualizzazione del proprio saldo.
5. Altri servizi simili.

La nostra architettura avrà il seguente aspetto:



Quindi saranno necessarie quattro figure di utenti:

- cliente, che effettua le richieste ai servizi trattati;
- gestore, che assegna i clienti alle camera e salda il conto alla fine del soggiorno del cliente;
- addetto bar, che visualizza le ordinazioni, modifica i dati di un prodotto (prezzo, stato, etc);
- addetto ristorante, che visualizza i menu prenotati, inserisce e/o modifica i menu.

## **3.2 REQUISITI FUNZIONALI**

### **SERVIZIO RISTORANTE**

#### **1 – VISUALIZZAZIONE MENU'**

**DESCRIZIONE:** questa funzionalità consente ai clienti dell'albergo di poter visualizzare nelle proprie camere il menù giornaliero disponibile; colazione, pranzo e cena.

**ATTORE:** cliente dell'albergo  
gestore del ristorante

**NOTE:** ovviamente anche il gestore dell'albergo è abilitato alla visualizzazione del menù.

#### **2 – PRENOTAZIONE MENU'**

**DESCRIZIONE:** questa funzionalità consente ai clienti di poter prenotare un tavolo al ristorante per la colazione, pranzo o cena scegliendo direttamente nel menù.

**ATTORE:** cliente dell'albergo.

**NOTE:** questa funzionalità permette la possibilità ai clienti dell'albergo di usufruire del servizio in camera.

#### **3 – MODIFICA PRENOTAZIONE**

**DESCRIZIONE:** questa funzionalità da al cliente la possibilità di effettuare delle modifiche sulla propria ordinazione.

**ATTORE:** cliente dell'albergo.

**NOTE:** la modifica si può effettuare solo se il cliente ha già fatto un ordinazione, e tale modifica va effettuata entro un certo orario.

#### **4 – ANNULLAMENTO PRENOTAZIONE**

**DESCRIZIONE:** questa funzionalità permette al cliente di effettuare l'annullamento della Prenotazione.

**ATTORE:** cliente dell'albergo.

**NOTE:** l'annullamento si può effettuare solo se il cliente ha già fatto un ordinazione, e tale modifica va effettuata entro un certo orario.

## **5 – VISUALIZZA PRENOTAZIONI**

**DESCRIZIONE:** il cliente deve poter visualizzare le informazioni sulle prenotazioni effettuate.

**ATTORE:** cliente.

## **6 – CONSULTAZIONE PRENOTAZIONI**

**DESCRIZIONE:** il gestore del ristorante ha il compito di leggere le prenotazioni.

**ATTORE:** gestore del ristorante.

## **7 – INSERIMENTO MENU'**

**DESCRIZIONE:** il gestore del ristorante definisce i menù, e le schermate che appariranno al cliente in camera.

**ATTORE:** gestore del ristorante

## **8 – MODIFICA MENU'**

**DESCRIZIONE:** il gestore del ristorante effettua delle modifiche al menù visualizzato dai clienti nelle proprie camere.

**ATTORE:** gestore del ristorante.

**NOTE:** il menù dovrà essere definito almeno 30 minuti prima dell'abilitazione alla ricezione ordini.

## **9-SALDO**

**DESCRIZIONE:** è la funzionalità che visualizza al cliente il saldo della stanza, sarà addebitata ogni ordinazione in tempo reale.

**ATTORE:** cliente dell'albergo

**NOTE:** ovviamente il saldo sarà pagato all'abbandono della stanza.

## **10-VALIDA**

**DESCRIZIONE:** è la funzionalità che permette il riconoscimento del gestore e che quindi ne ne permette l'accesso ad alcune funzionalità del sistema.

**ATTORE:** gestore del ristorante

## **SERVIZIO BAR**

### **1-VISUALIZZAZIONE DEI PRODOTTI DEL BAR**

**DESCRIZIONE:** permette al cliente la visualizzazione del listino del Bar con le relative informazioni dei vari prodotti disponibili.

**ATTORE:** cliente dell'albergo

### **2-SCELTA E ORDINAZIONE**

**DESCRIZIONE:** il cliente effettua la scelta di uno o più prodotti, tra quelli disponibili, confermando infine il tutto con l'ordinazione. Nell'ordinazione deve essere selezionato almeno un prodotto.

**ATTORE:** cliente dell'albergo

### **3-MODIFICA ORDINAZIONE**

**DESCRIZIONE:** nel caso che il cliente, per qualche ripensamento, voglia apportare delle modifiche ad una Ordinazione già avvenuta. Tale funzionalità è disponibile solo dopo che è stata effettuata una Ordinazione, e può avvenire in un "Tot Tempo".

Quando è possibile modificare l'ordinazione precedentemente effettuata lo stato di quest'ultima viene impostato in “modifica”, in questo caso l'addetto bar non visualizzerà l'ordinazione che sta subendo delle modifiche.

**ATTORE:** cliente dell'albergo

### **4-DISDETTA ORDINAZIONE**

**DESCRIZIONE:** Nel caso che il cliente, per qualche ripensamento, voglia annullare un ordine precedentemente inviato. Tale funzionalità è disponibile solo dopo che è stata effettuata una ordinazione, e può avvenire in un "Tot Tempo".

**ATTORE:** cliente dell'albergo

### **5-VISUALIZZAZIONE SALDO**

**DESCRIZIONE:** il cliente può visualizzare il suo attuale saldo presso la struttura, in questo modo è comodamente e costantemente informato delle spese effettuate.

**ATTORE:** cliente dell'albergo

### **6-INSERIMENTO DI UN PRODOTTO**

**DESCRIZIONE:** l'addetto alla gestione del bar, può decidere di acquistare nuovi

prodotti per il bar, quindi deve disporre di una funzionalità che gli permetta un inserimento nel listino, che viene visualizzato al cliente.

ATTORE: addetto bar

#### **7-MODIFICA PRODOTTO**

DESCRIZIONE: Permette al gestore del bar di modificare qualsiasi informazione (Prezzo, Disponibilità, ecc.) relativa ad un prodotto del listino. Questa operazione deve poter avvenire solo se il prodotto è già presente nel listino.

ATTORE: addetto bar

#### **8-CANCELLAZIONE PRODOTTO**

DESCRIZIONE: permette al gestore del bar, per una qualsiasi causa, di cancellare un prodotto dal listino. Questa operazione deve poter avvenire solo se il prodotto è già presente nel listino.

ATTORE: addetto bar

#### **9-VALIDA**

DESCRIZIONE: per effettuare operazioni di modifica, inserimento e cancellazione l'addetto al bar deve effettuare il login. Questa rende il sistema più sicuro.

ATTORE: addetto bar

### **SERVIZIO PRENOTAZIONE ESCURSIONI E ATTIVITA' SIMILI**

#### **1-VISUALIZZAZIONE ATTIVITA'**

DESCRIZIONE: il menu di visualizzazione si presenta composto da varie finestre, ognuna delle quali caratterizzata da un titolo che indica il nome dell'attività, completato da una breve descrizione dell'itinerario o del tipo di evento sportivo-ricreativo, dal relativo costo, data e ora di partenza, nonché dal tempo rimasto per una eventuale prenotazione.

Il cliente interessato può cliccare sulla finestra che gli interessa e gli apparirà una form amichevole di prenotazione.

ATTORE: cliente dell'albergo

#### **2-PRENOTAZIONE**

DESCRIZIONE: al momento della prenotazione l'utente deve specificare il numero di posti

prenotati e per ognuno fornire nome, cognome, sesso e scegliere tra eventuali opzioni (adulto/bambino, prenotazione pasto ecc.).

NOTE: la prenotazione può essere effettuata esclusivamente entro il limite di tempo evidenziato nella form.

### **3- MODIFICA PRENOTAZIONE**

DESCRIZIONE: il cliente è in grado di modificare la sua prenotazione, ad esempio cambiando il numero di partecipanti o scegliendo un'altra tra le iniziative disponibili.

ATTORE: cliente dell'albergo

NOTE: la modifica può essere effettuata esclusivamente entro il limite di tempo evidenziato nella form.

La modifica è possibile solo se precedentemente è stata effettuata una prenotazione.

### **4-ANNULLAMENTO PRENOTAZIONE**

DESCRIZIONE: è possibile annullare la prenotazione di uno o più partecipanti ad un'attività.

ATTORE: cliente dell'albergo

NOTE: l'annullamento può essere effettuato esclusivamente entro il limite di tempo evidenziato nella form.

L'annullamento è possibile solo se precedentemente è stata effettuata una prenotazione.

### **5-VALIDA**

DESCRIZIONE: il gestore, o chi per esso, per accedere a funzionalità quali inserimento, cancellazione e modifica, è tenuto a inserire "nome utente" e "password", onde evitare che un non addetto possa accedere a tali funzioni riservate.

ATTORE: gestore dell'albergo

NOTE: la valida non è richiesta per la funzione di visualizzazione.

### **6-VISUALIZZAZIONE**

DESCRIZIONE: il gestore è in grado di visualizzare l'interfaccia grafica per compiere determinate operazioni.

ATTORE: gestore dell'albergo

NOTE: per questa operazione non è richiesta la valida.

### **7-INSERIMENTO ATTIVITA'**

**DESCRIZIONE:** il gestore, quando inserisce un'attività nel menu, deve scegliere un titolo, deve dare una descrizione sintetica ma esaustiva e aggiungere data e ora di inizio attività, tempo rimasto per la prenotazione e, se previsto, aggiungere nella form opzioni speciali come prenotazione pasto, differenziazione costi adulto/bambino ecc, numero di posti disponibili.

**ATTORE:** gestore dell'albergo

**NOTE:** per questa operazione è richiesta la valida.

#### **8-CANCELLAZIONE ATTIVITA'**

**DESCRIZIONE:** il gestore è in grado di cancellare un'attività ma bisogna distinguere tra una cancellazione ordinaria, che avviene quando il tempo rimasto per la prenotazione è scaduto, e una cancellazione dovuta a motivi straordinari.

Nel primo caso la cancellazione elimina semplicemente l'attività dal menù, nel secondo caso la cancellazione deve essere resa nota tramite un messaggio ben visibile all'utente corredata anche da una breve motivazione.

**ATTORE:** gestore

**NOTE:** i dati dei clienti prenotati sono memorizzati in un server in modo da facilitare l'organizzazione dell'evento e addebitare il relativo costo sul conto del cliente. La cancellazione può avvenire solo se l'attività è stata precedentemente inserita. Per questa operazione è richiesta la valida.

#### **9-RESOCONTI PRENOTAZIONI**

**DESCRIZIONE:** il gestore deve poter visualizzare il resoconto delle prenotazioni per una singola attività extra dell'albergo.

**ATTORE:** gestore

### **SERVIZIO GESTIONE DELLE CAMERE**

#### **1-VISUALIZZAZIONE CAMERE**

**DESCRIZIONE:** questa funzionalità permette al gestore di visualizzare tutte le informazioni riguardanti le camere dell'albergo sia di quelle occupate che di quelle libere, ed inoltre visualizza il saldo della stanza in quel momento.

**ATTORE:** gestore delle camere

#### **2-ASSEGNA CAMERA**

**DESCRIZIONE:** il gestore dell'albergo assegna le camere in base ad alcune informazioni come il numero preciso di clienti che vogliono affittare una stanza e specifica anche se il cliente ha affittato una stanza a mezza pensione o a pensione completa.

**ATTORE:** gestore delle camere

### **3-MODIFICA CAMERA**

**DESCRIZIONE:** il gestore modifica le informazioni relative ad una specifica stanza all'interno dell'albergo.

**ATTORE:** gestore delle camere

### **4-LIBERA CAMERA**

**DESCRIZIONE:** con questa funzionalità il gestore libera la camera con la richiesta di conferma al cliente e l'eventuale stampa del conto finale.

**ATTORE:** gestore delle camere

## 3.3 REQUISITI NON FUNZIONALI

### 3.3.1 Usabilità

Per utilizzare “Quick Service” non occorre nessuna particolare abilità informatica in quanto si presta all’utilizzo di qualunque tipo di utente che troverà molto comprensibili le interfacce grafiche del sistema per accedere alle informazioni che desidera. Tale aspetto è molto importante soprattutto se si considera che il software è rivolto ai gestori, ma soprattutto ai clienti delle strutture alberghiere che, come è immaginabile, possono appartenere a qualunque età ed estrazione sociale.

Le interfacce “user friendly” eviteranno gli errori e le incomprensioni e, qualora vi fosse qualche errore in fase di imputazione, non causerebbe alcun problema in quanto l’utente sarà guidato dal sistema alla corretta reimputazione.

La form di presentazione, che utilizza una tecnologia “touch screen”, si presenta da subito molto intuitiva, con pulsanti grandi e menu chiari e di facile gestione, realizzati con una grafica che renda il tutto più semplice e piacevole.

### 3.3.2 Affidabilità

Il sistema deve garantire affidabilità dei servizi proposti. Il software sarà sviluppato in modo tale da controllare accuratamente le informazioni date in input dagli utenti. Tali controlli riguarderanno in generale l’aspetto formale (ad esempio la coerenza delle date) e saranno applicati sia alle informazioni immesse dai clienti sia a quelle di carattere gestionale. L’utente che commette un errore in fase di input sarà avvisato tempestivamente e invitato a reinserire le informazioni risultate incorrette.

Un esempio di errore da parte del cliente può essere il tentativo di prenotarsi ad un’attività extra dell’albergo quando il tempo disponibile per la prenotazione è scaduto: in questo caso il sistema dovrà informarlo dell’errore e non dovrà consentirgli di portare a termine l’operazione.

Un altro esempio, più estremo, potrebbe essere la richiesta contemporanea, da parte di due utenti, di un particolare prodotto del bar: se al momento dell’ordinazione la disponibilità del prodotto in questione non è tale da soddisfare entrambe le richieste, il sistema dovrà ricorrere al principio di mutua esclusione e decidere quale dei due servire onde evitare che il sistema stesso entri in uno stato anomalo compromettendo, così, il suo corretto funzionamento.

### 3.3.3 Performance

Il sistema non presenta particolari esigenze né di funzionamento multipiattaforma né di funzionamento real-time, per cui non sono richieste specifiche caratteristiche hardware.

### 3.3.4 Supportabilità

Le funzionalità offerte dal sistema Quick Service potranno essere integrate da altre in futuro grazie ad alcune modifiche. Possibili funzionalità aggiuntive sono le seguenti:

1. Utilizzo di tesserini magnetici, per ogni cliente dell'albergo, per poter effettuare consumazioni al bar, al ristorante o ad altri servizi, offerti dall'albergo, per poter addebitare il costo di tali servizi sul conto del cliente.
2. Automatizzazione delle ordinazioni da parte dei camerieri, dotandoli di palmari, smistano le ordinazioni direttamente alla cucina (adattamento del software di ordinazione per i palmari).
3. Possibilità di aggiungere altri servizi, come per esempio le iniziative ricreative, proposte dall'albergo.

Principalmente le modifiche riguarderanno le parti del sistema addette alla prenotazione (per l'eventuale utilizzo dei palmari) e al saldo del cliente (per poter permettere di addebitare le varie consumazioni fatte, con la scheda magnetica, permettendo anche di saldare il conto). Per quanto riguarda la realizzazione di nuovi servizi sarà necessaria una progettazione di un sottosistema che andrà aggiunta al software già realizzato.

### 3.3.5 Implementazione

Dal momento che l'architettura del sistema informativo che ci proponiamo di realizzare è basata sul modello “client-server”, è possibile individuare due tipologie di configurazioni hardware: una relativa alla macchina dedicata all'amministrazione del sistema (server) e l'altra relativa ai terminali touch-screen (client). Inoltre è richiesta l'esistenza di una rete di interconnessione tra il server e i vari client.

La macchina server deve essere sufficientemente potente da consentire l'esecuzione dell'intero sistema (compreso il DBMS per la gestione del database) e deve garantire al sistema stesso di poter gestire eventuali richieste concorrenti provenienti dai client. A tale scopo la configurazione hardware minima richiesta è la seguente:

- Processore Pentium IV (o equivalente) con frequenza di 2 GH
- 512 MB di memoria RAM
- Disco rigido da 40 Gb
- Scheda video e monitor in grado di supportare una risoluzione di almeno 1024x768 pixel con almeno 65 mila colori

Tutte le fasi operative che vanno dall'ordinazione fino al calcolo del conto finale saranno gestite dalla macchina server. Inoltre da tale postazione sarà possibile gestire i listini, gli ordini, i pagamenti e smistare le ordinazioni alle relative destinazioni (bar o cucina).

Da tutte le altre postazioni (client) i clienti dell'albergo potranno effettuare operazioni quali: ordinazioni al bar, ordinazioni al ristorante e prenotazioni delle escursioni. Alla luce di ciò non sono richieste particolari caratteristiche hardware, se non la presenza dei touch-screen.

Se si desiderasse di offrire la possibilità ai clienti di accedere ad Internet attraverso la rete locale allora sarà di vitale importanza, per ovvi motivi di sicurezza, disporre di una macchina dedicata al servizio di firewall per proteggere il server da accessi esterni non autorizzati ed evitare che virus o attacchi mirati di persone malintenzionate compromettano l'integrità e il corretto funzionamento del sistema.

### 3.3.6 Interfaccia

Per una gestione corretta e funzionale dello scambio delle informazioni tra gli utenti e il sistema sono state elaborate le seguenti linee guida:

1. Prevedere dei form che consentano agli utenti di usufruire delle diverse funzionalità scegliendo tra varie opzioni prestabilite, riducendo, in questo modo, il più possibile la necessità dell'immissione diretta dei dati e, di conseguenza, la possibilità di commettere errori.
2. Nel caso specifico delle funzionalità rivolte ai gestori (albergo, bar e cucina) saranno realizzati dei controlli accurati sui dati immessi per evitare che vengano inseriti valori errati che possano provocare malfunzionamenti al sistema.
3. Evitare che il sistema accetti dati provenienti dall'esterno.

Nei casi in cui l'immissione diretta dei dati è inevitabile si farà ricorso ad opportune componenti grafiche per l'immissione controllata (o mascherata) dei dati. In questo modo si riuscirà a ridurre la complessità del sistema (sarà possibile, infatti, ridurre notevolmente la granularità dei controlli) e, allo stesso tempo, si ridurrà notevolmente la quantità di messaggi di errore da proporre agli utenti. Ad esempio, è possibile stabilire un particolare formato per l'inserimento di date e orari e demandare a tali componenti la verifica della coerenza e della correttezza dei dati immessi.

### 3.3.7 Packaging

Per quanto riguarda il problema delle risorse, bisognerà seguire le seguenti considerazioni:

1. Il sistema, formato da parte software e hardware organizzato in un certo modo, sarà installato da un team specializzato con le conoscenze di funzionamento del sistema.
2. La manutenzione del sistema dovrebbe essere effettuata annualmente per controllare e aggiornare il sistema per il nuovo anno.
3. All'interno dell'albergo può essere selezionato un addetto particolare che potrebbe controllare il funzionamento del sistema (avvisando l'assistenza tecnica del software per informarli di eventuali malfunzionamenti) e per effettuare il backup giornaliero.

### 3.3.8 Legali

Il sistema sarà realizzato rispettando la privacy dell'utente.

## 3.4 Vincoli

Abbiamo previsto le seguenti costrizioni:

- Documentazione: qualsiasi software per la stila di documenti
- Diagrammi UML: utilizzo di ArgoUML
- Implementazione: utilizzo di JBuilder o qualsiasi altro pacchetto compatibile Java

Si è scelto tale linguaggio perché esso presenta dei requisiti particolari che bene si adattano per la realizzazione del programma, infatti:

- ✓ Permette di realizzare e gestire molto facilmente ed in maniera efficiente le applicazioni client-server.
- ✓ È un linguaggio orientato agli oggetti.
- ✓ Presenza una buona pulizia stilistica.
- ✓ È indipendente dall'architettura hardware e software sottostante (portabilità del codice grazie alla presenza della Java Virtual Machine).
- ✓ Facile interfacciamento con i database mediante JDBC.
- ✓ È utilizzato per la progettazione orientata alla rete ed offre peraltro la possibilità di realizzare interfacce utente personalizzabili.
- Librerie: qualsiasi tipo di libreria compatibile con Java
- Sistema: architettura client-server implementata tramite RMI

## 3.5 Modello di sistema

### 3.5.1 Scenari

Prima di descrivere dettagliatamente gli scenari viene presentata una sorta di “storia” che ci introduce allo scenario.

#### *Servizio Ristorante*

Come ogni mattina Franco, il gestore del ristorante, è davanti al suo terminale, effettua la valida e inserisce i menù del giorno. Accorgendosi però, che le forniture richieste sono state consegnate, decide di modificare i menù.

Nel frattempo Mario, uno dei clienti, pensa di ordinare il pranzo per se e per il suo compagno di stanza Carlo. Attraverso il Touch Screen della sua camera, visualizza il menù del giorno ed effettua la sua prenotazione.

A causa di un imprevisto però, Carlo non riesce a ritornare in albergo per ora di pranzo, quindi Mario è costretto ad annullare la prenotazione dell'amico e a modificare il suo menù poiché pranzare da solo si annoia.

Nell'attesa che arrivi il pranzo Mario decide di controllare il suo saldo e quindi ritorna a consultare il suo terminale.

Alle ore 12.00, invece, Franco essendo scaduto il tempo limite per le ordinazioni, controlla tutte le prenotazioni dei clienti in modo poter passare gli ordini alla cucina.

### Scenario Gestione Ristorante

<b>Nome dello scenario:</b>	visualizzazione_menu
<b>Attori Partecipanti:</b>	Emilio: Gestore
	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Giacomo dalla sua camera ed Emilio dal suo terminale, desiderano consultare il menù, quindi cliccano su <b>Visualizza Menù</b> .
	2. Il sistema esegue l'operazione richiesta.

<b>Nome dello scenario:</b>	prenotazione_menu
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Dopo aver visualizzato il menù Giacomo sceglie le pietanze e specificando di voler usufruire del servizio in camera, clicca su <b>Ordina</b> .
	2. Il sistema esegue l'ordinazione.

<b>Nome dello scenario:</b>	visualizza_ordinazioni
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Non ricordando cosa ha ordinato per il pranzo, Giacomo dalla sua camera clicca su <b>Visualizza Ordinazioni</b> .
	2. Il sistema visualizza la sua ordinazione.

<b>Nome dello scenario:</b>	modifica_ordinazione
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Dopo aver visualizzato le sue ordinazioni

	(visualizza_ordinazioni), Giacomo, decide di cambiare una delle pietanze prenotate e clicca su <b>Modifica Ordinazione</b> .
	2. Il sistema lancia un avviso di modifica.
	3. Giacomo risponde <b>Sì</b> .
	4. Il sistema modifica la prenotazione.

<b>Nome dello scenario:</b>	annulla_ordinazioni
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Giacomo annulla la propria cena ed utilizza il comando <b>Visualizza Ordinazioni</b>.</li> <li>2. Il sistema visualizza le ordinazioni.</li> <li>3. Clicca sulla cena e poi su <b>Annulla Ordinazione</b>.</li> <li>4. Il sistema chiede conferma d'annullamento.</li> <li>5. Giacomo risponde <b>Sì</b>.</li> <li>6. Il sistema annulla la prenotazione.</li> </ol>

<b>Nome dello scenario:</b>	consultazione_prenotazioni
<b>Attori Partecipanti:</b>	Giuseppe: Addetto Ristorante
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Giuseppe consulta le prenotazioni cliccando su <b>Prenotazioni</b>.</li> <li>2. Il sistema elenca tutte le prenotazioni.</li> </ol>

<b>Nome dello scenario:</b>	inserimento_menu
<b>Attori Partecipanti:</b>	Giuseppe: Addetto Ristorante
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Giuseppe inserisce il menù del giorno selezionando <b>Nuovo Menù</b>.</li> <li>2. Il sistema restituisce una form vuota.</li> <li>3. L'addetto inserisce i dati e clicca su <b>Conferma</b>.</li> <li>4. Il sistema ha registrato il nuovo menù.</li> </ol>

<b>Nome dello scenario:</b>	modifica_menu
<b>Attori Partecipanti:</b>	Giuseppe: Addetto Ristorante
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Leggendo il menù (visualizza_menu) Giuseppe decide di cambiare alcune pietanze e clicca su <b>Modifica</b>.</li> <li>2. Il sistema restituisce una form già compilata.</li> <li>3. Giuseppe modifica i campi che desidera e clicca su <b>Conferma</b>.</li> <li>4. Il sistema chiede conferma per la modifica.</li> </ol>

	5. L'addetto risponde <b>Sì</b> .
	6. Il sistema modifica il menù.

## ***Servizio Camere***

Luca dopo aver affrontato un lungo viaggio in sella alla sua motocicletta decide di fermarsi qualche giorno per riposarsi.

Si reca quindi alla reception per prenotare una stanza e il portiere tramite il suo computer controlla la disponibilità delle varie camere. Trovatane una adatta, il portiere procede con la compilazione dei dati personali di Luca che decide di affittare la stanza a mezza pensione.

Il giorno successivo, Luca, accortosi delle ottime pietanze servite dal ristorante decide di continuare la sua permanenza a pensione completa, quindi si reca dal portiere che effettua dal suo terminale questa modifica.

Dopo 3 giorni Luca decide di ripartire, riconsegna le chiavi al portiere che dal suo computer visualizza il saldo della sua camera e regolati tutti i conti, il portiere libera la camera.

### **Scenari Gestione Camere**

<b>Nome dello scenario:</b>	assegna_camera
<b>Attori Partecipanti:</b>	Emilio: Gestore
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>Emilio deve assegnare una stanza, compila gli appositi campi nel form e clicca su <b>Assegna Camera</b> .</li> <li>Il sistema elenca le camere disponibili.</li> <li>Emilio seleziona la camera da assegnare e clicca su <b>Conferma</b>.</li> <li>Il sistema assegna la camera.</li> </ol>

<b>Nome dello scenario:</b>	visualizza_camera
<b>Attori Partecipanti:</b>	Emilio: Gestore Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>Giacomo decide di controllare il suo saldo alla reception.</li> <li>Emilio deve controllare il saldo di un cliente. Dal suo terminale seleziona la stanza da lui occupata e clicca su <b>Visualizza</b>.</li> <li>Il terminale mostra tutte le informazioni.</li> </ol>

<b>Nome dello scenario:</b>	modifica_camera
<b>Attori Partecipanti:</b>	Emilio: Gestore Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>Emilio cambia il tipo di prenotazione di un cliente da mezza pensione a pensione completa tramite il suo terminale selezionando la stanza da lui occupata e clicca su <b>Modifica</b>.</li> <li>Il sistema visualizza tutte le informazioni riguardanti la camera.</li> </ol>

	3. Emilio modifica le opzioni riguardanti il tipo di permanenza e clicca su <b>Conferma</b> . 4. Il sistema lancia un avviso di modifica. 5. Emilio accetta cliccando sul <b>Sì</b> . 6. Il sistema esegue l'operazione richiesta.
--	---

<b>Nome dello scenario:</b>	libera_camera
<b>Attori Partecipanti:</b>	Emilio: Gestore
<b>Flusso di eventi:</b>	1. Un cliente lascia la sua camera ed Emilio seleziona la stanza cliccando su <b>Libera Camera</b> . 2. Il sistema chiede conferma. 3. Emilio accetta cliccando sul <b>Sì</b> . 4. Il sistema rende nuovamente disponibile la camera.

## ***Servizio Bar***

Antonio è nella sua camera e ha voglia di qualcosa da bere, allora con il touch screen della sua camera visualizza il listino del bar, decide di prendere una coca cola e degli arachidi, allora seleziona i prodotti e li ordina. Dopo un po' si ricorda di non poter mangiare gli arachidi allora prima che il tempo per una possibile modifica scada, modifica la sua prenotazione, togliendo gli arachidi.

Intanto al bar, l'addetto si occupa dei nuovi prodotti arrivati nella mattinata e poiché li vuole aggiungere al listino del bar, esegue la valida dal suo terminale e inserisce i nuovissimi tipi di birra arrivati nella mattinata, cancella un vecchio tipo di patatine che non sono state più ordinate e in fine modifica tutte le quantità, aggiungendo 20 cole e 12 succhi di frutta all'arancia.

Mentre Antonio sorseggia lentamente la sua coca cola gli viene voglia di qualcosa da mangiare allora va di nuovo a guardare il listino del bar e sceglie dei biscotti, li ordina, ma a quel punto non ricorda quanto ha speso nell'albergo ed allora controlla il suo saldo e vedendo che ha superato le previsioni di spesa, pensa che sia meglio non prendere quei biscotti ed allora di nuovo con il suo touch screen annulla la sua prenotazione.

## **Scenario Gestione Bar**

<b>Nome dello scenario:</b>	visualizza_listino
<b>Attori Partecipanti:</b>	Luca: Addetto Bar Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Giacomo dalla sua camera e Luca dal suo terminale vogliono visualizzare il listino del bar e cliccano su <b>Visualizza Listino</b> . 2. Il sistema svolge l'operazione richiesta.

<b>Nome dello scenario:</b>	nuovo_ordine
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato il listino (visualizza_listino) Giacomo seleziona una bibita e clicca su ordina.</li> <li>2. Il sistema esegue l'ordine.</li> </ol>

<b>Nome dello scenario:</b>	visualizza_ordine
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dalla sua camera Giacomo vuole visualizzare le cose che ha ordinato al bar, quindi clicca su <b>Visualizza Ordinazioni</b>.</li> <li>2. Il sistema restituisce la lista dei prodotti ordinati.</li> </ol>

<b>Nome dello scenario:</b>	modifica_ordine
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato le ordinazioni già effettuate (visualizza_ordine), Giacomo clicca su <b>Modifica Ordinazioni</b> per modificare l'ordinazione che aveva effettuato.</li> <li>2. Il sistema visualizza la lista dei prodotti ordinati.</li> <li>3. Giacomo seleziona o de-seleziona i prodotti e clicca su <b>Conferma</b>.</li> <li>4. Il sistema lancia un avviso di modifica.</li> <li>5. Giacomo risponde <b>Sì</b>.</li> <li>6. Il sistema modifica la prenotazione.</li> </ol>

<b>Nome dello scenario:</b>	cancella_ordine
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato le ordinazioni già effettuate (visualizza_ordine), Giacomo clicca su <b>Cancella Ordinazione</b> per cancellare l'ordinazione che aveva effettuato.</li> <li>2. Il sistema lancia un avviso di cancellazione.</li> <li>3. Giacomo clicca su <b>sì</b>.</li> <li>4. Il sistema effettua le operazioni richieste.</li> </ol>

<b>Nome dello scenario:</b>	inserimento_prodotti
<b>Attori Partecipanti:</b>	Luca: Addetto Bar
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Luca vuole inserire nuovi prodotti nel listino e clicca su <b>Nuovo Prodotto.</b></li> <li>2. Il sistema restituisce una form vuota.</li> <li>3. Luca la compila in tutte le sue parti e clicca su <b>Conferma.</b></li> <li>4. Il sistema svolge l'operazione richiesta.</li> </ol>

<b>Nome dello scenario:</b>	modifica_prodotti
<b>Attori Partecipanti:</b>	Luca: Addetto Bar
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato il listino (visualizza_listino), Luca seleziona una prodotto e clicca su <b>Modifica</b> per modificarlo.</li> <li>2. Il sistema restituisce una form già compilata.</li> <li>3. Luca la modifica e clicca su <b>Conferma.</b></li> <li>4. Il sistema lancia un avviso di modifica.</li> <li>5. Luca risponde <b>Sì.</b></li> <li>6. Il sistema modifica il/i prodotto/i.</li> </ol>

<b>Nome dello scenario:</b>	cancella_prodotti
<b>Attori Partecipanti:</b>	Luca: Addetto Bar
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato il listino (visualizza_listino), Luca seleziona uno o più prodotti e clicca su <b>Cancella Prodotti</b> per cancellarlo/i.</li> <li>2. Il sistema lancia un avviso di cancellazione.</li> <li>3. Luca risponde <b>Sì.</b></li> <li>4. Il sistema cancella il/i prodotto/i dal listino.</li> </ol>

## **Servizio Prenotazione Escursioni e Attività Simili.**

Mario il gestore dell'albergo effettua la valida e visualizza tutte le attività già attivate poiché vuole pensare a qualche nuova attività da inserire, mentre ci pensa nota che non c'è nessuna prenotazione per la visita ad Amalfi, allora pensa bene di rimuovere quell'attività. Cancellata l'attività, crede che un'escursione sul Vesuvio in questa stagione possa interessare ai propri clienti ed allora compilando tutti i dati aggiunge fra le attività anche questa.

Rileggendo tutte le attività vede che i posti per la visita agli scavi di Paestum sono esauriti ed allora va a controllare la resa economica di questa attività guardando il rispettivo resoconto.

Alessandro dalla sua camera, toccando il suo touch screen, guarda l'elenco di tutte le attività e pensa che la gita a Paestum sia molto interessante allora prenota per se e per suo figlio Andrea, specificando di volere il pranzo a sacco. Dopo qualche ora leggendo su un opuscolo di un ottimo ristorante in quella zona, decide di modificare la sua prenotazione, facendone una senza pranzo per approfittare di quel buon ristorante. Il giorno dopo Andrea essendo stato troppo tempo a giocare al sole con i suoi amichetti, prende un'insolazione, così che suo padre prende la decisione di non andare più a Paestum quindi, appena torna nella sua stanza annulla la loro prenotazione.

### **Scenario Gestione Attività Extra**

<b>Nome dello scenario:</b>	visualizza_attività
<b>Attori Partecipanti:</b>	Emilio: Gestore
	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Emilio dal suo terminale e Giacomo dalla sua camera visualizzano un'attività extra selezionandone prima una dalla lista e poi cliccando su <b>Attività Extra</b> .
	2. Il sistema visualizza i dettagli di tale attività.

<b>Nome dello scenario:</b>	nuova_prenotazione
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Dopo aver visualizzato l'attività (visualizza_attività), Giacomo la prenota cliccando su <b>Prenota</b> .
	2. Il sistema restituisce una form non compilata. 3. Giacomo compila essa in tutte le sue parti e clicca su <b>Conferma</b> . 4. Il sistema effettua la prenotazione.

<b>Nome dello scenario:</b>	visualizza_prenotazione
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	1. Giacomo visualizza le prenotazioni già effettuate cliccando

	su <b>Visualizza Prenotazioni</b> .
	2. Il sistema svolge l'operazione richiesta.

<b>Nome dello scenario:</b>	modifica_prenotazione
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato le prenotazioni (<b>visualizza_prenotazione</b>), Giacomo ne seleziona una e clicca su <b>Modifica</b> per modificarla.</li> <li>2. Il sistema restituisce una form già compilata.</li> <li>3. Giacomo modifica essa in tutte le sue parti e clicca su <b>Conferma</b>.</li> <li>4. Il sistema lancia un avviso di modifica.</li> <li>5. Giacomo risponde <b>Sì</b>.</li> <li>6. Il sistema modifica la prenotazione.</li> </ol>

<b>Nome dello scenario:</b>	annullamento_prenotazione
<b>Attori Partecipanti:</b>	Giacomo: Cliente
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato le prenotazioni (<b>visualizza_prenotazione</b>), Giacomo ne seleziona una e clicca su <b>Cancella</b> per cancellarla.</li> <li>2. Il sistema lancia un avviso di cancellazione.</li> <li>3. Giacomo risponde <b>Sì</b>.</li> <li>4. Il sistema cancella la prenotazione.</li> </ol>

<b>Nome dello scenario:</b>	inserimento_attività
<b>Attori Partecipanti:</b>	Emilio: Gestore
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Emilio vuole inserire una nuova attività, quindi clicca su <b>Nuova Attività</b>.</li> <li>2. Il sistema restituisce una form non compilata.</li> <li>3. Emilio la compila in tutte le sue parti e clicca su <b>Conferma</b>.</li> <li>4. Il sistema inserisce la nuova attività.</li> </ol>

<b>Nome dello scenario:</b>	modifica_attività
<b>Attori Partecipanti:</b>	Emilio: Gestore
<b>Flusso di eventi:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver visualizzato un'attività (<b>visualizza_attività</b>), Emilio clicca su <b>Modifica</b> per modificarla.</li> <li>2. Il sistema restituisce una form non compilata.</li> <li>3. Emilio modifica essa e clicca su <b>Conferma</b>.</li> </ol>

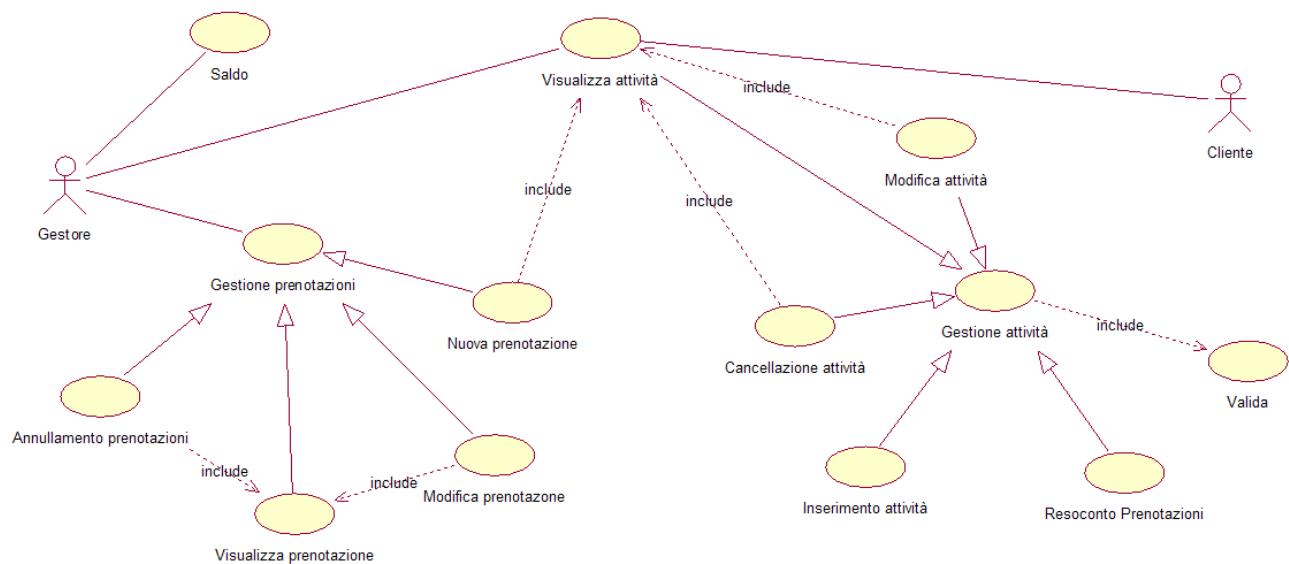
	4. Il sistema lancia un avviso di modifica. 5. Emilio risponde <b>Sì</b> . 6. Il sistema effettua l'operazione richiesta.
--	---

<b>Nome dello scenario:</b>	cancellazione_Attività
<b>Attori Partecipanti:</b>	Emilio: Gestore
<b>Flusso di eventi:</b>	1. Dopo aver visualizzato un'attività (visualizza_attività), Emilio clicca su <b>Cancella</b> per cancellarla. 2. Il sistema lancia un avviso di cancellazione. 3. Emilio risponde <b>Sì</b> . 4. Il sistema rimuove l'attività.

<b>Nome dello scenario:</b>	resoconto_prenotazioni
<b>Attori Partecipanti:</b>	Emilio: Gestore
<b>Flusso di eventi:</b>	1. Emilio visualizza il resoconto delle prenotazioni selezionando un'attività e poi cliccando su <b>Resoconto</b> . 2. Il sistema restituisce la lista delle prenotazioni effettuate ed il guadagno fino a quel momento.

### 3.5.2 Modello dei casi d'uso

#### 3.5.2.1 Modello Use-Case Attività Extra



#### Attori

Attore	Versione	Autore
Cliente	1.0	Iacoletti Alessandro Pacifico Marta
Specializza l'attore:		
	Nessuno	
Caso d'uso	Primario	Frequenza
1. Visualizza Attività	X	Alta
2. Nuova Prenotazione	X	Media
3. Modifica Prenotazione	X	Bassa
4. Annullamento Prenotazione	X	Bassa
5. Visualizza Prenotazione	X	Media
6. Saldo	X	Media

Attore	Versione	Autore
Gestore dell'Albergo	1.0	Iacoletti Alessandro Pacifico Marta
Specializza l'attore:		Nessuno
Caso d'uso	Primario	Frequenza
1. Visualizza Attività	X	Alta
2. Valida		
3. Inserimento Attività	X	Media
4. Modifica Attività	X	Media
5. Cancellazione Attività	X	Bassa
6. Resoconto Prenotazioni	X	Bassa
		Media

## Organizzazione del Modello

Gestione prenotazioni

1. Nuova Prenotazione
2. Modifica prenotazione
3. Annullamento prenotazione
4. Visualizza Prenotazione
5. Resoconto Prenotazione

Gestione attività

1. Visualizza Attività
2. Valida
3. Inserimento attività
4. Modifica attività
5. Cancellazione attività

## Modulo < Gestione Prenotazioni >

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>GEST_PREN</i>
	Use Cases Nome:	<i>Gestione Prenotazioni</i>
	Autori:	<i>Iacoletti Alessandro</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al cliente di inserire, annullare, modificare una prenotazione dell'attività desiderata</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole prenotare una o più attività extra dell'albergo.</i>
	Condizioni: Pre-condizioni	<i>Deve essere presente almeno un'attività nel menu</i>
	Post-condizioni di successo:	<i>L'utente riesce a compiere le operazioni desiderate. I dati vengono salvati</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che l'operazione non è stata effettuata. I dati non vengono salvati</i>
	Priorità	<i>Alta</i>
	Casi d'uso inclusi:	<i>Visualizza Attività</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Nessuno</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1		(Vedi casi d'uso che specializzano questo)
Iterazioni			
Autore	Data	Descrizione	
<i>Iacoletti Alessandro</i>	<i>04/04/2008</i>	<i>Prima versione del caso d'uso</i>	
<i>Pacifico Marta</i>	<i>07/04/2008</i>	<i>Revisione dei casi d'uso</i>	

## Modulo < Nuova Prenotazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>PRENOTA</i>
	Use Cases Nome:	<i>Nuova Prenotazione</i>
	Autori:	<i>Pacifico Marta</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al cliente di prenotare l'attività desiderata</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole prenotare una o più attività extra dell'albergo.</i>
	Condizioni: Pre-condizioni	<i>Ci deve essere almeno una attività</i>
	Post-condizioni di successo:	<i>L'utente riesce a prenotarsi per le attività desiderate. I dati vengono salvati</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la prenotazione non è stata effettuata. I dati non vengono salvati</i>
	Priorità	<i>Media</i>
	Casi d'uso inclusi:	<i>Visualizza Attività</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Prenotazioni</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Seleziona la funzione prenota attività.</i>
	2	<i>Sistema</i>	<i>Include &lt;Visualizza Attività &gt;</i>
	3	<i>Sistema</i>	<i>Visualizza la form di prenotazione dell'attività.</i>
	4	<i>Cliente</i>	<i>Compila la form in ogni sua parte</i>
	5	<i>Cliente</i>	<i>Invia la prenotazione</i>
	6	<i>Sistema</i>	<i>Registra i dati e informa l'utente che la prenotazione è stata effettuata</i>
	1° Scenario alternativo: Campi obbligatori non compi lati		
	Passo	Soggetto	Descrizione
	4.1	<i>Sistema</i>	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>
	4.2	<i>Sistema</i>	<i>Ritorna al punto 3</i>
	2° Scenario alternativo: Prenotazione non eseguibile		
	3.1	<i>Sistema</i>	<i>Informa l'utente che non è più possibile effettuare una prenotazione (Tempo disponibile scaduto o posti esauriti)</i>

	3.2	<i>Sistema</i>	<i>Ritorna al punto 3</i>		
<b>3° Scenario alternativo: Errore del sistema</b>					
	5.1	<i>Sistema</i>	<i>Informa l'utente che la prenotazione non è stata effettuata per un errore nel sistema</i>		
	5.2	<i>Sistema</i>	<i>Ritorna al punto 3</i>		
Annotazioni					
Descrizione					
	3.1				
		<i>L'utente deve compilare la form in tutte le sue parti, ed eventualmente specificare campi opzionali.(Adulto/Bambino, Prenotazione Pranzo, ecc...)</i>			
Iterazioni					
Autore      Data      Descrizione					
<i>Pacifico Marta</i>		<i>04/04/2008</i>	<i>Prima versione del caso d'uso</i>		
<i>Pacifico Marta</i>		<i>07/04/2008</i>	<i>Revisione dei casi d'uso</i>		

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>MOD_PREN</i>
	Use Cases Nome:	<i>Modifica Prenotazione</i>
	Autori:	<i>Iacoletti Alessandro</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al cliente di modificare la prenotazione</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole modificare la prenotazione di una o più attività extra dell'albergo.</i>
	Condizioni: Pre-condizioni	<i>Deve essere stata prenotata almeno una attività</i>
	Post-condizioni di successo:	<i>L'utente riesce a modificare la prenotazione. I dati vengono salvati</i>
	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la modifica non è stata effettuata. I dati non vengono salvati.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta</i>
	Casi d'uso inclusi:	<i>Visualizza Attività</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Prenotazioni</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Seleziona la funzione modifica prenotazione.</i>
	2	<i>Sistema</i>	<i>Include &lt; Visualizza Prenotazione &gt;</i>
	3	<i>Cliente</i>	<i>Sceglie la prenotazione da modificare</i>
	4	<i>Sistema</i>	<i>Visualizza la form della prenotazione</i>
	5	<i>Cliente</i>	<i>Modifica i campi desiderati</i>
	6	<i>Cliente</i>	<i>Conferma la modifica</i>
1° Scenario alternativo: Nessuna prenotazione effettuata in precedenza	7	<i>Sistema</i>	<i>Registra i dati e informa l'utente che la modifica è stata effettuata</i>
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa l'utente che non è stata effettuata nessuna prenotazione in precedenza</i>
2° Scenario alternativo: Campi obbligatori non compilati	2.2	<i>Sistema</i>	<i>Ritorna al menù principale</i>
	7.1	<i>Sistema</i>	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>

	7.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
<b>3° Scenario alternativo: Prenotazione non eseguibile</b>			
	4.1	<i>Sistema</i>	<i>Informa l'utente che non è più possibile modificare la prenotazione o effettuare questo tipo di modifica (Tempo disponibile scaduto o posti esauriti)</i>
	4.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
<b>4° Scenario alternativo: Errore del sistema</b>			
	6.1	<i>Sistema</i>	<i>Informa l'utente che la modifica non è stata effettuata per un errore nel sistema</i>
	6.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
Iterazioni			
	Autore	Data	Descrizione
	Iacoletti Alessandro	04/04/2008	Prima versione del caso d'uso
	Iacoletti Alessandro	07/04/2008	Revisione dei casi d'uso

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>ANN_PRE</i>
	Use Cases Nome:	<i>Annulloamento Prenotazione</i>
	Autori:	<i>Pacifico Marta</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al cliente di annullare la prenotazione</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole annullare la prenotazione di una o più attività extra dell'albergo.</i>
	Condizioni: Pre-condizioni	<i>Deve essere stata prenotata almeno una attività</i>
	Post-condizioni di successo:	<i>L'utente riesce ad annullare la prenotazione. I dati vengono salvati</i>
	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la cancellazione non è stata effettuata. Il sistema rimane stabile</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta</i>
	Casi d'uso inclusi:	<i>Visualizzazione</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Prenotazioni</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Seleziona la funzione annulla prenotazione.</i>
	2	<i>Sistema</i>	<i>Include &lt; Visualizza Prenotazione &gt;</i>
	3	<i>Cliente</i>	<i>Sceglie la prenotazione da annullare</i>
	4	<i>Sistema</i>	<i>Chiede all'utente se è sicuro di annullare la prenotazione</i>
	5	<i>Cliente</i>	<i>Conferma l'annullamento</i>
	6	<i>Sistema</i>	<i>Cancello la prenotazione e i dati forniti dal cliente e lo informa che l'annullamento è stato effettuato</i>
	1° Scenario alternativo: Nessuna prenotazione effettuata in precedenza		
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa l'utente che non è stata effettuata nessuna prenotazione in precedenza</i>
	2.2	<i>Sistema</i>	<i>Ritorna al menu principale</i>
	2° Scenario alternativo: Annullamento non confermato		
	5.1	<i>Cliente</i>	<i>Non conferma l'annullamento</i>
	5.2	<i>Sistema</i>	<i>Ritorna al punto 2</i>

	3° Scenario alternativo: Nessuna prenotazione effettuata in precedenza		
6.1	Sistema	<i>Informa l'utente che non è più possibile annullare la prenotazione (Tempo disponibile scaduto)</i>	
6.2	Sistema	<i>Ritorna al punto 2</i>	
4° Scenario alternativo: Errore del Sistema			
6.1	Sistema	<i>Informa l'utente che la cancellazione non è stata effettuata per un errore nel sistema</i>	
6.2	Sistema	<i>Ritorna al punto 2</i>	
Iterazioni			
Autore	Data	Descrizione	
Pacifico Marta	04/04/2008	<i>Prima versione del caso d'uso</i>	
Pacifico Marta	07/04/2008	<i>Revisione dei casi d'uso</i>	

## Modulo <Visualizza Prenotazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>VIS_PREN</i>
	Use Cases Nome:	<i>Visualizza Prenotazione</i>
	Autori:	<i>Vicidomini Vincenzo</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Visualizzare l'elenco delle attività extra prenotate in precedenza dal cliente.</i>
	Attori Primario:	<i>Cliente</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vogliono visualizzare informazioni relative alle prenotazioni effettuate delle attività extra dell'albergo</i>
	Condizioni: Pre-condizioni	<i>Ci deve essere almeno una attività extra prenotata.</i>
	Post-condizioni di successo:	<i>L'utente riesce a leggere le informazioni riguardanti le attività prenotate</i>
	Post-condizioni per Fallimento:	<i>In caso di fallimento nessuna attività verrà visualizzata</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Bassa</i>
	Casi d'uso inclusi:	<i>Nessuno</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Prenotazioni</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

<b>Descrizione degli scenari</b>	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Seleziona la funzione visualizza prenotazioni.</i>
	2	<i>Sistema</i>	<i>Visualizzazione delle attività prenotate.</i>
	3	<i>Cliente/Gestore</i>	<i>Seleziona una attività per avere maggiori informazioni</i>
	4	<i>Sistema</i>	<i>Visualizza la form della prenotazione selezionata</i>
Scenario alternativo: Nessuna prenotazione effettuata in precedenza	Scenario alternativo: Nessuna prenotazione effettuata in precedenza		
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa l'utente che non si è prenotato per nessuna attività.</i>
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
Iterazioni	Iterazioni		
	Autore	Data	Descrizione

	<i>Vicidomini Vincenzo</i>	04/04/2008	<i>Prima versione del caso d'uso</i>
	<i>Vicidomini Vincenzo</i>	07/04/2008	<i>Revisione dei casi d'uso</i>

## Modulo <Resoconto Prenotazioni>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>RES_PREN</i>
	Use Cases Nome:	<i>Resoconto Prenotazioni</i>
	Autori:	<i>Pacifico Marta</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere ai gestori di avere un resoconto di tutti i prenotati ad una certa attività</i>
	Attori Primario:	<i>Gestore</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole visualizzare un resoconto dei prenotati</i>
	Condizioni: Pre-condizioni	<i>L'utente deve aver immesso login e password valide. Almeno un cliente si è prenotato</i>
	Post-condizioni di successo:	<i>L'utente riesce a visualizzare il resoconto</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema avvisa il cliente che il resoconto non può essere visualizzato</i>
	Priorità	<i>Bassa</i>
	Casi d'uso inclusi:	<i>Valida Visualizza Attività</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Attività</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Sistema</i>	<i>Include &lt;Visualizza Attività&gt;</i>
	2	<i>Gestore</i>	<i>Sceglie l'attività ci cui vuole conoscere il resoconto</i>
	3	<i>Sistema</i>	<i>Visualizza il resoconto di tale attività</i>
1° Scenario alternativo: Nessun cliente prenotato			
	Passo	Soggetto	Descrizione
	3.1	<i>Sistema</i>	<i>Nessun cliente si è prenotato per tale attività</i>
	3.2	<i>Sistema</i>	<i>Ritorna al menu principale</i>
2° Scenario alternativo: Cancellazione non conferma ta			

	3.1	<i>Sistema</i>	<i>Informa l'utente che il resoconto non può essere visualizzato per un errore del sistema</i>
	3.2	<i>Sistema</i>	<i>Ritorna al passo 2</i>
<b>Iterazioni</b>			
	Autore	Data	Descrizione
	<i>Pacifico Marta</i>	<i>04/04/2008</i>	<i>Prima versione del caso d'uso</i>
	<i>Pacifico Marta</i>	<i>07/04/2008</i>	<i>Revisione dei casi d'uso</i>

## Modulo < Gestione Attività >

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>GEST_ATT</i>
	Use Cases Nome:	<i>Gestione Attività</i>
	Autori:	<i>Vicidomini Vincenzo</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al gestore di inserire, annullare, modificare un'attività extra nel menu</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole inserire/modificare una o più attività extra dell'albergo.</i>
	Condizioni: Pre-condizioni	<i>L'utente deve aver immesso login e password valide</i>
	Post-condizioni di successo:	<i>L'utente riesce a compiere le operazioni desiderate</i>
	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che l'operazione non è stata effettuata.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta</i>
	Casi d'uso inclusi:	<i>Visualizza Attività</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Nessuno</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1		<i>(Vedi casi d'uso che specializzano questo)</i>
Iterazioni			
Autore	Data	Descrizione	
<i>Vicidomini Vincenzo</i>	<i>04/04/2008</i>	<i>Prima versione del caso d'uso</i>	
<i>Vicidomini Vincenzo</i>	<i>07/04/2008</i>	<i>Revisione dei casi d'uso</i>	

## Modulo <Visualizza Attività>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>VIS_ATT</i>
	Use Cases Nome:	<i>Visualizza Attività</i>
	Autori:	<i>Iacoletti Alessandro</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Visualizzare le informazioni di una determinata attività, da parte del cliente, e per il management delle attività extra da parte del gestore.</i>
	Attori Primario:	<i>Cliente, Gestore</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vogliono visualizzare informazioni relative alle attività extra dell'albergo</i>
	Condizioni: Pre-condizioni	<i>Ci deve essere almeno una attività extra</i>
	Post-condizioni di successo:	<i>L'utente riesce a leggere le informazioni riguardanti le attività</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>In caso di fallimento nessuna attività verrà visualizzata</i>
	Priorità	<i>Bassa</i>
	Casi d'uso inclusi:	<i>Nessuno</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Attività</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Cliente/Gestore</i>	<i>Seleziona la funzione visualizza attività.</i>
	2	<i>Sistema</i>	<i>Visualizzazione delle attività.</i>
	3	<i>Cliente/Gestore</i>	<i>Seleziona una attività per avere maggiori informazioni</i>
	4	<i>Sistema</i>	<i>Visualizza la form dell'attività selezionata</i>
	Scenario alternativo: Nessuna attività da visualizzare		
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa l'utente che non è presente nessuna attività</i>
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
Iterazioni	Autore		
	Autore	Data	Descrizione

	<i>Iacoletti Alessandro</i>	04/04/2008	<i>Prima versione del caso d'uso</i>
	<i>Iacoletti Alessandro</i>	07/04/2008	<i>Revisione dei casi d'uso</i>

### **Modulo <Valida>**

Vedi modello use-case relativo al ristorante.

## Modulo <Inserimento Attività>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>INS_ATT</i>
	Use Cases Nome:	<i>Inserimento Attività</i>
	Autori:	<i>Vicidomini Vincenzo</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere ai gestori dell'albergo di inserire nuove attività extra</i>
	Attori Primario:	<i>Gestore</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vogliono compiere operazioni di inserimento delle attività extra dell'albergo</i>
	Condizioni: Pre-condizioni	<i>L'utente deve aver immesso log-in e password valide</i>
	Post-condizioni di successo:	<i>L'utente riesce ad inserire le attività desiderate . I dati vengono salvati</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema avvisa il cliente che l'inserimento non è avvenuto con successo. Nessuna modifica è apportata al sistema</i>
	Priorità	<i>Media</i>
	Casi d'uso inclusi:	<i>Valida</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Attività</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Gestore</i>	<i>Chiede di poter inserire nuove attività nel sistema</i>
	2	<i>Sistema</i>	<i>Visualizza la form di inserimento</i>
	3	<i>Gestore</i>	<i>Compila la form in tutti i suoi campi</i>
	4	<i>Gestore</i>	<i>Conferma l'inserimento</i>
1° Scenario alternativo: Inserimento non confermato	5	<i>Sistema</i>	<i>Informa l'utente dell'avvenuto inserimento e memorizza la nuova attività nel menu di visualizzazione</i>
	3.1	<i>Gestore</i>	<i>Decide di annullare l'inserimento</i>
	3.2	<i>Sistema</i>	<i>Ritorna al menu principale</i>
	2° Scenario alternativo: Errore del Sistema		

	4.1	<i>Sistema</i>	<i>Informa l'utente che l'inserimento non è avvenuto per un errore del sistema</i>
	4.2	<i>Sistema</i>	<i>Ritorna al passo 3</i>
<b>Iterazioni</b>			
	Autore	Data	Descrizione
	<i>Vicidomini Vincenzo</i>	<i>04/04/2008</i>	<i>Prima versione del caso d'uso</i>
	<i>Vicidomini Vincenzo</i>	<i>07/04/2008</i>	<i>Revisione dei casi d'uso</i>

## Modulo <Modifica Attività>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>MOD_ATT</i>
	Use Cases Nome:	<i>Modifica Attività</i>
	Autori:	<i>Iacoletti Alessandro</i>
	Data:	<i>04/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere ai gestori dell'albergo di modificare le attività extra</i>
	Attori Primario:	<i>Gestore</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vogliono compiere operazioni di modifica delle attività extra dell'albergo</i>
	Condizioni: Pre-condizioni	<i>L'utente deve aver immesso log-in e password valide Deve essere presente almeno un'attività</i>
	Post-condizioni di successo:	<i>L'utente riesce a modificare le attività desiderate. I dati vengono memorizzati</i>
	Post-condizioni per Fallimento:	<i>Il sistema avvisa il cliente che la modifica non è avvenuta con successo. Nessun dato viene salvato dal sistema</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta</i>
	Casi d'uso inclusi:	<i>Valida Visualizza Attività</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione Attività</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Sistema</i>	<i>Include &lt; Visualizza Attività &gt;</i>
	2	<i>Gestore</i>	<i>Modifica i campi che intende modificare</i>
	3	<i>Gestore</i>	<i>Conferma la modifica</i>
	4	<i>Sistema</i>	<i>Informa l'utente dell'avvenuto modifica e aggiorna i cambiamenti</i>
1° Scenario alternativo: Modifica non confermata			
	Passo	Soggetto	Descrizione
	3.1	<i>Gestore</i>	<i>Decide di annullare la modifica</i>
	3.2	<i>Sistema</i>	<i>Ritorna al menu principale</i>
2° Scenario alternativo: Errore del Sistema			
	4.1	<i>Sistema</i>	<i>Informa l'utente che l'inserimento non è avvenuto per un errore del sistema</i>
	4.2	<i>Sistema</i>	<i>Ritorna al passo 3</i>
Iterazioni			

	Autore	Data	Descrizione
	<i>Iacoletti Alessandro</i>	04/04/2008	<i>Prima versione del caso d'uso</i>
	<i>Iacoletti Alessandro</i>	07/04/2008	<i>Revisione dei casi d'uso</i>

## Modulo <Cancellazione Attività>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	CANC_ATT
	Use Cases Nome:	Cancellazione Attività
	Autori:	Vicidomini Vincenzo
	Data:	04/04/2008
	Versione:	1.0
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere ai gestori dell'albergo di cancellare le attività extra</i>
	Attori Primario:	Gestore
	Altri:	Nessuno
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vogliono compiere operazioni di cancellazione delle attività extra dell'albergo</i>
	Condizioni: Pre-condizioni	<i>L'utente deve aver immesso login e password valide Deve essere presente almeno un'attività</i>
	Post-condizioni di successo:	<i>L'utente riesce a cancellare le attività desiderate. I dati vengono salvati</i>
	Post-condizioni per Fallimento:	<i>Il sistema avvisa il cliente che la cancellazione non è avvenuta con successo. Nessun dato viene salvato</i>
<b>Altre informazioni (opzionali):</b>	Priorità	Alta
	Casi d'uso inclusi:	Valida Visualizza Attività
	Estende il caso d'uso:	Nessuno
	Specializza il caso d'uso:	Nessuno
	Requisiti aggiuntivi	Nessuno
	Problemi vari	Nessuno

Descrizione degli scenari	Scenario principale		
	Passo	Soggetto	Descrizione
	1	Sistema	<i>Include &lt;Visualizza Attività&gt;</i>
	2	Gestore	<i>Sceglie l'attività che desidera cancellare</i>
	3	Sistema	<i>Chiede all'utente se veramente desidera cancellare tale attività</i>
	4	Gestore	<i>Conferma la cancellazione</i>
	5	Sistema	<i>Informa l'utente dell'avvenuta cancellazione e elimina l'attività</i>
1° Scenario alternativo: Cancellazione non confermata			
Passo	Soggetto	Descrizione	
4.1	Gestore	<i>Decide di annullare la cancellazione</i>	
4.2	Sistema	<i>Ritorna al menu principale</i>	
2° Scenario alternativo: Cancellazione non confermata			

	5.1	Sistema	<i>Informa l'utente che la cancellazione non è avvenuto per un errore del sistema</i>
	5.2	Sistema	<i>Ritorna al passo 4</i>
Annotazioni		Descrizione	
5.1		<i>Dopo la cancellazione i clienti prenotati all'attività dovranno essere informati</i>	
Iterazioni			
Autore		Data	Descrizione
<i>Vicidomini Vincenzo</i>		<i>04/04/2008</i>	<i>Prima versione del caso d'uso</i>
<i>Vicidomini Vincenzo</i>		<i>07/04/2008</i>	<i>Revisione dei casi d'uso</i>

### 3.5.2.2 Modello Casi d'uso Servizio Bar

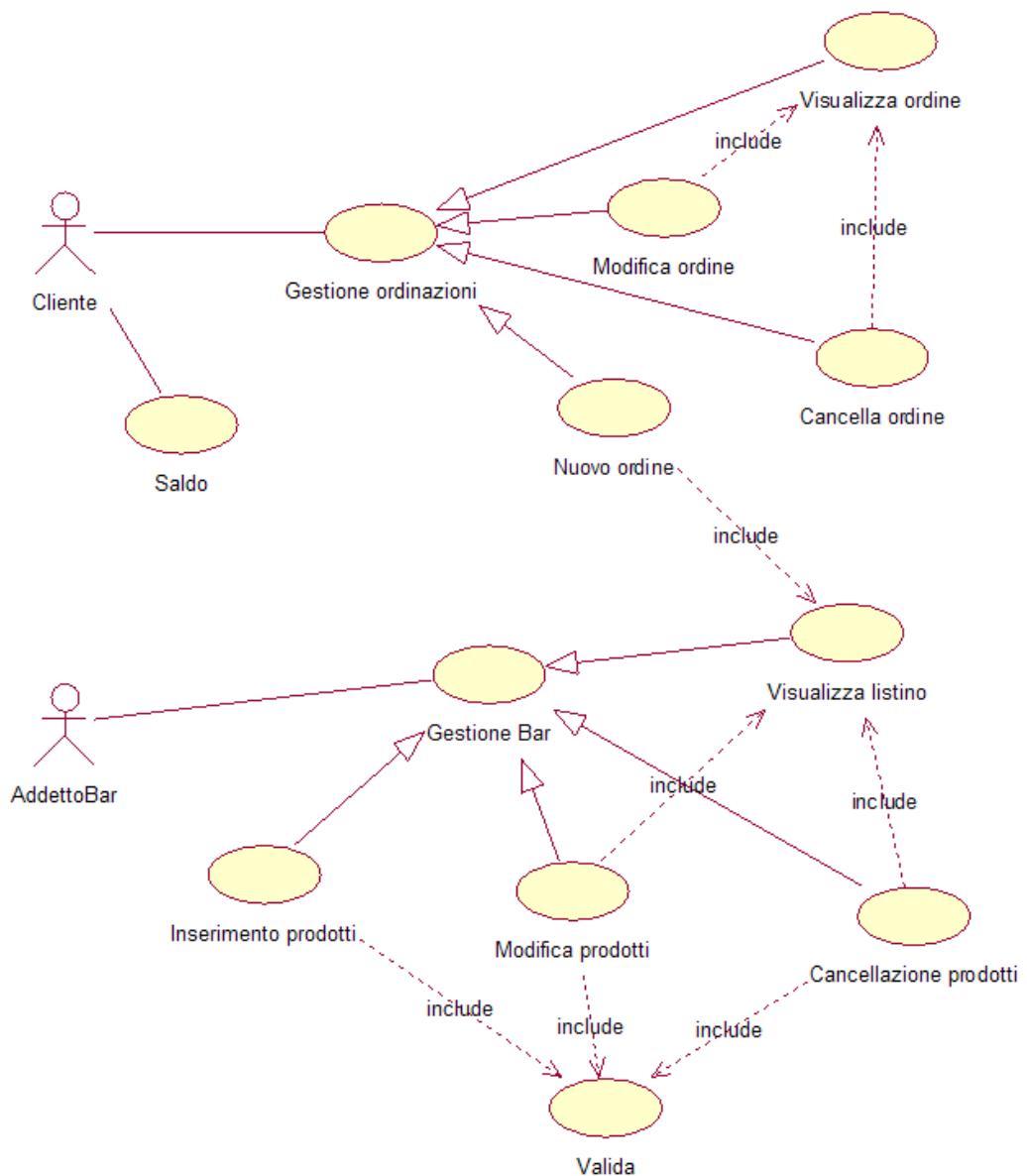


Figura 2: Casi d'uso Servizio Bar

## Attori

Attore	Versione	Autore
<i>Cliente</i>	1.1	<i>Pizza Ernesto</i>
Specializza l'attore:		
<i>Nessuna</i>		
Caso d'uso	Primario	Frequenza
1. Gestione Ordinazione	X	Alta
1. Visualizzazione Ordinazione	X	Alta
2. Nuova Ordinazione	X	Alta
3. Modifica Ordinazione	X	Bassa
4. Disdetta Ordinazione	X	Bassa
5. Visualizzazione Saldo	X	Media

Attore	Versione	Autore
<i>Addetto Bar</i>	1.0	<i>Mercurio Antonio</i>
Specializza l'attore:		
<i>Nessuna</i>		
Caso d'uso	Primario	Frequenza
1. Visualizza listino	X	Alta
2. Inserimento prodotto	X	Media
3. Modifica prodotto	X	Bassa
4. Elimina prodotto	X	Bassa
5. Valida	X	Media

## Organizzazione del Modello

### Gestione ordinazione

1. Visualizza ordinazione
2. Nuova ordinazione
3. Modifica ordinazione
4. Disdetta ordinazione
5. Visualizza saldo

### Gestione Bar

1. Visualizza listino
2. Inserimento prodotto
3. Modifica prodotto

4. Elimina prodotto

5. Valida

### **Modulo <Gestione Ordinazione>**

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	GES_ORD	
	Use Cases Nome:	Gestione Ordinazione	
	Autori:	Pizza Ernesto	
	Data:	07 / 04 /2008	
	Versione:	1.0	
<b>Informazioni principali:</b>	Obiettivi:	<i>E' una sorta di generalizzazione delle funzionalità richieste per la gestione delle ordinazioni; le operazioni fondamentali sono quelle di Nuova Ordinazione, Modifica Ordinazione, Disdetta Ordinazione e Visualizzazione Ordinazione.</i>	
	Attori Primario:	<i>Cliente.</i>	
	Altri:	<i>Nessuno.</i>	
	Azione d'avvio:	<i>Il caso d'uso è innescato dalla necessità di usare le funzioni di Nuova Ordinazione, Modifica di una Ordinazione, Disdetta di una Ordinazione e Visualizzazione Ordinazione.</i>	
	Condizioni: Pre-condizioni	<i>Si deve essere clienti della struttura.</i>	
	Post-condizioni di successo:	<i>Si riescono ad utilizzare le funzioni che specializzano questo caso d'uso.</i>	
	Post-condizioni per Fallimento:	<i>Non si riescono ad utilizzare le funzioni che specializzano questo caso d'uso.</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta.</i>	
	Casi d'uso inclusi:	<i>Nessuno.</i>	
	Estende il caso d'uso:	<i>Nessuno.</i>	
	Specializza il caso d'uso:	<i>Nessuno.</i>	
	Requisiti aggiuntivi	<i>Nessuno.</i>	
	Problemi vari	<i>Nessuno.</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1		(Vedi casi d'uso che lo specializzano)
	Autore	Data	Descrizione
	<i>Pizza Ernesto</i>	<i>07/04/2008</i>	<i>Prima versione del caso d'uso.</i>
	<i>Pizza Ernesto</i>	<i>07/04/2008</i>	<i>Revisione del caso d'uso.</i>

## Modulo <Visualizza Ordinazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>VIS_ORD</i>
	Use Cases Nome:	<i>Visualizzazione Ordinazione</i>
	Autori:	<i>Pizza Ernesto</i>
	Data:	<i>07 / 04 /2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permette al cliente la visualizzazione delle ordinazioni fatte al Bar, con tutte le informazioni relativi ai prodotti ordinati, alle quantità, ecc. .</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno.</i>
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole visualizzare l'elenco delle ordinazioni fatte al bar.</i>
	Condizioni: Pre-condizioni	<i>Si deve essere clienti della struttura.</i>
	Post-condizioni di successo:	<i>Vengono visualizzate le ordinazioni fatte al bar.</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Viene visualizzato un messaggio che informa il cliente che non ha effettuato ordinazioni al bar.</i>
	Priorità	<i>Bassa.</i>
	Casi d'uso inclusi:	<i>Nessuno.</i>
	Estende il caso d'uso:	<i>Nessuno.</i>
	Specializza il caso d'uso:	<i>Gestione Ordinazione.</i>
<b>Descrizione degli scenari</b>	Requisiti aggiuntivi	<i>Nessuno.</i>
	Problemi vari	<i>Nessuno.</i>
	Scenario principale*	
	Passo	Soggetto
	1	<i>Cliente</i>
		<i>Seleziona la funzione di Visualizzazione Ordinazione.</i>
	2	<i>Sistema</i>
		<i>Visualizza le ordinazioni fatte al bar.</i>
	3	<i>Cliente</i>
		<i>Seleziona una ordinazione per avere tutte le informazioni relative a quella</i>
	4	<i>Sistema</i>
		<i>Visualizza i dettagli dell'ordinazione selezionata.</i>
	Scenario alternativo*	
	Passo	Soggetto
	2.1	<i>Sistema</i>
		<i>Informa il cliente che non ha effettuato ordinazioni al bar.</i>
	2.2	<i>Sistema</i>
		<i>Termina il caso d'uso</i>
	Autore	Data
	<i>Pizza Ernesto</i>	<i>07/04/2008</i>
		<i>Prima versione del caso d'uso.</i>
	<i>Pizza Ernesto</i>	<i>07/04/2008</i>
		<i>Revisione del caso d'uso.</i>

## Modulo <Nuova Ordinazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>NUO_ORD</i>
	Use Cases Nome:	<i>Nuova Ordinazione</i>
	Autori:	<i>Pizza Ernesto</i>
	Data:	<i>07 / 04 /2008</i>
	Versione:	<i>1.1</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Il cliente effettua la scelta di uno o più prodotti, tra quelli disponibili, confermando infine il tutto con l'ordinazione.</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno.</i>
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole ordinare uno o più prodotti.</i>
	Condizioni: Pre-condizioni	<i>Si deve ordinare almeno un prodotto.</i>
	Post-condizioni di successo:	<i>I dati vengono salvati. Viene visualizzato un messaggio al cliente che lo informa dell'avvenuta ordinazione. Viene avviata l'erogazione del servizio.</i>
	Post-condizioni per Fallimento:	<i>Viene visualizzato un messaggio che informa il cliente che non è stata effettuata l'ordinazione. I dati non vengono salvati.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Media.</i>
	Casi d'uso inclusi:	<i>Visualizzazione Listino.</i>
	Estende il caso d'uso:	<i>Nessuno.</i>
	Specializza il caso d'uso:	<i>Gestione Ordinazione.</i>
	Requisiti aggiuntivi	<i>Nessuno.</i>
	Problemi vari	<i>Nessuno.</i>
<b>Descrizione degli scenari</b>	Scenario principale*	
	Passo	Soggetto
	1	<i>Cliente</i>
	2	<i>Sistema</i>
	3	<i>Cliente</i>
	4	<i>Cliente</i>
	5	<i>Sistema</i>
	6	<i>Sistema</i>
	Scenario alternativo: Prodotto non disponibile	
	Passo	Soggetto
	5.1	<i>Sistema</i>
	5.2	<i>Sistema</i>
	Annotazione	Descrizione
		Descrizione

	6	<p>Una volta che l'ordine è stato effettuato con successo devono passare un TOT_MINUTI affinché la postazione Bar riceva tale ordine. Entro questi TOT_MINUTI il cliente può modificare o disdire l'ordine.</p> <p>Una volta passato questo lasso di tempo l'ordinazione viene stampata nella postazione Bar e viene consegnata.</p> <p>Nel caso in cui l'addetto bar si accorge che un prodotto è esaurito provvede a modificare lo stato del prodotto e comunica al cliente l'indisponibilità di quest'ultimo.</p>
	5.1	<p>Questo scenario è previsto nel caso in cui due o più clienti vogliono ordinare lo stesso prodotto contemporaneamente ma la disponibilità non può soddisfare tutte le ordinazioni. Il sistema da la precedenza a chi ha inviato prima l'ordinazione e di conseguenza ai clienti cui non sarà possibile soddisfare la richiesta apparirà un messaggio che li avvertirà che il prodotto non è più disponibile.</p>
<b>Autore</b>		
<i>Pizza Ernesto</i>	07/04/2008	<i>Prima modifica alla prima versione del caso d'uso.</i>
<i>Pizza Ernesto</i>	07/04/2008	<i>Revisione del caso d'uso.</i>

## Modulo <Modifica Ordinazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>MOD_ORD</i>
	Use Cases Nome:	<i>Modifica Ordinazione</i>
	Autori:	<i>Pizza Ernesto</i>
	Data:	<i>07 / 04 /2008</i>
	Versione:	<i>1.1</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Nel caso che il cliente, per qualche ripensamento, voglia apportare delle modifiche ad una Ordinazione già avvenuta.</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno.</i>
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole modificare una ordinazione già fatta.</i>
	Condizioni: Pre-condizioni	<i>Si deve aver ordinato almeno un prodotto.</i>
	Post-condizioni di successo:	<i>I dati vengono salvati. Viene visualizzato un messaggio al cliente di avvenuta modifica dell'ordinazione Viene avviata l'erogazione del servizio</i>
	Post-condizioni per Fallimento:	<i>Viene visualizzato un messaggio che informa il cliente che non è stata effettuata l'ordinazione. I dati non vengono salvati.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta.</i>
	Casi d'uso inclusi:	<i>Visualizzazione Ordinazione.</i>
	Estende il caso d'uso:	<i>Nessuno.</i>
	Specializza il caso d'uso:	<i>Gestione Ordinazione.</i>
	Requisiti aggiuntivi	<i>Nessuno.</i>
	Problemi vari	<i>Nessuno.</i>
<b>Descrizione degli scenari</b>	Scenario principale*	
	Passo	Soggetto
	1	<i>Cliente</i>
	2	<i>Sistema</i>
	3	<i>Cliente</i>
	4	<i>Sistema</i>
	5	<i>Sistema</i>
	6	<i>Sistema</i>
	7	<i>Cliente</i>
	8	<i>Cliente</i>
	9	<i>Sistema</i>
	10	<i>Sistema</i>
	Scenario alternativo 1 (Nessuna Ordinazione)	
	Passo	Soggetto
	2.1	<i>Sistema</i>

	<b>2.2</b>	<i>Sistema</i>	<i>Termina il caso d'uso.</i>
<b>Scenario alternativo 2 (Tempo Modifica Scaduto)</b>			
	<b>Passo</b>	<b>Soggetto</b>	<b>Descrizione</b>
	4.1	<i>Sistema</i>	<i>Informa il cliente che non è più possibile effettuare la modifica, in quanto è scaduto il tempo a disposizione per tale</i>
	4.2	<i>Sistema</i>	<i>Ritorna al punto 2.</i>
<b>Scenario alternativo 3 (Errore del Sistema)</b>			
	<b>Passo</b>	<b>Soggetto</b>	<b>Descrizione</b>
	10.1	<i>Sistema</i>	<i>Informa il cliente che la modifica non è stata effettuata per un errore nel</i>
	10.2	<i>Sistema</i>	<i>Ritorna al punto 7.</i>
<b>Scenario alternativo 4 (Prodotto non disponibile)</b>			
	<b>Passo</b>	<b>Soggetto</b>	<b>Descrizione</b>
	9.1	<i>Sistema</i>	<i>Visualizza un messaggio che non è possibile modificare l'ordinazione per disponibilità non sufficiente.</i>
	9.2	<i>Sistema</i>	<i>Ritorna al punto 2.</i>
	<b>Autore</b>	<b>Data</b>	<b>Descrizione</b>
	<i>Pizza Ernesto</i>	<i>07/04/2008</i>	<i>Prima modifica alla prima versione del caso d'uso.</i>
	<i>Pizza Ernesto</i>	<i>07/04/2008</i>	<i>Revisione del caso d'uso.</i>

## Modulo <Cancella Ordinazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	CAN_ORD
	Use Cases Nome:	Cancella Ordinazione
	Autori:	Pizza Ernesto
	Data:	07 / 04 /2008
	Versione:	1.1
<b>Informazioni principali:</b>	Obiettivi:	<i>Nel caso che il cliente, per qualche ripensamento, voglia annullare un ordine precedentemente inviato.</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno.</i>
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole disdire una ordinazione già fatta.</i>
	Condizioni: Pre-condizioni	<i>Si deve aver ordinato almeno un prodotto.</i>
	Post-condizioni di successo:	<i>I dati vengono salvati. Viene visualizzato un messaggio al cliente di avvenuta disdetta dell'ordinazione.</i>
	Post-condizioni per Fallimento:	<i>Viene visualizzato un messaggio che informa il cliente che non è stata effettuata la disdetta dell'ordinazione. I dati non vengono salvati.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta.</i>
	Casi d'uso inclusi:	<i>Visualizzazione Ordinazione.</i>
	Estende il caso d'uso:	<i>Nessuno.</i>
	Specializza il caso d'uso:	<i>Gestione Ordinazione.</i>
	Requisiti aggiuntivi	<i>Nessuno.</i>
<b>Descrizione degli scenari</b>	Scenari principale*	
	Passo	Soggetto
	1	<i>Cliente</i>
		<i>Selezione la funzione Cancella Ordinazione.</i>
	2	<i>Sistema</i>
		<i>Include (Visualizzazione Ordinazione).</i>
	3	<i>Cliente</i>
		<i>Sceglie l'ordinazione da disdire.</i>
	4	<i>Sistema</i>
		<i>Controlla se è ancora possibile disdire l'ordinazione selezionata.</i>
	5	<i>Sistema</i>
		<i>Mostra al cliente l'ordinazione</i>
	6	<i>Cliente</i>
		<i>Disdice l'ordinazione.</i>
	7	<i>Cliente</i>
		<i>Conferma la disdetta.</i>
	8	<i>Sistema</i>
		<i>Salva i dati.</i>
	9	<i>Sistema</i>
		<i>Informa il cliente dell'avvenuta disdetta.</i>
	Scenario alternativo 1 (Nessuna Ordinazione)	
	Passo	Soggetto
	2.1	<i>Sistema</i>
		<i>Informa il cliente che non è stata effettuata alcuna ordinazione.</i>
	2.2	<i>Sistema</i>
	Scenario alternativo 2 (Tempo Disdetta Scaduto)	
	Passo	Soggetto
		Descrizione

	5.1	<i>Sistema</i>	<i>Informa il cliente che non è più possibile effettuare la disdetta, in quanto è scaduto il tempo a disposizione per tale</i>
	5.2	<i>Sistema</i>	<i>Ritorna al punto 2.</i>
<b>Scenario alternativo 3 (Errore del Sistema)</b>			
	Passo	Soggetto	Descrizione
	9.1	<i>Sistema</i>	<i>Informa il cliente che la disdetta non è stata effettuata per un errore nel</i>
	9.2	<i>Sistema</i>	<i>Ritorna al punto 7.</i>
	Autore	Data	Descrizione
	Pizza Ernesto	07/04/2008	<i>Prima modifica alla prima versione del caso d'uso.</i>
	Pizza Ernesto	07/04/2008	<i>Revisione del caso d'uso.</i>

## Modulo <Visualizzazione Saldo>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	VIS_SAL
	Use Cases Nome:	Visualizzazione Saldo
	Autori:	Pizza Ernesto
	Data:	06 / 04 /2008
	Versione:	1.0
<b>Informazioni principali:</b>	Obiettivi:	<i>Il cliente può visualizzare il suo attuale saldo presso la struttura, in questo modo è comodamente e costantemente informato delle spese effettuate.</i>
	Attori Primario:	<i>Cliente.</i>
	Altri:	<i>Nessuno.</i>
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole visualizzare il saldo presso la struttura.</i>
	Condizioni: Pre-condizioni	<i>Si deve essere clienti della struttura.</i>
	Post-condizioni di successo:	<i>L'utente riesce a visualizzare il proprio saldo.</i>
	Post-condizioni per Fallimento:	<i>Il sistema informa il cliente che non è momentaneamente possibile visualizzare il saldo.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Bassa.</i>
	Casi d'uso inclusi:	<i>Nessuno.</i>
	Estende il caso d'uso:	<i>Nessuno.</i>
	Specializza il caso d'uso:	<i>Nessuno.</i>
	Requisiti aggiuntivi	<i>Nessuno.</i>
	Problemi vari	<i>Nessuno.</i>
<b>Descrizione degli scenari</b>	Scenario principale*	
	Passo	Soggetto
	1	<i>Cliente</i>
		<i>Accede alla funzione di Visualizzazione Saldo.</i>
	2	<i>Sistema</i>
		<i>Accede al conto del cliente.</i>
	3	<i>Sistema</i>
		<i>Visualizza il saldo del cliente.</i>
	Autore	Data
	<i>Pizza Ernesto</i>	06/04/2008
		<i>Prima versione del caso d'uso.</i>
	<i>Pizza Ernesto</i>	06/04/2008
		<i>Revisione del caso d'uso.</i>

## Modulo <Gestione Bar>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	GES_BAR	
	Use Cases Nome:	Gestione Bar	
	Autori:	Mercurio Antonio	
	Data:	07 / 04 /2008	
	Versione:	1.0	
<b>Informazioni principali:</b>	Obiettivi:	<i>E' una sorta di generalizzazione delle funzionalità richieste per la gestione dei prodotti del bar; le operazioni fondamentali sono quelle di Inserimento prodotto, Modifica prodotto, Eliminazione prodotto e Visualizzazione prodotti.</i>	
	Attori Primario:	<i>Addetto bar.</i>	
	Altri:	<i>Nessuno.</i>	
	Azione d'avvio:	<i>Il caso d'uso è innescato dalla necessità di usare le funzioni di Inserimento, Modifica, Cancellazione e Visualizzazione dei prodotti del bar.</i>	
	Condizioni: Pre-condizioni	<i>L'utente deve essere loggato al sistema.</i>	
	Post-condizioni di successo:	<i>Si riescono ad utilizzare le funzioni che specializzano questo caso d'uso.</i>	
	Post-condizioni per Fallimento:	<i>Non si riescono ad utilizzare le funzioni che specializzano questo caso d'uso.</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta.</i>	
	Casi d'uso inclusi:	<i>Nessuno.</i>	
	Estende il caso d'uso:	<i>Nessuno.</i>	
	Specializza il caso d'uso:	<i>Nessuno.</i>	
	Requisiti aggiuntivi	<i>Nessuno.</i>	
	Problemi vari	<i>Nessuno.</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1		(Vedi casi d'uso che lo specializzano)
	Autore	Data	Descrizione
	Mercurio Antonio	07/04/2008	<i>Prima versione del caso d'uso.</i>
	Mercurio Antonio	07/04/2008	<i>Revisione del caso d'uso.</i>

## Modulo <Visualizza Listino>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	VIS_LIS
	Use Cases Nome:	<i>Visualizza listino</i>
	Autori:	<i>Mercurio Antonio</i>
	Data:	06/04/2008
	Versione:	1.0
<b>Informazioni principali:</b>	Obiettivi:	<i>Visualizzare il listino</i>
	Attori Primario:	<i>Addetto Bar, Cliente</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è avviato quando un utente vuole visualizzare il listino dei prodotti</i>
	Condizioni: Pre-condizioni	<i>Deve esserci un listino da visualizzare</i>
	Post-condizioni di successo:	<i>L'utente visualizza il listino</i>
	Post-condizioni per Fallimento:	<i>Il sistema avverte che non è possibile visualizzare il listino</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Bassa</i>
	Casi d'uso inclusi:	<i>Nessuno</i>
	Estende il caso d'uso:	<i>Nessuno</i>
	Specializza il caso d'uso:	<i>Gestione bar</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>

Descrizione degli scenari	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	<i>Addetto bar, cliente</i>	<i>L'utente richiede la visualizzazione del listino</i>
	2	<i>Sistema</i>	<i>Il sistema mostra il listino</i>
	Scenario alternativo Listino non disponibile		
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa l'utente che il listino non è disponibile</i>
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
	Annotazioni		
	Descrizione		
	2	<i>Il listino deve essere disponibile</i>	
	3	<i>Nel listino vengono visualizzati solamente i prodotti con disponibilità maggiore di 0.</i>	
	Iterazioni		
	Autore	Data	Descrizione
	<i>Mercurio Antonio</i>	06/04/2008	<i>Prima versione del caso d'uso</i>

	<i>Mercurio Antonio</i>	<i>07/04/2008</i>	<i>Revisione del caso d'uso</i>
--	-----------------------------	-------------------	---------------------------------

### **Modulo <Inserimento Prodotto>**

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>/INS_PRO</i>	
	Use Cases Nome:	<i>Inserimento prodotto</i>	
	Autori:	<i>Mercurio Antonio</i>	
	Data:	<i>06/04/2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Inserire un nuovo prodotto nel listino</i>	
	Attori	Primario: <i>Addetto Bar</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il caso d'uso è avviato quando l'addetto al bar deve inserire un nuovo prodotto</i>	
	Condizioni: Pre-condizioni	<i>L'addetto al bar deve essere loggato al sistema</i>	
	Post-condizioni di successo:	<i>Il prodotto viene inserito</i>	
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il prodotto non viene inserito</i>	
	Priorità	<i>Alta</i>	
	Casi d'uso inclusi:	<i>VAL</i>	
	Estende il caso d'uso:	<i>Nessuno</i>	
	Specializza il caso d'uso:	<i>Gestione Bar</i>	
<b>Descrizione degli scenari</b>	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
	<b>Scenario principale*</b>		
	Passo	Soggetto	Descrizione
	1	<i>Addetto Bar</i>	<i>Richiede l'inserimento di un nuovo prodotto</i>
	2	<i>Sistema</i>	<i>Mostra l'interfaccia per inserire il nuovo prodotto</i>
	3	<i>Addetto Bar</i>	<i>Inserisce i dati relativi al prodotto da inserire</i>
	4	<i>Sistema</i>	<i>Controlla l'esattezza dei dati</i>
	5	<i>Sistema</i>	<i>Controlla che il prodotto non sia già presente</i>
	6	<i>Sistema</i>	<i>Aggiunge il nuovo prodotto</i>
	<b>Scenario alternativo Dati non corretti</b>		
	Passo	Soggetto	Descrizione
	4.1	<i>Sistema</i>	<i>Mostra un messaggio di errore</i>

	4.2	<i>Sistema</i>	<i>Ritorna al punto 2</i>		
<b>Scenario alternativo Prodotto già presente</b>					
	Passo	<i>Soggetto</i>	<i>Descrizione</i>		
	5.1	<i>Sistema</i>			
	5.2	<i>Sistema</i>			
Annotazioni					
Descrizione					
	2.1	<i>Tutti i campi della maschera sono obbligatori</i>			
Iterazioni					
	Autore	Data	Descrizione		
	<i>Mercurio Antonio</i>	<i>06/04/2008</i>	<i>Prima Versione del caso d'uso</i>		
	<i>Mercurio Antonio</i>	<i>07/04/2008</i>	<i>Revisione del caso d'uso</i>		

## Modulo <Modifica Prodotto>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	MOD_PRO	
	Use Cases Nome:	Modifica prodotti	
	Autori:	Mercurio Antonio	
	Data:	06/04/2008	
	Versione:	1.0	
<b>Informazioni principali:</b>	Obiettivi:	Modifica di un prodotto del listino	
	Attori		
	Primario:	Addetto bar	
	Altri:	Nessuno	
	Azione d'avvio:	Il caso d'uso è avviato quando l'addetto bar deve modificare un prodotto	
	Condizioni:	L'addetto al bar deve essere loggato al sistema Il listino non deve essere vuoto	
	Pre-condizioni		
	Post-condizioni di successo:	Il prodotto viene modificato	
	Post-condizioni per Fallimento:	Il prodotto non viene modificato	
	Priorità	Alta	
	Casi d'uso inclusi:	VISL/S, VAL	
<b>Altre informazioni (opzionali):</b>	Estende il caso d'uso:	Nessuno	
	Specializza il caso d'uso:	Gestione Bar	
	Requisiti aggiuntivi	Nessuno	
	Problemi vari	Nessuno	
	Scenari principale*		
<b>Descrizione degli scenari</b>	Passo	Soggetto	Descrizione
	1	Addetto bar	Richiede modifica di un prodotto
	2	Sistema	Include (Visualizza listino)
	3	Addetto bar	Seleziona prodotto da modificare
	4	Sistema	Mostra maschera per la modifica del prodotto
	5	Addetto bar	Modifica i dati della maschera
	6	Sistema	Controlla la correttezza dei dati
	7	Sistema	Applica modifiche
	Scenari alternativi* Dati errati		
	Passo	Soggetto	Descrizione
	7.1	Sistema	Informa l'utente che non è possibile applicare le modifiche
	7.2	Sistema	Ritorna al punto 2
	Annotazioni		
		Descrizione	
	3.1	Tutti i campi devono essere compilati correttamente	
	Iterazioni		
	Autore	Data	Descrizione

*Mercurio Antonio* | *06/04/2008* | *Prima versione del caso d'uso*

## **Modulo <Elimina Prodotto>**

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>ELI_PRO</i>	
	Use Cases Nome:	<i>Elimina prodotto</i>	
	Autori:	<i>Mercurio Antonio</i>	
	Data:	<i>06/04/2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere la modifica di un prodotto</i>	
	Attori Primario:	<i>Addetto Bar</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il caso d'uso è avviato quando l'addetto bar deve eliminare un prodotto dal listino</i>	
	Condizioni: Pre-condizioni	<i>L'addetto al bar deve essere loggato al sistema Il listino non deve essere vuoto</i>	
	Post-condizioni di successo:	<i>Il prodotto viene eliminato</i>	
	Post-condizioni per Fallimento:	<i>Il prodotto non viene eliminato</i>	
	Priorità	<i>Alta</i>	
<b>Altre informazioni (opzionali):</b>	Casi d'uso inclusi:	<i>VISLIS, VAL</i>	
	Estende il caso d'uso:	<i>Nessuno</i>	
	Specializza il caso d'uso:	<i>Gestione bar</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
	Scenario principale*		
<b>Descrizione degli scenari</b>	Passo	Soggetto	Descrizione
	1	<i>Addetto bar</i>	<i>Richiede eliminazione prodotto</i>
	2	<i>Sistema</i>	<i>Include (Visualizza listino)</i>
	3	<i>Addetto bar</i>	<i>Seleziona un prodotto da eliminare</i>
	4	<i>Sistema</i>	<i>Chiede conferma eliminazione prodotto</i>
	5	<i>Addetto bar</i>	<i>L'addetto conferma l'eliminazione.</i>
	6	<i>Sistema</i>	<i>Elimina prodotto</i>
<b>Annotazioni</b>	Descrizione		
	3.1	<i>Deve essere selezionato almeno un prodotto</i>	
	Iterazioni		
Autore	Data	Descrizione	
<i>Mercurio Antonio</i>	<i>06/04/2008</i>	<i>Prima versione del caso d'uso</i>	
<i>Mercurio Antonio</i>	<i>07/04/2008</i>	<i>Revisione del caso d'uso</i>	

## **Modulo <Valida>**

Vedi modello use-case relativo al ristorante.

### 3.5.2.2 Modello Casi d'uso Servizio Ristorante

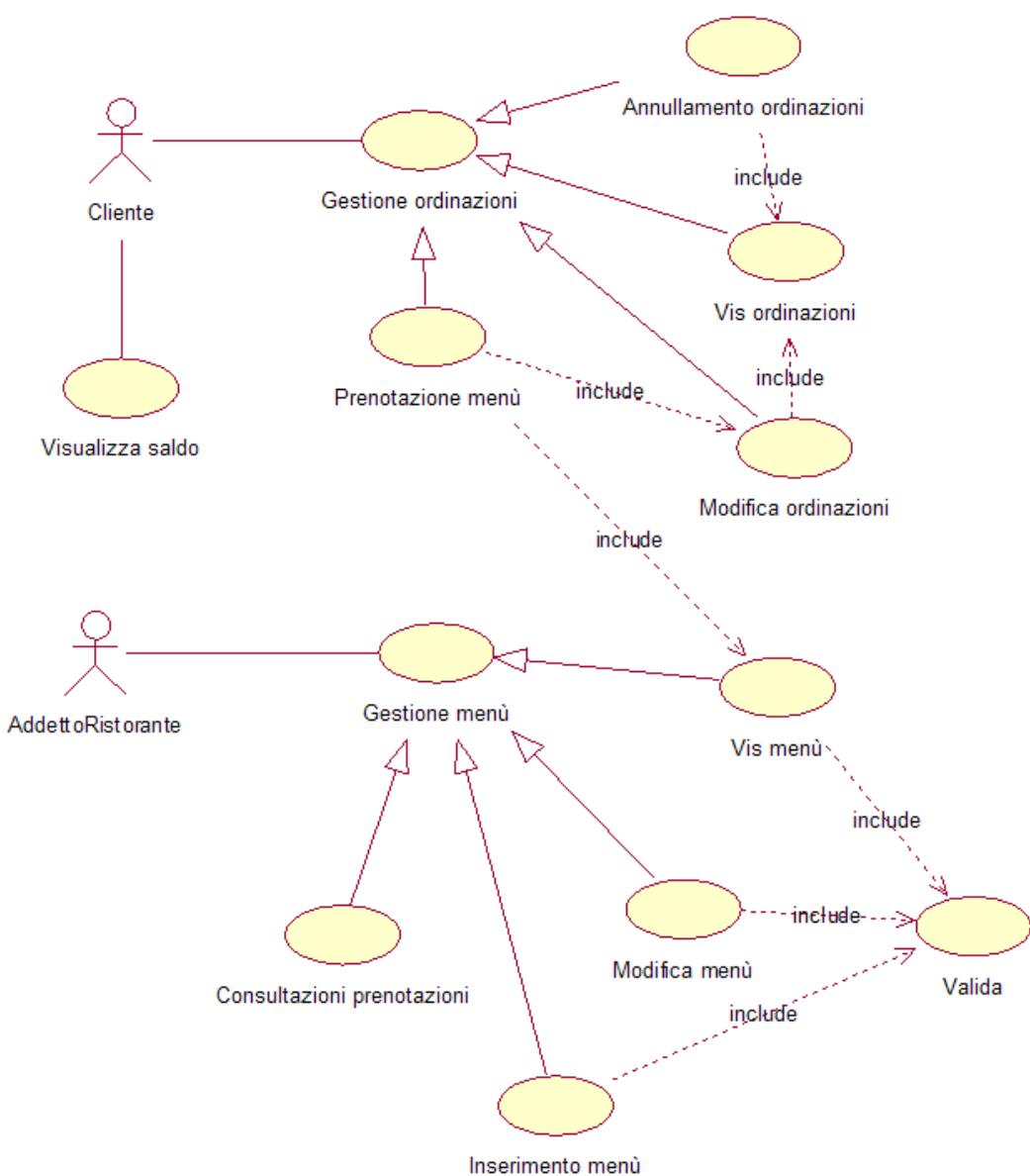


Figura 3: Casi d'uso Servizio Ristorante

## Attori

Attore	Versione	Autore
<i>Cliente</i>	1.0	<i>Pizza Ernesto</i>
Specializza l'attore:		
<i>Nessuno</i>		
Caso d'uso	Primario	Frequenza
1. Visualizza menù;	X	Alta Alta
2. Prenotazione menù;	X	Bassa
3. Modifica prenotazione;	X	Bassa
4. Annulla prenotazione;	X	Media
5. Visualizza saldo;	X	

Attore	Versione	Autore
<i>Gestore</i>	1.0	<i>Pizza Ernesto</i>
Specializza l'attore:		
<i>Nessuno</i>		
Caso d'uso	Primario	Frequenza
1. Valida;	X	Alta
2. Visualizzazione menù;	X	Alta
3. Inserimento menù;	X	Alta
4. Modifica menù;		
5. consultazione prenotazione;	X	Bassa
	X	Media

**Organizzazione del modello**

Gestione  
prenotazione

1. Visualizza menù;
2. Prenotazione menù;
3. Modifica prenotazione;
4. Annullamento prenotazione;
5. Visualizza prenotazione;
6. Visualizza saldo;

Gestione menù

1. Valida;
2. visualizza menù;
3. inserimento menù;
4. modifica menù;
5. Consultazione prenotazioni;

### **Modulo <Gestione Prenotazione>**

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>GES_PRE</i>
	Use Cases Nome:	<i>Gestione prenotazione</i>
	Autori:	<i>Pizza Ernesto</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>E' la generalizzazione delle funzionalità dei clienti per la gestione delle ordinazioni dalle proprie camere; le principali operazioni sono la prenotazione di un menù, la modifica di un menù e le varie visualizzazioni</i>
	Attori Primario:	<i>Cliente</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso viene innescato quando il cliente vuole effettuare delle ordinazioni dalla propria stanza da albergo</i>
	Condizioni: Pre-condizioni	<i>deve esistere almeno un cliente</i>

	Post-condizioni di successo:	<i>Il cliente riesce ad effettuare delle prenotazioni</i>	
	Post-condizioni per Fallimento:	<i>Il cliente non riesce ad effettuare alcun tipo di operazione</i>	
	Priorità	<i>Alta</i>	
<b>Altre informazioni (opzionali):</b>	Casi d'uso inclusi:	<i>nessuno</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>nessuno</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>(Vedi casi d'uso che specializzano questo)</i>
	Iterazioni		
	Autore	Data	Descrizione
	<i>Pizza Ernesto</i>	<i>7/04/2008</i>	<i>Prima versione caso d'uso</i>

## Modulo <Prenotazione Menu>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>PREN_MENU'</i>
	Use Cases Nome:	<i>Prenotazione menù</i>
	Autori:	<i>Iacoletti Alessandro</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>il sistema permette al cliente di effettuare delle prenotazioni</i>
	Attori Primario:	<i>cliente</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>il caso d'uso è innescato quando si vuole effettuare una o più prenotazioni</i>
	Condizioni: Pre-condizioni	<i>deve esistere almeno un menù</i>
	Post-condizioni di successo:	<i>L'utente riesce a prenotare i menù desiderati. I dati vengono salvati</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la prenotazione non è stata effettuata. Non viene salvato nessun dato</i>
	Priorità	<i>Media</i>
	Casi d'uso inclusi:	<i>Visualizza menù</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>gestione prenotazione</i>
<b>Descrizione degli scenari</b>	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>
	Scenario principale	
	Passo	Soggetto
	1	<i>Cliente</i>
	2	<i>Sistema</i>
	3	<i>Cliente</i>
	4	<i>Sistema</i>
	5	<i>Cliente</i>
	6	<i>Cliente</i>
	7	<i>Sistema</i>
	1 Scenario alternativo <DATA NON CORRETTA>	
	Passo	Soggetto
	3.1	<i>Sistema</i>
	3.2	<i>Sistema</i>
	2 Scenario alternativo <ERRORE COMPILAZIONE>	

	5.1	Sistema	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>
	5.2	Sistema	<i>Ritorna al punto 5</i>
	<b>3 Scenario alternativo &lt;TEMPO SCADUTO&gt;</b>		
	3.1	Sistema	<i>Informa l'utente che non è più possibile effettuare una prenotazione (tempo disponibile scaduto)</i>
	3.2	Sistema	<i>Ritorna al punto 2</i>
	<b>4 Scenario alternativo &lt;ERRORE SISTEMA&gt;</b>		
	6.1	Sistema	<i>Informa l'utente che la prenotazione non è stata effettuata per un errore di sistema</i>
	6.2	Sistema	<i>Ritorna al punto 2</i>
	Annotazioni		
	Desrizione		
	3.1	<i>L'utente deve compilare la form in tutte le sue parti, ed eventualmente specificare campi opzionali</i>	
	3.2	<i>L'utente deve specificare la quantità ( es: più persone nella stanza)</i>	
	Iterazioni		
	Autore	Data	Descrizione
	Iacoletti Alessandro	7/04/2008	<i>Prima versione caso d'uso</i>
	Pizza Ernesto	08/04/2008	<i>Revisione del caso d'uso</i>

## Modulo <Modifica Prenotazione>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>MOD_PREN</i>
	Use Cases Nome:	<i>Modifica prenotazione</i>
	Autori:	<i>Iacoletti Alessandro</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al cliente di modificare la prenotazione</i>
	Attori Primario: Altri:	<i>Cliente</i> <i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole effettuare la modifica della propria ordinazione</i>
	Condizioni: Pre-condizioni	<i>Ci deve essere almeno una prenotazione</i>
	Post-condizioni di successo:	<i>L'utente riesce a modificare la prenotazione. I dati vengono salvati</i>
	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la modifica non è stata effettuata. I dati non vengono salvati</i>
	Priorità	<i>Media</i>
<b>Altre informazioni (opzionali):</b>	Casi d'uso inclusi:	<i>visualizzazione prenotazione</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>Gestione prenotazione</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>
<b>Descrizione degli scenari</b>	Scenario principale	
	Passo	Soggetto
	1	<i>Cliente</i>
		<i>Selezione la funzione modifica prenotazione</i>
	2	<i>Sistema</i>
		<i>Include (Visualizzazione prenotazione)</i>
	3	<i>Cliente</i>
		<i>Sceglie la prenotazione da modificare</i>
	4	<i>Sistema</i>
		<i>Visualizza la form della prenotazione</i>
	5	<i>Cliente</i>
		<i>Modifica i campi desiderati</i>
	6	<i>Cliente</i>
		<i>Conferma la modifica</i>
	7	<i>Sistema</i>
		<i>Registra i dati e informa l'utente che la modifica è stata effettuata</i>
	1 Scenario alternativo <NESSUNA PRENOTAZIONE>	
	Passo	Soggetto
	2.1	<i>Sistema</i>
		<i>Informa l'utente che non è stata effettuata nessuna prenotazione in precedenza</i>

	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>		
<b>2° Scenario alternativo &lt;ERRATA COMPILAZIONE&gt;</b>					
	7.1	<i>Sistema</i>	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>		
	7.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>		
<b>3° Scenario alternativo &lt;PRENOTAZIONE SCADUTA&gt;</b>					
	4.1	<i>Sistema</i>	<i>Informa l'utente che non è più possibile modificare la prenotazione (tempo scaduto)</i>		
	4.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>		
<b>4° Scenario alternativo &lt;ERRORE SISTEMA&gt;</b>					
	6.1	<i>Sistema</i>	<i>Informa l'utente che la prenotazione non è stata modificata per un errore di sistema</i>		
	6.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>		
Annotazioni					
Desrizione					
	5	<i>L'utente deve compilare la form in tutte le sue parti, ed eventualmente specificare campi opzionali</i>			
Iterazioni					
Autore      Data      Descrizione					
<i>Iacolletti Alessandro</i>	7/04/2008	<i>Prima versione caso d'uso</i>			
<i>Pizza Ernesto</i>	08/04/2008	<i>Revisione del caso d'uso</i>			

## Modulo <Annullamento Prenotazione>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	CANC_PREN	
	Use Cases Nome:	Annullamento prenotazione	
	Autori:	Vicidomini Vincenzo	
	Data:	7/04/2008	
	Versione:	1.0	
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al cliente di effettuare la cancellazione della propria prenotazione</i>	
	Attori Primario:	<i>Cliente</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il caso d'uso è innescato quando si vuole effettuare l'annullamento della propria ordinazione</i>	
	Condizioni: Pre-condizioni	<i>Ci deve essere almeno una prenotazione</i>	
	Post-condizioni di successo:	<i>L'utente riesce ad annullare la prenotazione. I dati vengono salvati</i>	
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che l'annullamento non è stata effettuato. I dati non vengono salvati</i>	
	Priorità	<i>Media</i>	
	Casi d'uso inclusi:	<i>visualizzazione prenotazioni</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>Gestione prenotazione</i>	
<b>Descrizione degli scenari</b>	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
<b>Scenario principale</b>	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Seleziona la funzione annulla prenotazione</i>
	2	<i>Sistema</i>	<i>Include (Visualizzazione prenotazione)</i>
	3	<i>Cliente</i>	<i>Sceglie la prenotazione da annullare</i>
	4	<i>Sistema</i>	<i>Chiede all'utente se è sicuro di annullare la prenotazione</i>
	5	<i>Cliente</i>	<i>Conferma l'annullamento</i>
	6	<i>Sistema</i>	<i>Cancella la prenotazione</i> <i>Salva i dati</i>
	7	<i>Sistema</i>	<i>Informa l'utente che la cancellazione è stata effettuata</i>
<b>Scenario alternativo &lt;NESSUNA PRENOTAZIONE&gt;</b>			
	Passo	Soggetto	Descrizione

	2.1	<i>Sistema</i>	<i>Informa l'utente che non è stata effettuata nessuna prenotazione in precedenza</i>
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
<b>2° Scenario alternativo &lt;ANNULLAMENTO&gt;</b>			
	5.1	<i>Cliente</i>	<i>Non conferma l'annullamento</i>
	5.2	<i>Sistema</i>	<i>Ritorna al punto 2</i>
<b>3° Scenario alternativo &lt;PRENOTAZIONE SCADUTA&gt;</b>			
	4.1	<i>Sistema</i>	<i>Informa l'utente che non è più possibile annullare la prenotazione (tempo scaduto)</i>
	4.2	<i>Sistema</i>	<i>Ritorna al punto 2</i>
<b>4° Scenario alternativo &lt;ERRORE SISTEMA&gt;</b>			
	6.1	<i>Sistema</i>	<i>Informa l'utente che la prenotazione non è stata annullata per un errore di sistema</i>
	6.2	<i>Sistema</i>	<i>Ritorna al punto 2</i>
<b>Iterazioni</b>			
	<b>Autore</b>	<b>Data</b>	<b>Descrizione</b>
	Vicidomini Vincenzo	7/04/2008	<i>Prima versione caso d'uso</i>
	Vicidomini Vincenzo	08/04/2008	<i>Revisione del caso d'uso</i>

## Modulo <Visualizzazione Prenotazione>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<b>VISPRE</b>	
	Use Cases Nome:	<i>Visualizzazione Prenotazione</i>	
	Autori:	<i>Mercurio Antonio</i>	
	Data:	<i>07 / 04 /2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Permette al cliente la visualizzazione delle prenotazioni fatte al ristorante, con tutte le informazioni relativi ai piatti ordinati, alle quantità, ecc. .</i>	
	Attori Primario:	<i>Cliente.</i>	
	Altri:	<i>Nessuno.</i>	
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole visualizzare l'elenco delle prenotazioni fatte al ristorante.</i>	
	Condizioni: Pre-condizioni	<i>Si deve essere clienti della struttura.</i>	
	Post-condizioni di successo:	<i>Vengono visualizzate le prenotazioni fatte al ristorante.</i>	
	Post-condizioni per Fallimento:	<i>Viene visualizzato un messaggio che informa il cliente che non ha effettuato prenotazioni al ristorante.</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Bassa.</i>	
	Casi d'uso inclusi:	<i>Nessuno.</i>	
	Estende il caso d'uso:	<i>Nessuno.</i>	
	Specializza il caso d'uso:	<i>Gestione Menù.</i>	
	Requisiti aggiuntivi	<i>Nessuno.</i>	
<b>Descrizione degli scenari</b>	Problemi vari	<i>Nessuno.</i>	
	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Seleziona la funzione di Visualizzazione Prenotazione.</i>
	2	<i>Sistema</i>	<i>Visualizza le prenotazioni effettuate</i>
	3	<i>Cliente</i>	<i>Seleziona una prenotazione per avere tutte le informazioni relative a quella</i>
	4	<i>Sistema</i>	<i>Visualizza i dettagli della prenotazione selezionata.</i>
	Scenario alternativo*		
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa il cliente che non ha effettuato prenotazioni al ristorante.</i>
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
	Autore	Data	Descrizione
	<i>Mercurio Antonio</i>	<i>07/04/2008</i>	<i>Prima versione del caso d'uso.</i>
	<i>Mercurio Antonio</i>	<i>07/04/2008</i>	<i>Revisione del caso d'uso.</i>

## Modulo <Visualizza Saldo >

Questo modulo a differenza degli altri non rientra in tutti gli scenari in quanto il cliente può visualizzare il suo saldo in ogni momento.

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	VIS_SAL	
	Use Cases Nome:	Visualizzazione Saldo	
	Autori:	Mercurio Antonio	
	Data:	06 / 04 /2008	
	Versione:	1.0	
<b>Informazioni principali:</b>	Obiettivi:	<i>Il cliente può visualizzare il suo attuale saldo presso la struttura, in questo modo è comodamente e costantemente informato delle spese effettuate</i>	
	Attori Primario:	<i>Cliente</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il caso d'uso è avviato quando si vuole visualizzare il saldo presso la struttura</i>	
	Condizioni: Pre-condizioni	<i>Si deve essere clienti della struttura</i>	
	Post-condizioni di successo:	<i>L'utente riesce a visualizzare il proprio saldo</i>	
	Post-condizioni per Fallimento:	<i>Il sistema informa il cliente che non è momentaneamente possibile visualizzare il saldo</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Bassa</i>	
	Casi d'uso inclusi:	<i>Nessuno</i>	
	Estende il caso d'uso:	<i>Nessuno</i>	
	Specializza il caso d'uso:	<i>Gestione prenotazione</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	<i>Cliente</i>	<i>Accede alla funzione di Visualizzazione Saldo.</i>
	2	<i>Sistema</i>	<i>Accede al conto del cliente.</i>
	3	<i>Sistema</i>	<i>Visualizza il saldo del cliente.</i>
	Autore	Data	Descrizione
	<i>Mercurio Antonio</i>	<i>06/04/2008</i>	<i>Prima versione del caso d'uso.</i>
	<i>Mercurio Antonio</i>	<i>08/04/2008</i>	<i>Revisione del caso d'uso</i>

## Modulo <Gestione Menu>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>GES_MENU'</i>
	Use Cases Nome:	<i>Gestione menù</i>
	Autori:	<i>Mercurio Antonio</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>E' una sorta di generalizzazione delle funzionalità richieste per la gestione dei menù per il servizio in camera; le operazioni fondamentali sono quelle di inserimento, modifica, visualizzazione e consultazione prenotazioni</i>
	Attori	
	Primario:	<i>Gestore</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>L'evento d'innesto è per l'esigenza di gestione o di modifica di un determinato menù</i>
	Condizioni:	
	Pre-condizioni	<i>Il gestore deve essere riconosciuto dal sistema</i>
	Post-condizioni di successo:	<i>Il gestore riesce ad effettuare con successo la gestione dei menù</i>
	Post-condizioni per Fallimento:	<i>Il fallimento avviene solo in caso di mancato riconoscimento nel sistema.</i>
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta</i>
	Casi d'uso inclusi:	<i>nessuno</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>nessuno</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>
<b>Descrizione degli scenari</b>	Scenario principale*	
	Passo	Soggetto
	1	<i>Gestore</i>
	<i>(Vedi casi d'uso che specializzano questo)</i>	
	Iterazioni	
	Autore	Data
	<i>Mercurio Antonio</i>	<i>7/04/2008</i>
		<i>Prima versione caso d'uso</i>

## Modulo <Visualizza Menu>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>VIS_MENU'</i>
	Use Cases Nome:	<i>Visualizza menù</i>
	Autori:	<i>Pacifico Marta</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Visualizzare le informazioni riguardanti il menù del cliente e per il management dei menù da parte del gestore</i>
	Attori Primario:	<i>Cliente</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>il caso d'uso è innescato quando si vogliono visualizzare dei menù</i>
	Condizioni: Pre-condizioni	<i>deve esistere almeno un menù</i>
	Post-condizioni di successo:	<i>L'utente riesce a leggere le informazioni richieste</i>
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>In caso di fallimento non verrà visualizzato nessun menù</i>
	Priorità	<i>Bassa</i>
	Casi d'uso inclusi:	<i>nessuno</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>gestione prenotazione</i>
<b>Descrizione degli scenari</b>	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>
	Scenario principale*	
	Passo	Soggetto
	1	<i>Cliente, Gestore</i>
	2	<i>Sistema</i>
	3	<i>Cliente, Gestore</i>
	4	<i>Sistema</i>
	1° Scenario alternativo <NESSUNA ATTIVITA'>	
	Passo	Soggetto
	2.1	<i>Sistema</i>
	2.2	<i>Sistema</i>
	Annotazioni	
	Desrizione	
	3	<i>Il cliente non può visualizzare menù dei giorni precedenti, e non può visualizzare menù riferiti alla settimana successiva</i>

	Iterazioni		
	Autore	Data	Descrizione
	Pacifico Marta	7/04/2008	<i>Prima versione caso d'uso</i>
	Pizza Ernesto	08/04/2008	<i>Revisione del caso d'uso</i>

## Modulo <Valida Utente>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>LOG_PWD</i>	
	Use Cases Nome:	<i>Valida</i>	
	Autori:	<i>Pizza Ernesto</i>	
	Data:	<i>7/04/2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Il gestore viene riconosciuto dal sistema e quindi accede ad uno o più casi d'uso</i>	
	Attori Primario:	<i>gestore</i>	
	Altri:	<i>nessuno</i>	
	Azione d'avvio:	<i>il caso d'uso è innescato quando il gestore ha intenzione di inserire o modificare un menù</i>	
	Condizioni: Pre-condizioni	<i>deve esistere almeno un utente</i>	
	Post-condizioni di successo:	<i>Il gestore è stato riscontrato nel sistema</i>	
	Post-condizioni per Fallimento:	<i>Tentativi effettuati pari al numero massimo previsti</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Alta</i>	
	Casi d'uso inclusi:	<i>nessuno</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>gestione menù</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	<i>Gestore</i>	<i>Selezione inserimento/modifica di un menù</i>
	2	<i>Sistema</i>	<i>Visualizza la form di Login e Password</i>
	3	<i>Gestore</i>	<i>Imposta Login e Password</i>
	4	<i>Sistema</i>	<i>Verifica che la coppia di valori (Login, PWD) individui il gestore nel DB del sistema</i>
	5	<i>Sistema</i>	<i>Visualizza menù dedicato alla modifica o all'inserimento di un menù</i>
	1° Scenario alternativo<LOGIN & PASSWORD FALLITA>		
	Passo	Soggetto	Descrizione
	4.1	<i>Sistema</i>	<i>Verifica Login e PWD fallita</i>
	4.2	<i>Sistema</i>	<i>Riprende dal punto 2</i>
	Annotazioni		
	Descrizione		

	3	<i>Il gestore deve compilare correttamente la form LOGIN e PASSWORD</i>
<b>Iterazioni</b>		
Autore	Data	Descrizione
<i>Pizza Ernesto</i>	07/04/2008	<i>Prima versione caso d'uso</i>
<i>Pizza Ernesto</i>	08/04/2008	<i>Revisione del caso d'uso</i>

## Modulo <Inserimento Menù>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>INS_MENU'</i>
	Use Cases Nome:	<i>Inserimento menù</i>
	Autori:	<i>Pacifico Marta</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>inserire un nuovo menù</i>
	Attori	
	Primario:	<i>Gestore</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Connessione al sistema e convalida</i>
	Condizioni:	<i>il gestore deve essere riconosciuto dal sistema</i>
	Pre-condizioni	
<b>Altre informazioni (opzionali):</b>	Post-condizioni di successo:	<i>Il gestore esegue il corretto inserimento di un menù. I dati vengono salvati</i>
	Post-condizioni per Fallimento:	<i>Il gestore non riesce ad eseguire nessun inserimento. Il sistema rimane invariato</i>
	Priorità	<i>Alta</i>
<b>Altre informazioni (opzionali):</b>	Casi d'uso inclusi:	<i>valida utente visualizzazione menu</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>gestione menù</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>nessuno</i>
<b>Descrizione degli scenari</b>	Scenario principale*	
	Passo	Soggetto
	1	<i>Gestore</i>
	2	<i>Sistema</i>
	3	<i>Sistema</i>
	4	<i>Gestore</i>
	5	<i>Sistema</i>
	6	<i>Gestore</i>
	7	<i>Gestore</i>
	8	<i>Sistema</i>
1 Scenario alternativo<ERRORE DATA>	Passo	
	4.1	<i>Sistema</i>
	4.2	<i>Sistema</i>
	Ritorna al punto 4	

	2° Scenario alternativo <ERRORE COMPILAZIONE>		
6.1	Sistema	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>	
6.2	Sistema	<i>Ritorna al punto 5</i>	
3° Scenario alternativo <ERRORE SISTEMA>			
7.1	Sistema	<i>Informa l'utente che la prenotazione non è stata annullata per un errore di sistema</i>	
7.2	Sistema	<i>Ritorna al punto 3</i>	
Iterazioni			
Autore	Data	Descrizione	
Pacifico Marta	7/04/2008	<i>Prima versione caso d'uso</i>	
Mercurio Antonio	08/04/2008	<i>Revisione del caso d'uso</i>	

## Modulo <Modifica Menù>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>MOD_MENU'</i>	
	Use Cases Nome:	<i>Modifica menù</i>	
	Autori:	<i>Pacifico Marta</i>	
	Data:	<i>7/04/2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al gestore di modificare un menù</i>	
	Attori Primario:	<i>Gestore</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il caso d'uso è innescato quando il gestore vuole effettuare la modifica di un menù</i>	
	Condizioni: Pre-condizioni	<i>deve esistere almeno un menù</i>	
	Post-condizioni di successo:	<i>Il gestore riesce a modificare un menù. Il sistema salva i dati</i>	
<b>Altre informazioni (opzionali):</b>	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la modifica non è stata effettuata. I dati non vengono salvati</i>	
	Priorità	<i>Alta</i>	
	Casi d'uso inclusi:	<i>visualizzazione menù valida utente</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>Gestione menù</i>	
<b>Descrizione degli scenari</b>	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
<b>Scenario principale</b>	Passo	Soggetto	Descrizione
	1	<i>Gestore</i>	<i>Seleziona la funzione modifica prenotazione</i>
	2	<i>Sistema</i>	<i>Include (valida)</i>
	3	<i>Sistema</i>	<i>Include (visualizzazione)</i>
	4	<i>Sistema</i>	<i>Visualizza la form per scegliere la data in cui inserire il menù</i>
	5	<i>Gestore</i>	<i>Compila form data</i>
	6	<i>Sistema</i>	<i>Visualizza la form del menù da modificare</i>
	7	<i>Gestore</i>	<i>Modifica i campi desiderati</i>
	8	<i>Gestore</i>	<i>Conferma la modifica</i>
	9	<i>Sistema</i>	<i>Registra i dati e informa l'utente che la modifica è stata effettuata</i>
<b>1° Scenario alternativo &lt;NESSUN MENU'&gt;</b>			
<b>Passo</b>			
<b>Soggetto</b>			
<b>Descrizione</b>			

	2.1	<i>Sistema</i>	<i>Informa l'utente che non è stata effettuata nessuna prenotazione in precedenza</i>
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
<b>2° Scenario alternativo &lt;ERRATA COMPILAZIONE&gt;</b>			
	7.1	<i>Sistema</i>	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>
	7.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
<b>3° Scenario alternativo &lt;ERRATA DATA&gt;</b>			
	4.1	<i>Sistema</i>	<i>Informa l'utente che è stata immessa una data errata (non si possono modificare menù riferiti al giorno corrente e alle settimane successive o precedenti)</i>
	4.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
<b>4° Scenario alternativo &lt;ERRORE SISTEMA&gt;</b>			
	6.1	<i>Sistema</i>	<i>Informa l'utente che la prenotazione non è stata modificata per un errore di sistema</i>
	6.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
Iterazioni			
	Autore	Data	Descrizione
	Pacifico Marta	7/04/2008	<i>Prima versione caso d'uso</i>
	Iacoletti Alessandro	08/04/2008	<i>Revisione del caso d'uso</i>

## Modulo <Consultazione Prenotazioni>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	CON_PRE	
	Use Cases Nome:	<i>Consultazione prenotazioni</i>	
	Autori:	<i>Vicidomini Vincenzo</i>	
	Data:	<i>7/04/2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Visualizzare l'elenco delle prenotazioni</i>	
	Attori Primario:	<i>Gestore</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il gestore vuole visualizzare le prenotazioni effettuate</i>	
	Condizioni: Pre-condizioni	<i>deve esistere almeno una prenotazione</i>	
	Post-condizioni di successo:	<i>Il gestore visualizza le informazioni relative alle prenotazioni</i>	
	Post-condizioni per Fallimento:	<i>Il gestore non riesce a visualizzare le prenotazioni</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	<i>Media</i>	
	Casi d'uso inclusi:	<i>nessuno</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>Gestione menù</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	Gestore	<i>Seleziona la funzione consulta prenotazioni</i>
	2	Sistema	<i>Visualizza l'elenco delle prenotazioni del giorno corrente</i>
	3	Gestore	<i>Sceglie una prenotazione per avere maggiori informazioni</i>
	Annotazioni		
	Desrizione		
	2	<i>Le prenotazioni da visualizzare devono rientrare nelle date di soggiorno dei clienti</i>	
	Iterazioni		
	Autore	Data	Descrizione
	<i>Vicidomini Vincenzo</i>	<i>7/04/2008</i>	<i>Prima versione caso d'uso</i>
	<i>Vicidomini Vincenzo</i>	<i>08/04/2008</i>	<i>Revisione del caso d'uso</i>

### 3.5.2.2 Modello Casi d'uso Servizio Camere

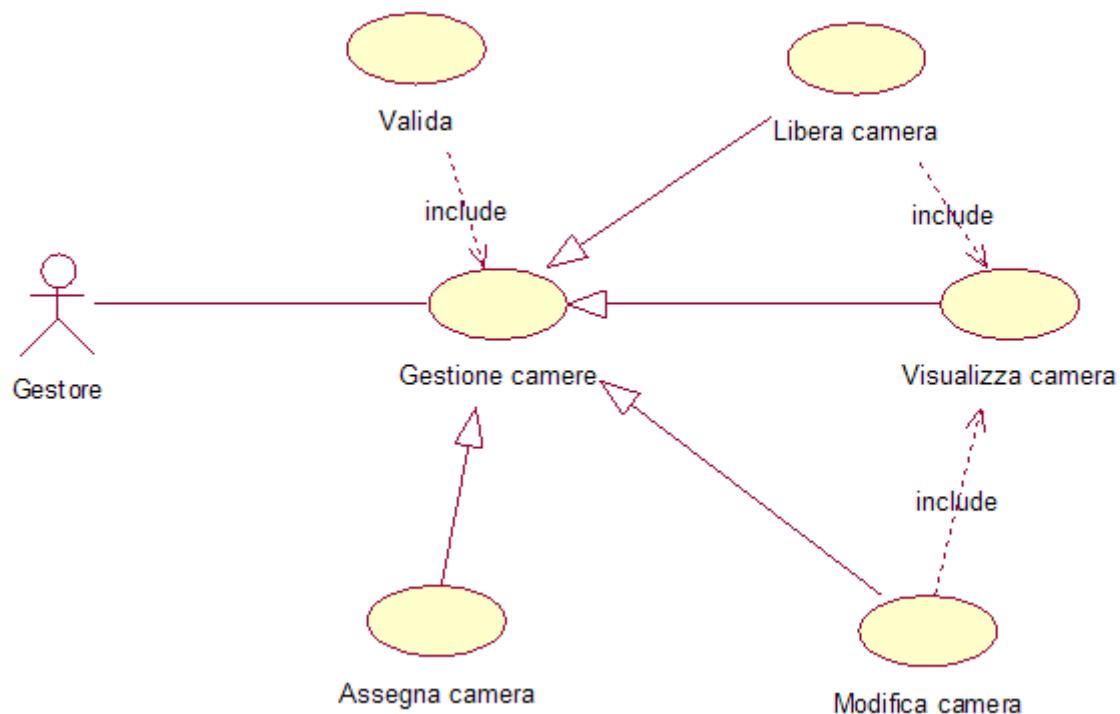


Figura 4: Casi d'uso Servizio Camere

### Attori

Attore	Versione	Autore
Gestore delle camere	1.0	Mercurio Antonio
Interazioni		
Specializza l'attore:		
Caso d'uso	Primario	Frequenza
6. Valida	X	Alta
7. Visualizza camera;	X	Alta
8. Assegna camera;	X	Alta
9. Modifica camera;	X	Bassa
10. Libera camera;	X	Alta

## Organizzazione del modello

### Gestione Camere

1. Visualizza camera
2. Valida
3. Assegna camera
4. Modifica camera
5. Libera camera

### Modulo <Gestione Camere>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>GES_CAM</i>
	Use Cases Nome:	<i>Gestione camere</i>
	Autori:	<i>Mercurio Antonio</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>E' l'insieme delle funzionalità che permettono al gestore la gestione completa sulle camere di un albergo, come ad esempio l'assegnazione di una camera, la stampa del saldo</i>
	Attori Primario:	<i>Gestore delle camere</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso viene innescato quando il gestore delle camere vuole avere delle informazioni su alcune camere o assegnare una stanza ad un cliente</i>
	Condizioni: Pre-condizioni	<i>deve esistere almeno una camera</i>
	Post-condizioni di successo:	<i>Il gestore riesce ad effettuare le sue funzionalità</i>
	Post-condizioni per Fallimento:	<i>Il gestore non riesce ad avere alcuna informazione sulle camere dell'albergo</i>
	Priorità	<i>Alta</i>

<b>Altre informazioni (opzionali):</b>	Casi d'uso inclusi:	<i>nessuno</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>nessuno</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	Gestore delle camere	(Vedi casi d'uso che specializzano questo )
	Iterazioni		
	Autore	Data	Descrizione
	<i>Mercurio Antonio</i>	<i>7/04/2008</i>	<i>Prima versione caso d'uso</i>

### **Modulo <Visualizza Camera>**

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	<i>VIS_CAM</i>
	Use Cases Nome:	<i>visualizza camera</i>
	Autori:	<i>Vicidomini Vincenzo</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Visualizzare le informazioni riguardanti le camere dell'albergo</i>

	Attori Primario:  Altri:	<i>Gestore</i>  <i>Nessuno</i>
	Azione d'avvio:	<i>il caso d'uso è innescato quando si vogliono visualizzare delle informazioni su alcune camere</i>
	Condizioni: Pre-condizioni	<i>ci deve essere almeno una camera</i>
	Post-condizioni di successo:	<i>L'utente riesce a leggere le informazioni richieste</i>
	Post-condizioni per Fallimento:	<i>In caso di fallimento non verrà visualizzata nessuna camera</i>
	Priorità	<i>Bassa</i>
	Altre informazioni (opzionali):	Casi d'uso inclusi: <i>nessuno</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>gestione camere</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>
Descrizione degli scenari	Scenario principale*	
	Passo	Soggetto
	1	<i>Gestore</i>
	2	<i>Sistema</i>
	3	<i>Gestore</i>
	4	<i>Sistema</i>
	Iterazioni	
	Autore	Data
	<i>Vicidomini Vincenzo</i>	<i>7/04/2008</i>
	<i>Vicidomini Vincenzo</i>	<i>8/04/2008</i>

## Modulo <Valida>

Vedi modello use-case relativo al ristorante.

## Modulo <Assegna Camera>

<b>Identificazione dei casi d'uso:</b>	Use Cases ID:	ASS_CAM	
	Use Cases Nome:	Assegna camera	
	Autori:	Pizza Ernesto	
	Data:	7/04/2008	
	Versione:	1.0	
<b>Informazioni principali:</b>	Obiettivi:	<i>il gestore deve fare l'assegnazione delle camere ai rispettivi clienti secondo le proprie esigenze</i>	
	Attori Primario:	<i>gestore delle camere</i>	
	Altri:	<i>nessuno</i>	
	Azione d'avvio:	<i>il caso d'uso viene innescato quando vi è un nuovo cliente nell'albergo</i>	
	Condizioni: Pre-condizioni	<i>ci deve essere una stanza libera all'interno dell'albergo</i>	
	Post-condizioni di successo:	<i>Il gestore effettua l'assegnazione delle camere e i dati vengono salvati dal sistema</i>	
	Post-condizioni per Fallimento:	<i>Il gestore non effettua alcuna assegnazione e il sistema rimane immutato</i>	
<b>Altre informazioni (opzionali):</b>	Priorità	Alta	
	Casi d'uso inclusi:	<i>nessuno</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
	Specializza il caso d'uso:	<i>gestione camere</i>	
	Requisiti aggiuntivi	Nessuno	
<b>Descrizione degli scenari</b>	Scenario principale*		
	Passo	Soggetto	Descrizione
	1	Gestore	<i>Seleziona la funzione assegna camera</i>
	2	Sistema	<i>Visualizza le camere libere all'interno dell'albergo</i>
	3	Gestore	<i>Seleziona camera giusta</i>

	4	<i>Sistema</i>	<i>Visualizza la form riguardante la stanza</i>		
	5	<i>Gestore</i>	<i>Compila la form in ogni suo campo</i>		
	6	<i>Sistema</i>	<i>Salva i dati</i>		
<b>1 Scenario alternativo &lt;NESSUNA CAMERA LIBERA&gt;</b>					
	Passo	<i>Soggetto</i>	<i>Descrizione</i>		
	2.1	<i>Sistema</i>	<i>Non ci sono camere libere</i>		
	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>		
<b>2 Scenario alternativo &lt;ERRORE COMPILAZIONE&gt;</b>					
	5.1	<i>Sistema</i>	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>		
	5.2	<i>Sistema</i>	<i>Ritorna al punto 5</i>		
<b>Annotazioni</b>					
<b>Descrizione</b>					
	5	<i>Il gestore per effettuare un esatta assegnazione della camera deve compilare la form in ogni suo campo; fondamentali sono i campi del numero delle persone e la modalità della stanza mezza pensione o pensione completa</i>			
<b>Iterazioni</b>					
<b>Autore</b> <b>Data</b> <b>Descrizione</b>					
Pizza Ernesto	7/04/2008	<i>Prima versione caso d'uso</i>			
Pizza Ernesto	8/04/2008	<i>Revisione del caso d'uso</i>			

## Modulo <Modifica Camera >

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>MOD_CAM</i>
	Use Cases Nome:	<i>Modifica camera</i>
	Autori:	<i>Pacifico Marta</i>
	Data:	<i>7/04/2008</i>
	Versione:	<i>1.0</i>
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al gestore di modificare delle camere</i>
	Attori	
	Primario:	<i>Gestore delle camere</i>
	Altri:	<i>Nessuno</i>
	Azione d'avvio:	<i>Il caso d'uso è innescato quando il gestore vuole effettuare la modifica di una camera</i>
	Condizioni:	
Pre-condizioni		<i>deve esistere almeno una camera occupata</i>
	Post-condizioni di successo:	<i>Il gestore riesce a modificare una camera. Il sistema salva i dati</i>
	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la modifica non è stata effettuata. I dati non vengono salvati</i>
	Priorità	<i>Alta</i>
<b>Altre informazioni (opzionali):</b>	Casi d'uso inclusi:	<i>visualizza camera</i>
	Estende il caso d'uso:	<i>nessuno</i>
	Specializza il caso d'uso:	<i>Gestione camere</i>
	Requisiti aggiuntivi	<i>Nessuno</i>
	Problemi vari	<i>Nessuno</i>
<b>Descrizione degli scenari</b>	Scenario principale	
	Passo	Soggetto
	1	<i>Gestore</i>
		<i>Selezione la funzione modifica camera</i>
	2	<i>Sistema</i>
		<i>Include (visualizza camera)</i>
	3	<i>Sistema</i>
		<i>Visualizza le stanze occupate</i>
	4	<i>Gestore</i>
		<i>Selezione la stanza da modificare</i>
	5	<i>Sistema</i>
		<i>Visualizza la form della camera da modificare</i>
	6	<i>Gestore</i>
		<i>Modifica i campi desiderati</i>
	7	<i>Gestore</i>
		<i>Conferma la modifica</i>
	8	<i>Sistema</i>
		<i>Registra i dati e informa l'utente che la modifica è stata effettuata</i>
	1° Scenario alternativo <NESSUNA CAMERA>	
	Passo	Soggetto
	2.1	<i>Sistema</i>
		<i>Informa l'utente che non c'è nessuna stanza occupata</i>

	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
<b>2° Scenario alternativo &lt;ERRATA COMPILAZIONE&gt;</b>			
	6.1	<i>Sistema</i>	<i>Informa l'utente che non sono stati compilati dei campi obbligatori</i>
	6.2	<i>Sistema</i>	<i>Ritorna al punto 5</i>
<b>4° Scenario alternativo &lt;ERRORE SISTEMA&gt;</b>			
	8.1	<i>Sistema</i>	<i>Informa l'utente che la camera non è stata modificata per un errore di sistema</i>
	8.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
Iterazioni			
Autore	Data	Descrizione	
Pacifico Marta	7/04/2008	<i>Prima versione caso d'uso</i>	
Pacifico Marta	8/04/2008	<i>Revisione del caso d'uso</i>	

## Modulo <Libera Camera>

<b>Identificazione Dei casi d'uso:</b>	Use Cases ID:	<i>LIB_CAM</i>	
	Use Cases Nome:	<i>Libera camera</i>	
	Autori:	<i>Iacoletti Alessandro</i>	
	Data:	<i>7/04/2008</i>	
	Versione:	<i>1.0</i>	
<b>Informazioni principali:</b>	Obiettivi:	<i>Permettere al gestore di liberare le camere</i>	
	Attori		
	Primario:	<i>Gestore delle camere</i>	
	Altri:	<i>Nessuno</i>	
	Azione d'avvio:	<i>Il caso d'uso è innescato quando il gestore vuole liberare una camera</i>	
	Condizioni:		
	Pre-condizioni	<i>deve esistere almeno una camera occupata</i>	
<b>Altre informazioni (opzionali):</b>	Post-condizioni di successo:	<i>Il gestore riesce a liberare una camera. Il sistema salva i dati</i>	
	Post-condizioni per Fallimento:	<i>Il sistema informa l'utente che la modifica non è stata effettuata. I dati non vengono salvati</i>	
	Priorità	<i>Alta</i>	
	Casi d'uso inclusi:	<i>visualizza camera</i>	
	Estende il caso d'uso:	<i>nessuno</i>	
<b>Descrizione degli scenari</b>	Specializza il caso d'uso:	<i>Gestione camere</i>	
	Requisiti aggiuntivi	<i>Nessuno</i>	
	Problemi vari	<i>Nessuno</i>	
	Scenario principale		
	Passo	Soggetto	Descrizione
	1	<i>Gestore</i>	<i>Seleziona la funzione libera camera</i>
	2	<i>Sistema</i>	<i>Include (visualizza camera)</i>
	3	<i>Sistema</i>	<i>Visualizza le stanze occupate</i>
	4	<i>Gestore</i>	<i>Seleziona la stanza da liberare</i>
	5	<i>Sistema</i>	<i>Conferma liberazione</i>
	6	<i>Gestore</i>	<i>Effettua conferma</i>
	7	<i>Gestore</i>	<i>Stampa il saldo della camera</i>
	8	<i>Sistema</i>	<i>Registra i dati e informa l'utente che la modifica è stata effettuata</i>
1° Scenario alternativo <NESSUNA CAMERA>			
	Passo	Soggetto	Descrizione
	2.1	<i>Sistema</i>	<i>Informa l'utente che non c'è nessuna stanza occupata</i>

	2.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
<b>2° Scenario alternativo &lt;NESSUNA CONFERMA&gt;</b>			
	5.1	<i>Gestore</i>	<i>Decide di non confermare</i>
	5.2	<i>Sistema</i>	<i>Termina il caso d'uso</i>
<b>3° Scenario alternativo &lt;ERRORE SISTEMA&gt;</b>			
	8.1	<i>Sistema</i>	<i>Informa l'utente che la camera non è stata liberata per un errore di sistema</i>
	8.2	<i>Sistema</i>	<i>Ritorna al punto 4</i>
<b>Iterazioni</b>			
	Autore	Data	Descrizione
	Iacoletti Alessandro	7/04/2008	<i>Prima versione caso d'uso</i>
	Iacoletti Alessandro	8/04/2008	<i>Revisione del caso d'uso</i>

### 3.5.3 Modello ad oggetti

Il modello dei Diagrammi delle Classi di seguito descritto è composto da una descrizione estesa delle classi che implementeranno gli oggetti individuati nel modello dei Casi d'uso. Trattandosi di un applicazione Client-Server le classi, sono state suddivise in tre livelli:

- **Livello di presentazione**
- **Livello di applicazione**
- **Livello data**

Tali livelli sono strettamente relati tra loro, inoltre per avere un raggruppamento omogeneo delle classi si è ricorsi all'utilizzo dei “*Package Diagrams*” forniti da UML. Di seguito saranno riportate le funzionalità di ogni classe ricorrendo all'utilizzo di “*Template*” simili a quelli utilizzati nel modello dei Casi d'Uso. Inoltre per dare un maggiore chiarezza alle principali funzionalità saranno utilizzati dei “*Sequence Diagram*”.

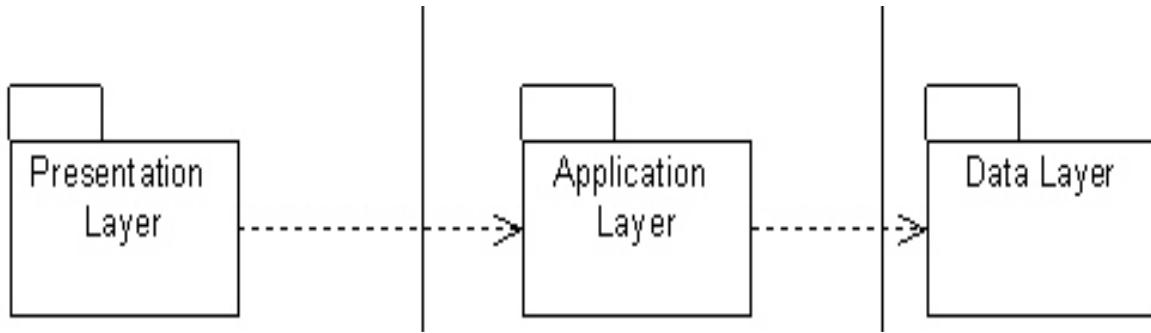


Figura 5: Package Diagram dei tre livelli

### **3.5.3.1 Livello di Presentazione**

La sezione del livello di presentazione si divide in varie parti, una per ogni tipologia di utente presente nel sistema anche se alcune schermate sono uguali per alcune coppie di utente (es. visualizza menu tra cliente e addetto ristorante).

In ognuno dei diagrammi che seguono vi è sempre una classe principale, di solito denominata FrameApplication che rappresenta il frame principale dell'applicazione. Appare quando il computer client su cui è installato viene acceso. Essa è in relazione con tutte le altre classi che rappresentano frame dello stesso package.

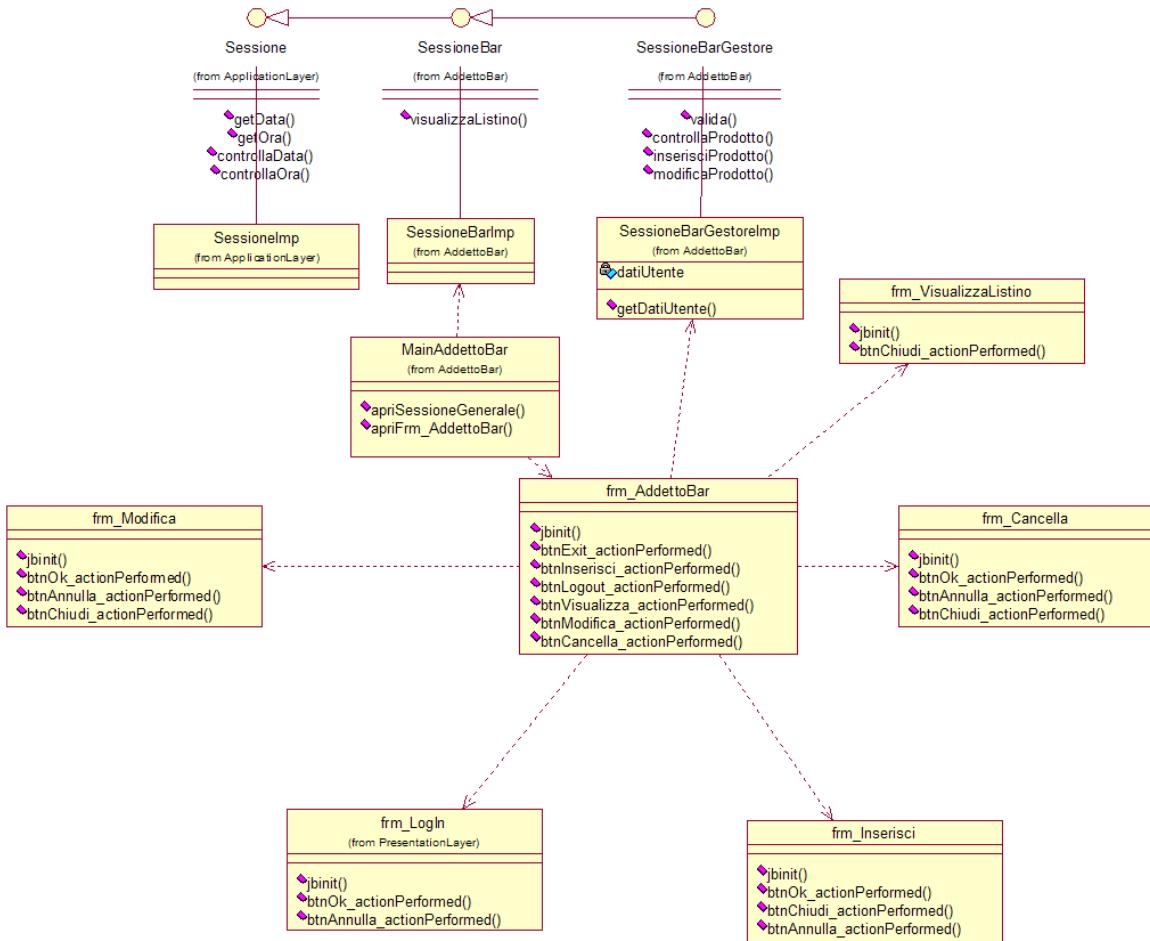
Per l'accesso ai dati sul DataBase il FrameApplication usa una classe che implementa un oggetto remoto.

Inoltre, le altre classi sono state utilizzate soprattutto per la visura dei dati, anche perché l'utente non ha possibilità di modificarli. Il loro ruolo è quello di fornire delle finestre contenenti i dati richiesti.

E' previsto quindi un livello di presentazione per ogni tipologia di utente del sistema, che sono:

- Addetto Bar;
- Addetto Ristorante;
- Cliente;
- Gestore;

## **Livello di presentazione – Addetto Bar**



**Figura 6: Livello di presentazione Addetto Bar**

## Livello di presentazione – Addetto ristorante

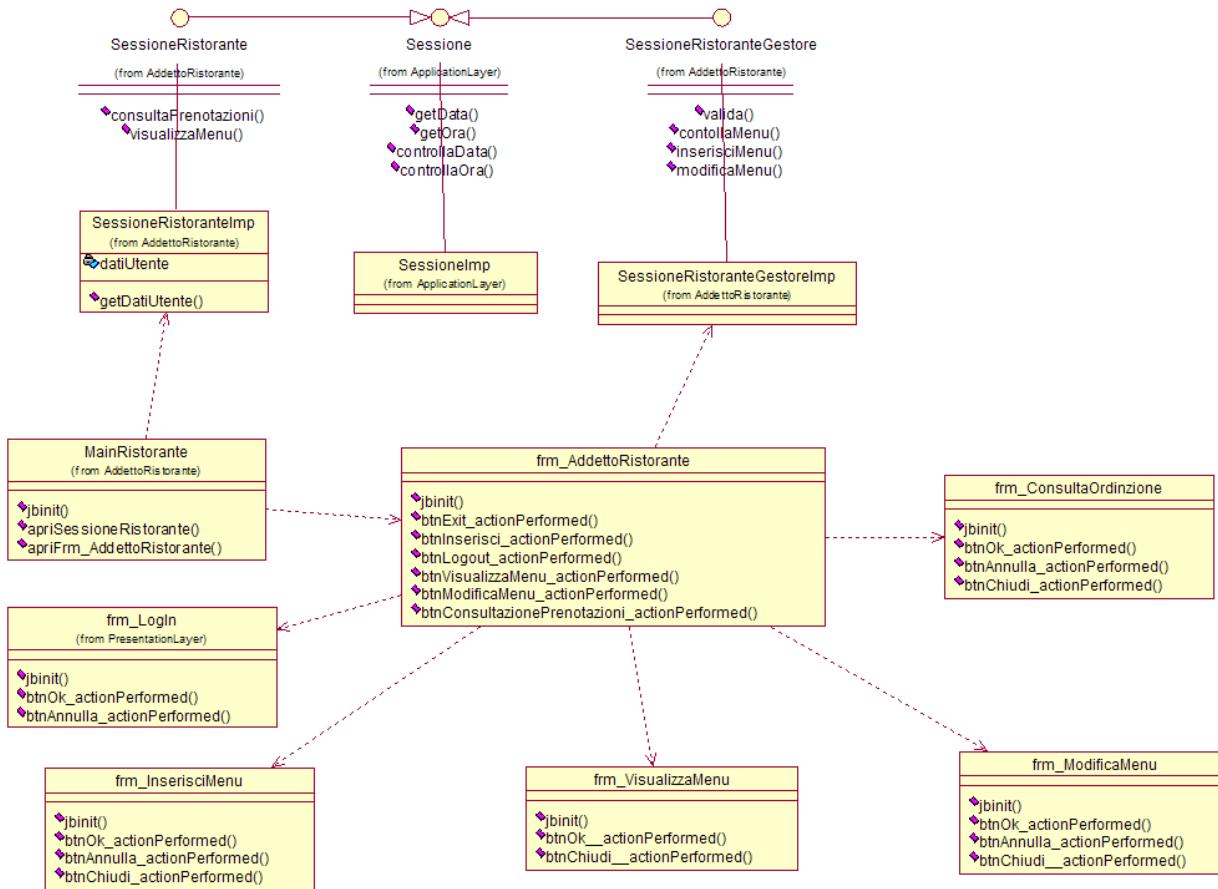


Figura 7: Livello di presentazione Addetto Ristorante

## Livello di presentazione – Gestore

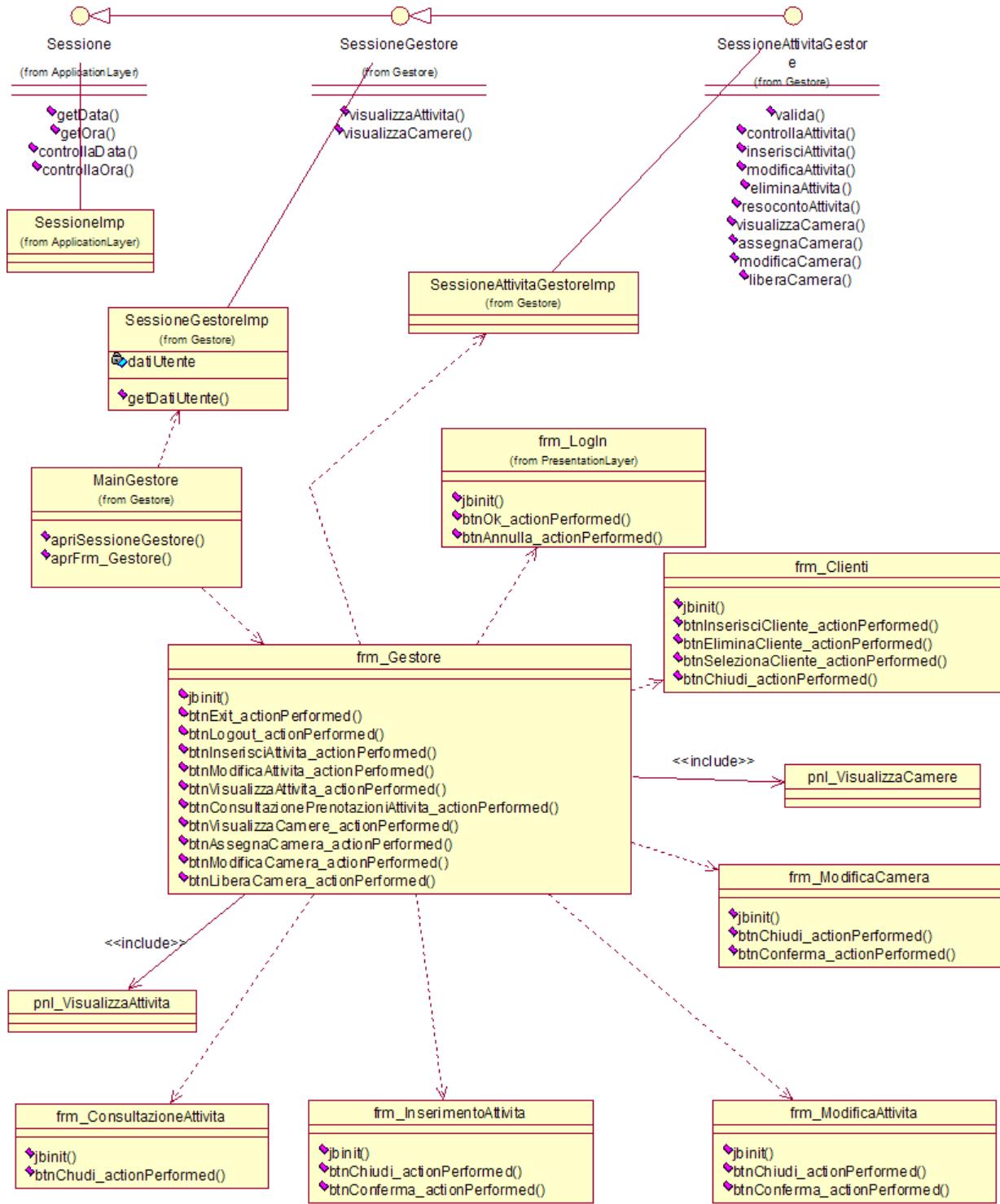


Figura 8: Livello di presentazione Gestore

## Livello di presentazione – Cliente

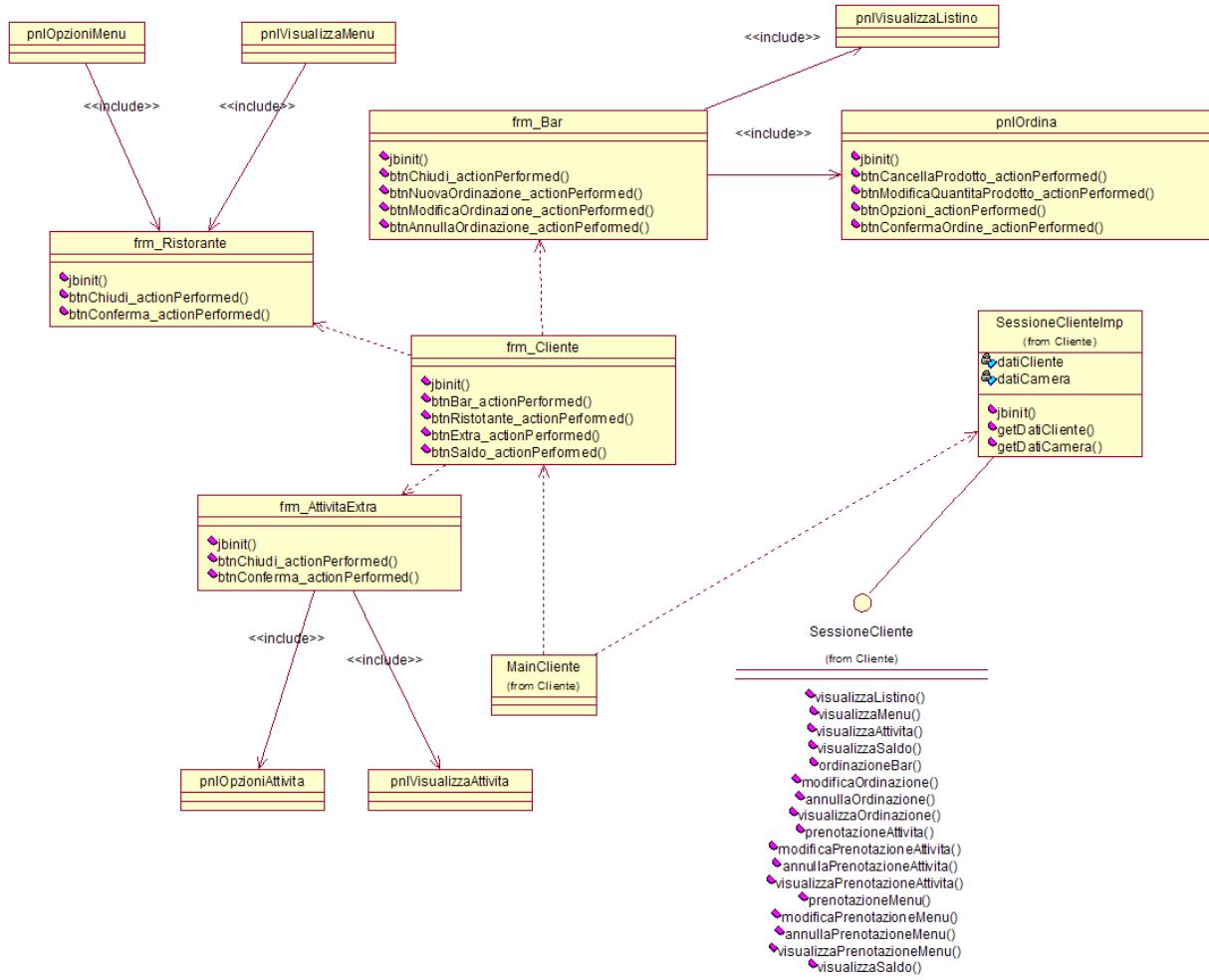


Figura 9: Livello di presentazione Cliente

### 3.5.3.2 Livello di applicazione

Il presente Livello di applicazione è stato anch'esso suddiviso. La funzionalità principale del Layer è quella di illustrare classi e metodi che accedono e che sono utili per capire il funzionamento del sistema.

Si è diviso il tutto in quattro package, che cercano di mostrare la suddivisione del sistema dal punto di vista delle funzionalità degli utenti. Per questo si ha a che fare con i seguenti package che gestiscono le categorie di utenti:



Figura 10: Package del Livello di applicazione

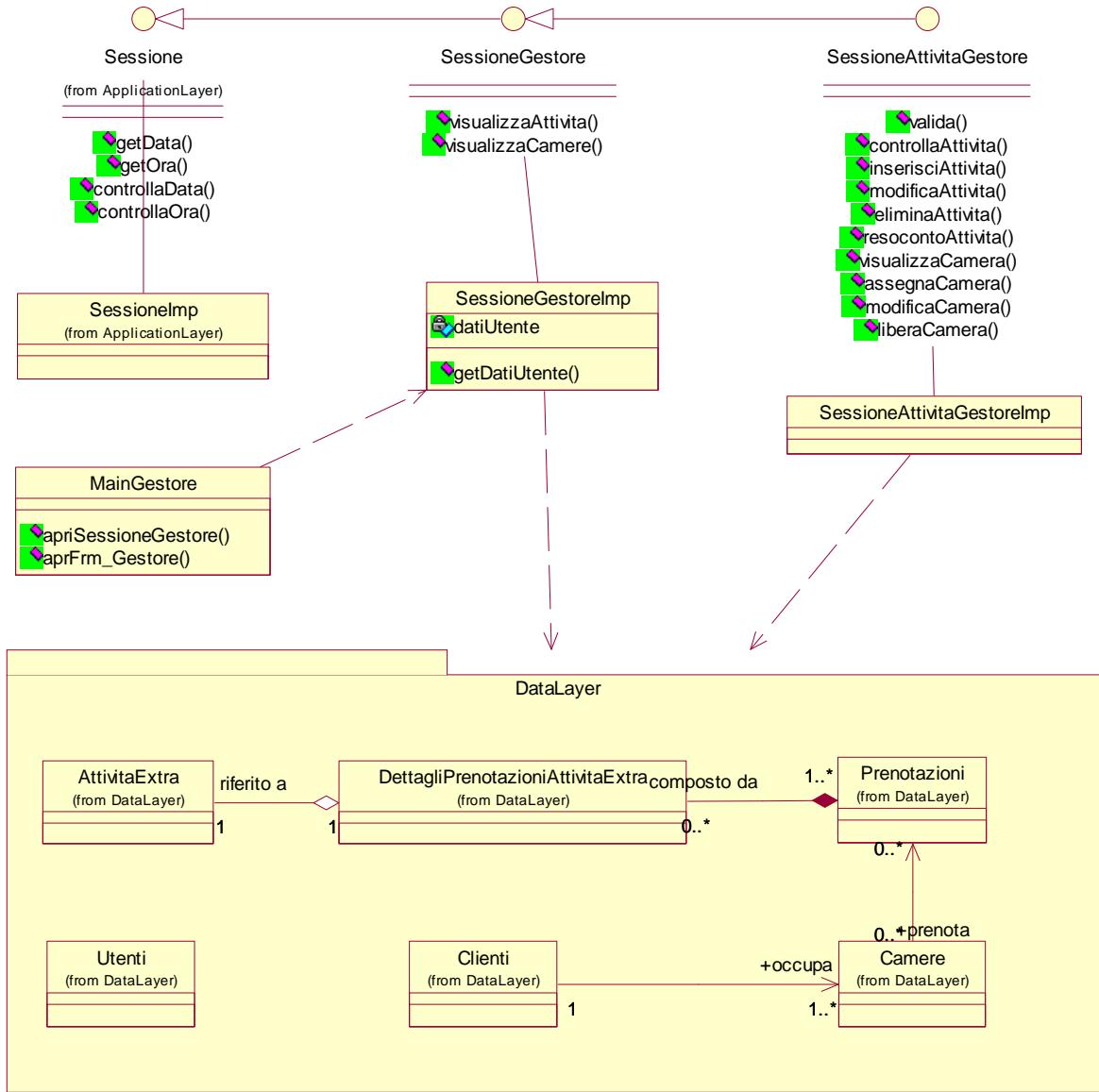
**Gestore:** riguarda l'aspetto funzionale più vasto del progetto in quanto permette all'utente che vi accede (*il Gestore*) di occuparsi della gestione delle camere e delle attività extra. Egli potrà assegnare le camere ai clienti, liberare una camera, modificare il saldo del cliente e così via.

**Addetto Bar:** riguarda l'aspetto funzionale della gestione del servizio in camera del Bar; egli provvede a rispondere agli ordini dei clienti. Avrà inoltre l'incarico di aggiornare il listino, cancellare i prodotti dal listino. Dovrà segnalare l'eventuale mancanza di un determinato prodotto modificandone lo stato.

**Addetto Ristorante:** riguarda la gestione delle prenotazioni dei menu da parte dei clienti. Sarà suo compito inserire i menu relativi di un determinato giorno. Inoltre al momento opportuno dovrà accedere alla funzionalità “Resoconto” mediante la quale potrà entrare in possesso di tutte le prenotazioni dei clienti con le loro relative scelte.

**Cliente:** è l'aspetto funzionale attorno al quale si basa QUICK SERVICE, ognuno dei servizi del nostro software è rivolto alla sua figura. Egli potrà ordinare al bar dalla propria camera, decidere il menu del giorno, partecipare alle attività extra proposte dall'albergo.

# Package Gestore



**Figura 11: Package Gestore**

## Descrizione delle Classi

NOME	SessioneAttivitàGestoreImp
DESCRIZIONE	Accede a questa classe il gestore del ristorante e delle attività extra . Permette di svolgere tutte le operazioni concesse al gestore.
<b>OPERAZIONI</b>	
<b>Valida()</b>	Permette al gestore di accedere al sistema tramite Login e password
Input	Login-Password
Output	True se i valori sono corretti e False se sono errati
<b>controllaAttività()</b>	Durante l'inserimento di una nuova attività controlla se tutti i campi sono stati compilati correttamente.
Input	Dati compilati durante inserimento attività
Output	Null
<b>inserisciAttività()</b>	Inserisce una nuova attività
Input	Dati relativi ad un attività
Output	Null
<b>modificaAttività()</b>	Modifica le informazioni relative ad un attività
Input	Modifica dei campi desiderati
Output	Null
<b>eliminaAttività()</b>	Serve per eliminare un attività
Input	idAttività
Output	Null
<b>resocontoAttività()</b>	Serve per il resoconto di tutte le attività con le eventuali prenotazioni
Input	Null
Output	Fornisce una tabella con le informazioni relative alle attività e le prenotazioni per quelle attività
<b>visualizzaCamera()</b>	Visualizza tutte le informazioni relative ad una camera
Input	idCamera
Output	Fornisce una tabella con tutte le informazioni relative ad una camera
<b>assegnaCamera()</b>	Serve per assegnare la camera
Input	idCamera, idCliente
Output	Null
<b>modificaCamera()</b>	Serve per modificare una camera
Input	idCamera
Output	Null
<b>liberaCamera()</b>	Serve per liberare una camera
Input	idCamera
Output	Null

NOME	SessioneImp
DESCRIZIONE	Realizza l'interfaccia comune a tutte le sessioni definite nell'applicazione.
OPERAZIONI	
<b>getOra()</b>	Visualizza l'ora.
Input	Null.
Output	Restituisce l'ora
<b>getData()</b>	Visualizza la data.
Input	Null.
Output	Restituisce la data
<b>controllaOra()</b>	Serve al thread per controllare se è possibile modificare una ordinazione
Input	Ora in cui è stata effettuata la prenotazione
Output	Null
<b>controllaData()</b>	Serve al thread per controllare se è possibile modificare una ordinazione
Input	Data in cui è stata effettuata la prenotazione
Output	Null

## Package Addetto Bar

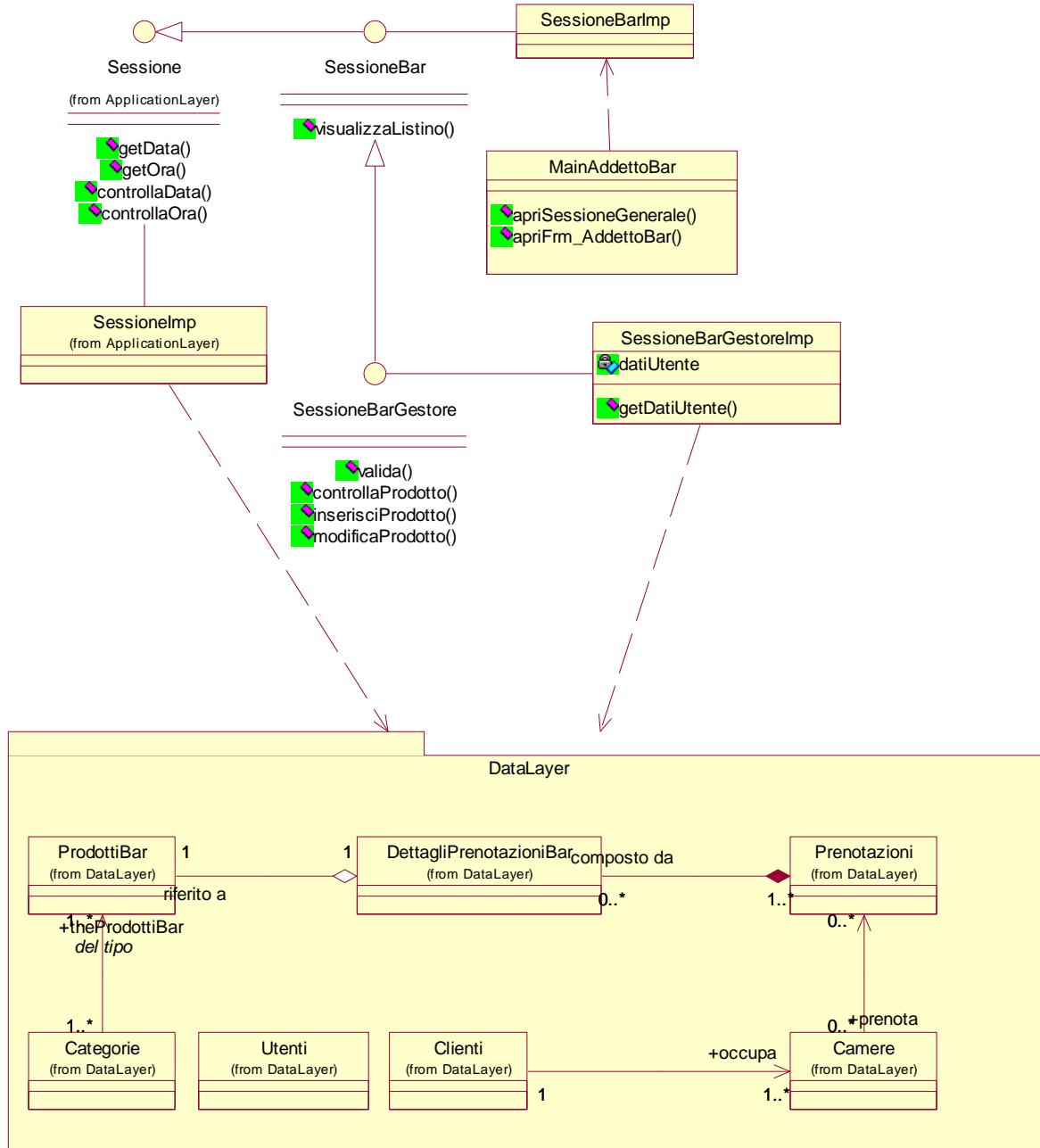


Figura 12: Package Addetto Bar

## **Descrizione delle Classi**

<b>NOME</b>	<b>Sessione Bar</b>
DESCRIZIONE	Informazioni sui prodotti del bar. Accessibile a tutti gli utenti, che lavorano nel bar.
<b>OPERAZIONI</b>	
<b>visualizzaListino()</b>	Visualizza il listino dei prodotti del bar.
Input	Null.
Output	Fornisce una tabella con nome, codice, prezzo e disponibilità dei prodotti del bar. Inoltre è presente un campo che indica se il prodotto ha disponibilità illimitata.

NOME	Sessione Bar Gestore	
DESCRIZIONE	Accedono a questa classe quegli utenti che sono registrati come "Gestore del Bar". Permette di aprire una sessione per svolgere i compiti di gestione del bar.	
OPERAZIONI		
<b>valida()</b>		Serve a loggare il gestore del bar, per permettergli di effettuare modifiche ed inserimenti tra i prodotti del listino.
Input	Login e password.	
Output	Indicatore di successo.	
<b>inserisciProdotto()</b>		Inserisce un nuovo prodotto all'interno del listino.
Input	Nome, Codice, Prezzo, Disponibilità e Presenza illimitata del nuovo prodotto da inserire.	
Output	Indicatore di successo.	
<b>modificaProdotto()</b>		Modifica i dati di un prodotto già presente nel listino.
Input	Codice e tutti i dati da modificare del prodotto.	
Output	Indicatore di successo.	
<b>controllaProdotto()</b>		Controlla se sono stati compilati tutti i campi durante l'inserimento di un nuovo prodotto.
Input	Tutti i campi compilati al momento dell'inserimento di un nuovo prodotto.	
Output	True / False	

## Package Addetto Ristorante

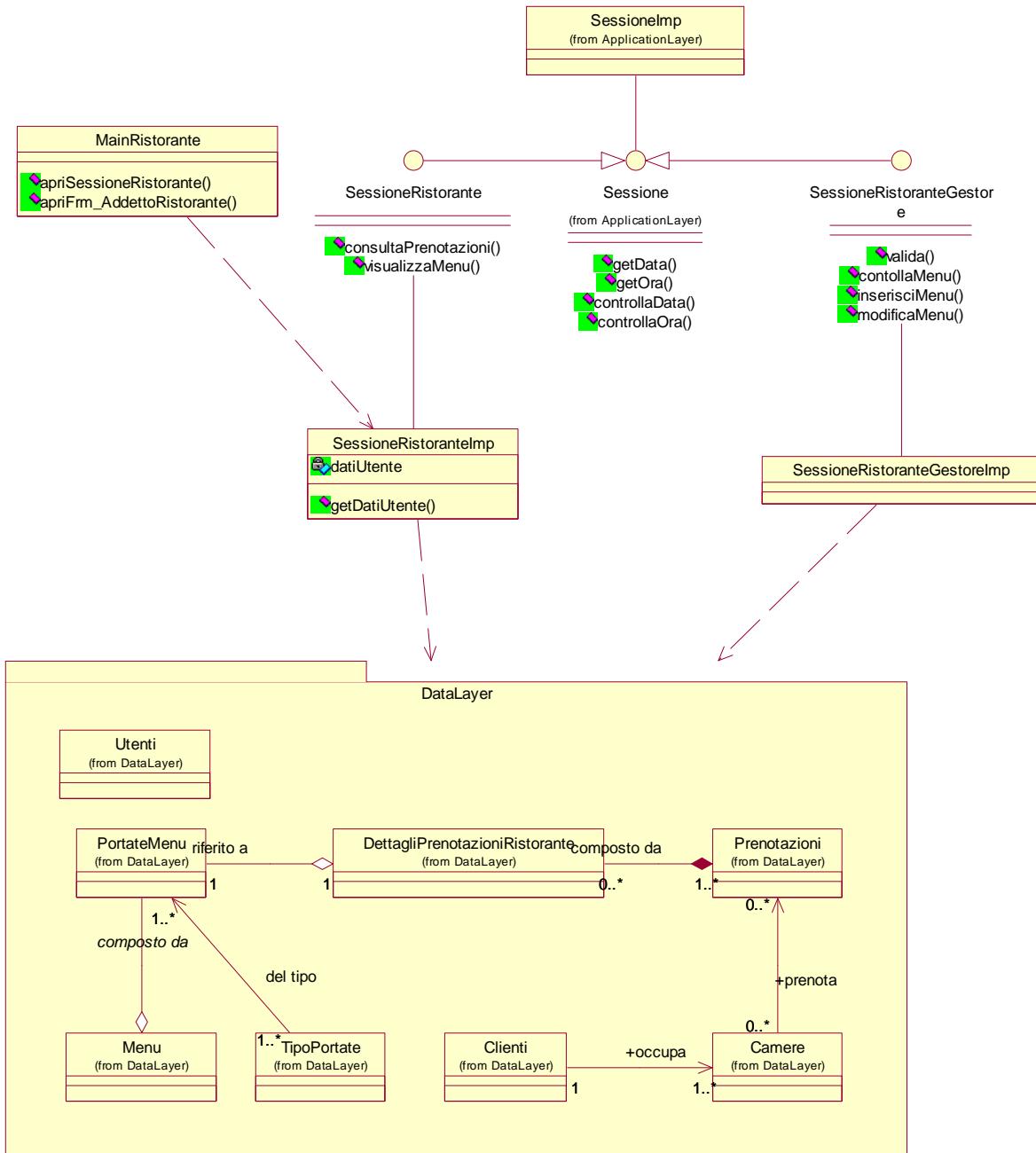


Figura 13: Package Addetto Ristorante

## Descrizione delle Classi

<b>Entity</b>	
<b>Main Ristorante</b>	Accedono a questa classe quegli utenti che sono registrati come “Addetto del Ristorante”. Permette di aprire una sessione per svolgere i compiti di gestione del ristorante.
<b>Boundary</b>	
<b>apriSessioneRistorante()</b>	Apre la sessione “Ristorante”
Input	Null.
Output	Indicatore di successo.
<b>ApriFrm_AddettoRistorante()</b>	Apre la form riservata all’ ”addetto Ristorante”.
Input	Null.
Output	Indicatore di successo.

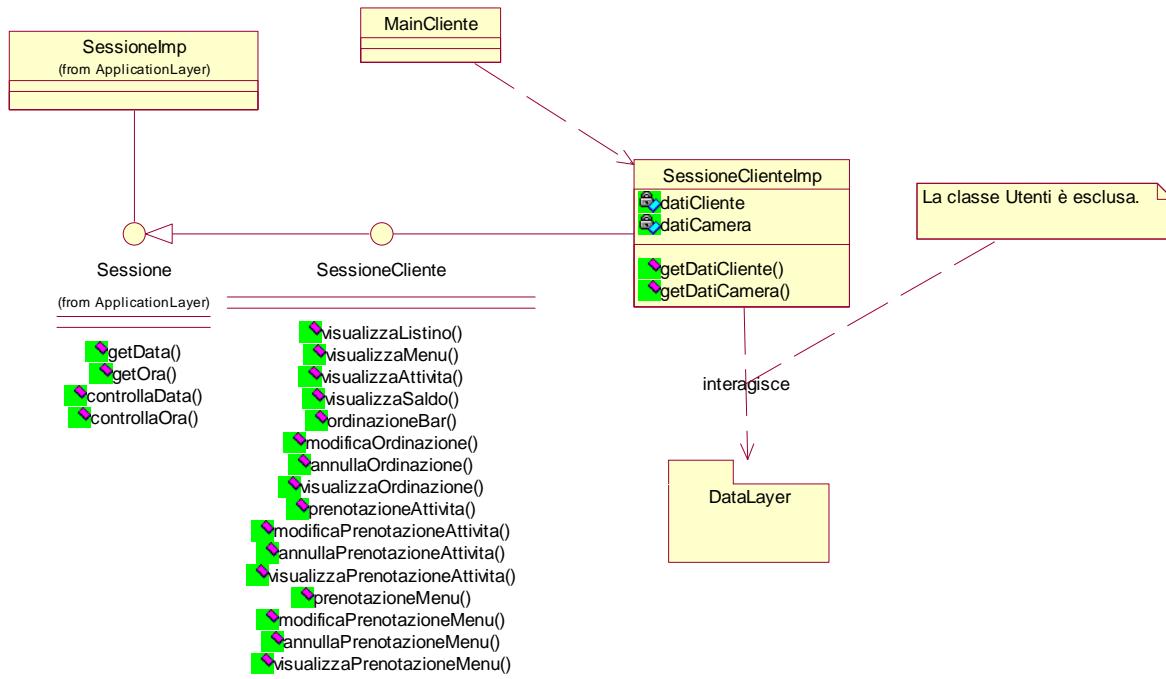
<b>NOME</b>	<b>Sessione Ristorante</b>
DESCRIZIONE	Accedono a questa classe quegli utenti che sono registrati come “Addetto del Ristorante”. Permette all’utente di controllare, inserire e modificare menu.
<b>OPERAZIONI</b>	
<b>consultaPrenotazioni()</b>	Fornisce una lista di tutte le prenotazioni effettuate al ristorante.
Input	Data delle prenotazioni da visualizzare specificando pranzo o cena
Output	Fornisce una tabella con data, ora, tipo (pranzo, cena, prodotto bar, attività extra) e costo totale della prenotazione.
<b>visualizzaMenu()</b>	Apre la form riservata all’ ”addetto Ristorante”.
Input	Prenotazione da visualizzare.
Output	Menu scelto dalla relativa prenotazione.

<b>NOME</b>	<b>Sessione Ristorante Gestore</b>
DESCRIZIONE	Accedono a questa classe quegli utenti che sono registrati come “Gestore del Ristorante”. Permette all’utente di consultare le prenotazioni effettuate con i relativi menu.
<b>OPERAZIONI</b>	
<b>valida()</b>	Permette l’accesso ad alcune informazioni riservate solo agli utenti registrati.
Input	Log-in e password dell’utente
Output	Indicatore di successo.

<b>controllaMenu()</b>	Permette di controllare il menu del giorno.
Input	Menu che si vuole controllare
Output	Menu scelto.
<b>inserisciMenu()</b>	Inserisce un nuovo menù all'interno dell'elenco dei menù.
Input	Data e tipo (pranzo o cena) del menù e/o descrizione, tipo (primo, secondo, contorno, ...), prezzo e disponibilità ("True" se disponibile, "False" altrimenti) delle portate del menù.
Output	Indicatore di successo.
<b>modificaMenu()</b>	Modifica un menù già presente in elenco.
Input	Tutti i dati da modificare del menù.
Output	Indicatore di successo.

NOME	Sessione RistoranteImp
DESCRIZIONE	Permette all'addetto Ristorante di conoscere i dati dell'utente
<b>OPERAZIONI</b>	
<b>getDatiUtente()</b>	Permette all'addetto Ristorante di conoscere i dati dell'utente che ha effettuato una prenotazione.
Input	Scelta dei dati da visualizzare
Output	Dati dell'utente.

## Package Cliente



**Figura 14: Package Cliente**

## Descrizione delle Classi

NOME	Sessione Cliente
DESCRIZIONE	Accedono a questa classe quegli utenti che sono “Clienti”. Permette di svolgere tutte le attività concesse al cliente della struttura. Tutte queste operazioni vengono svolte direttamente dalla camera del cliente.
<b>OPERAZIONI</b>	
<b>visualizzaListino()</b>	Visualizza il listino dei prodotti del bar.
Input	Null.
Output	Fornisce una tabella con nome, codice, prezzo e disponibilità dei prodotti del bar. Inoltre è presente un campo che indica se il prodotto ha disponibilità illimitata.
<b>visualizzaMenu()</b>	Fornisce l'elenco dei menù del ristorante.
Input	Null.
Output	Fornisce una tabella con data e tipo (pranzo o cena) e le portate del menù.
<b>viasualizzaAttivita()</b>	Visualizza l'elenco delle attività disponibili al momento presso la struttura.
Input	Null.
Output	Fornisce una tabella con nome, descrizione, data, prezzo, posti disponibili e opzioni speciali delle attività extra.
<b>visualizzaSaldo()</b>	Fornisce il saldo del cliente presso la struttura.
Input	Numero camera.
Output	Fornisce un valore reale che corrisponde al saldo di colui che richiede la funzione.
<b>ordinazioneBar()</b>	Serve per ordinare un prodotto al bar.
Input	Codice, quantità e note del prodotto da ordinare al bar.
Output	Indicatore di successo
<b>modificaOrdinazione()</b>	Serve per modificare una ordinazione al bar effettuata in precedenza.
Input	Codice e tutti i dati da modificare dell'ordinazione.
Output	Indicatore di successo
<b>annullaOrdinazione()</b>	Serve per disdire una ordinazione effettuata al bar in precedenza.
Input	Codice dell'ordinazione.
Output	Indicatore di successo
<b>visualizzaOrdinazione()</b>	Serve a visualizzare tutte le ordinazioni effettuate.
Input	Null
Output	Fornisce una tabella contenente le ordinazioni effettuate con codice, nome, prezzo e quantità del prodotto ordinato.
<b>prenotazioneMenu()</b>	Serve per prenotare un menù al ristorante.
Input	Codice, data, tipo (pranzo o cena), note e codice delle portate scelte del menù.
Output	Indicatore di successo.

<b>modificaPrenotazioneMenu()</b>	Serve per modificare una prenotazione al ristorante effettuata in precedenza.
Input	Codice e tutti i dati da modificare della prenotazione.
Output	Indicatore di successo
<b>annullaPrenotazioneMenu()</b>	Serve per disdire una prenotazione effettuata al ristorante in precedenza.
Input	Codice della prenotazione.
Output	Indicatore di successo
<b>visualizzaPrenotazioneMenu()</b>	Serve a visualizzare tutte le prenotazioni effettuate al ristorante.
Input	Null
Output	Fornisce una tabella contenente le prenotazioni effettuate con codice, data, tipo (pranzo o cena), note della prenotazione effettuata al ristorante .
<b>prenotazioneAttivita()</b>	Serve per prenotare una attività extra.
Input	Codice, numero adulti, numero bambini e note dell’attività extra da prenotare.
Output	Indicatore di successo
<b>modificaPrenotazioneAttivita()</b>	Serve per modificare una prenotazione delle attività extra effettuata in precedenza.
Input	Codice e tutti i dati da modificare della prenotazione.
Output	Indicatore di successo
<b>annullaPrenotazioneAttivita()</b>	Serve per disdire una prenotazione effettuata alle attività extra in precedenza.
Input	Codice dell’attività extra.
Output	Indicatore di successo
<b>visualizzazionePrenotazioni()</b>	Fornisce un elenco di tutte le prenotazioni effettuate in precedenza.
Input	Null.
Output	Codice, numero adulti, numero bambini e note dell’attività extra prenotate.

<b>NOME</b>	<b>Sessione</b>
DESCRIZIONE	Fornisce l’interfaccia comune a tutte le sessioni definite nell’applicazione.
<b>OPERAZIONI</b>	
<b>getOra()</b>	Visualizza l’ora.
Input	Null.
Output	Restituisce l’ora
<b>getData()</b>	Visualizza la data.
Input	Null.
Output	Restituisce la data
<b>controllaOra()</b>	Serve al thread per controllare se è possibile modificare una ordinazione
Input	Ora in cui è stata effettuata la prenotazione

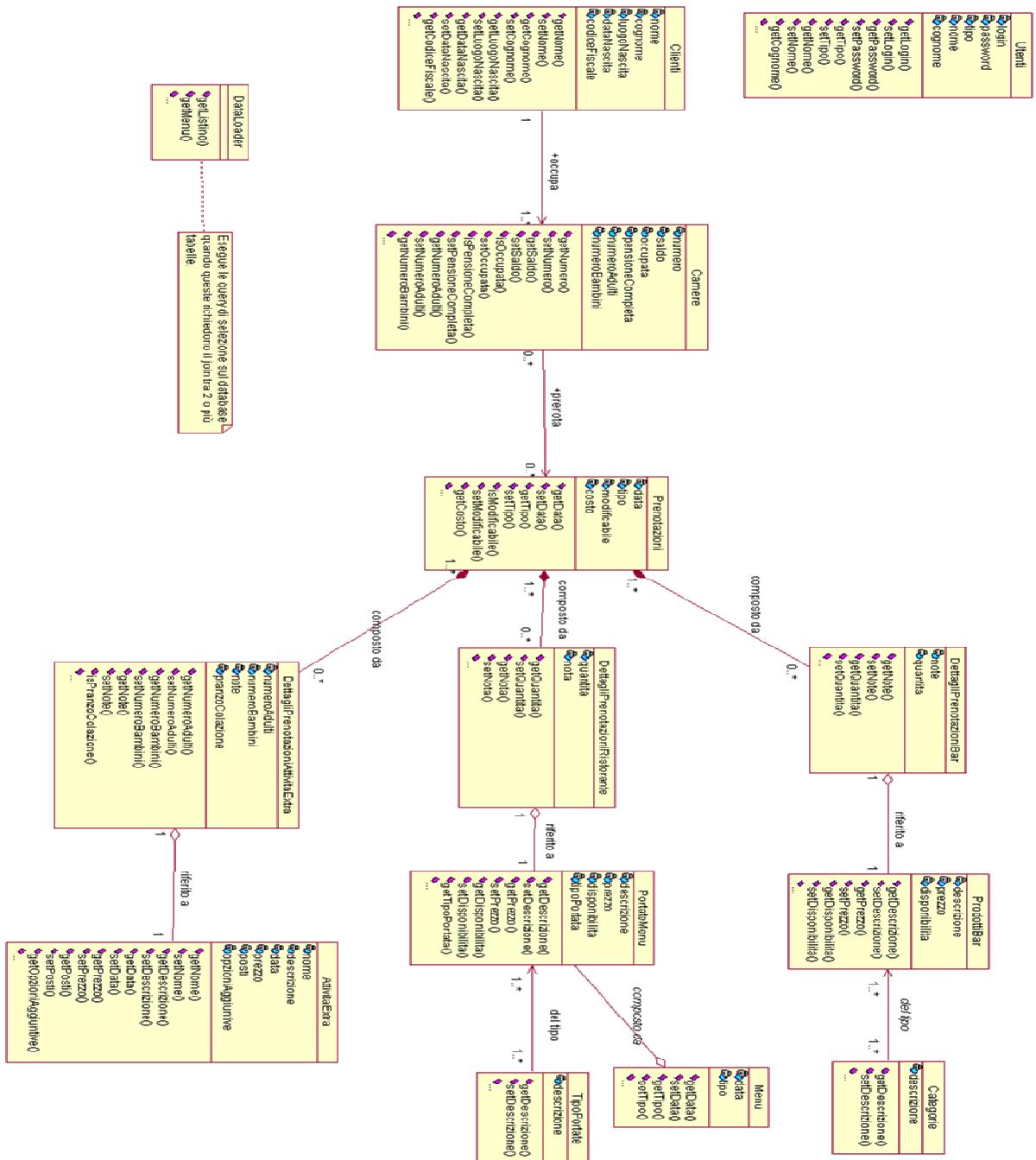
Output	Null
<b>controllaData()</b>	Serve al thread per controllare se è possibile modificare una ordinazione
Input	Data in cui è stata effettuata la prenotazione
Output	Null

NOME	Sessione ClienteImp
DESCRIZIONE	E' la classe che permette ai clienti di usufruire dei servizi messi a disposizione dalla struttura nel suo insieme.
<b>OPERAZIONI</b>	
<b>getDatiCliente()</b>	Restituisce i dati relativi al cliente
Input	Id camera
Output	Tutti i dati del cliente
<b>getDatiCamera()</b>	Restituisce i dati relativi alla camera
Input	Id camera
Output	Tutti i dati della camera

### **3.5.3.2 Livello dati**

Dato che il nostro sistema è basato su un’architettura three-layer, il presente diagramma delle classi è legato alla parte più vicina ai dati della nostra astrazione.

La funzionalità principale è quella di far riconoscere le classi e i metodi che interagiscono col DataBase, ed illustrare come i dati sono accessibili.



**Figura 15:** Livello dati

## Descrizione delle Classi

<b>NOME</b>	<b>Cliente</b>
<b>DESCRIZIONE</b>	Contiene le informazioni sui clienti che accedono al sistema per effettuare la prenotazione e consente anche la modifica e la creazione di clienti.
<b>OPERAZIONI</b>	
<b>LoadCliente()</b>	Carica i dati dell'utente dal DataBase.
Input	Codice dell'utente
Output	Dati dell'utente
<b>CreaCliente()</b>	Crea un nuovo utente
Input	Dati dell'utente
Output	Codice dell'utente
<b>DeleteCliente()</b>	Elimina un utente
Input	Codice dell'utente
Output	Indicatore di successo
<b>UpdateCliente()</b>	Aggiorna i dati di un utente
Input	Codice dell'utente ed ulteriori campi
Output	Indicatore di successo
<b>StorageCliente()</b>	Memorizza i dati dell'utente nel DataBase.
Input	Codice dell'utente e attributi associati all'utente
Output	Indicatore di successo
<b>GetCliente()</b>	Visualizza i dati anagrafici di un utente.
Input	Nessuno
Output	Dati utente
<b>SetCliente()</b>	Inserisce nuovi dati anagrafici per l'utente.
Input	Codice e dati anagrafici dell'utente.
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Nome</b>	Nome dell'utente
<b>Cognome</b>	Cognome dell'utente
<b>DNascita</b>	Data di nascita dell'utente
<b>CodiceFiscale</b>	Codice fiscale dell'utente
<b>LuogoDiNascita</b>	Luogo di nascita dell'utente
<b>newAttr</b>	Memorizza un nuovo attributo

<b>NOME</b>	<b>Camera</b>
<b>DESCRIZIONE</b>	Contiene le informazioni della camera con dati riguardanti i numeri di posti disponibili e sul suo stato (occupata o no)
<b>OPERAZIONI</b>	
<b>LoadCamera()</b>	Carica i dati della camera dal DataBase
Input	Codice della camera
Output	Indicatore di successo
<b>NewOperation()</b>	Permette di creare una nuova operazione
Input	Codice dell'operazione
Output	Risultato dell'operazione
<b>GetCamera()</b>	Visualizza i dati della camera
Input	Nessuno
Output	Dati della camera
<b>SetCamera()</b>	Modifica i dati della camera
Input	Dati da modificare
Output	Nessuno
<b>GetOccupata()</b>	Ritorna lo stato della camera cioè indica se la camera è occupata da un cliente oppure è libera
Input	Nessuno
Output	Stato della camera
<b>SetOccupata()</b>	Modifica lo stato della camera riferito al fatto se è occupata oppure no
Input	Stato della camera da modificare (valore true o false)
Output	Nessuno
<b>UpdateCamera()</b>	Oggiorna i dati della camera
Input	Codice camera
Output	Indicatore di successo
<b>StorageCamera()</b>	Memorizza i dati della camera nel DataBase
Input	Codice della camera
Output	Indicatore di successo
<b>GetPensioneCompleta()</b>	Ritorna lo stato della pensione completa
Input	Nessuno
Output	Stato della pensione completa
<b>SetPensioneCompleta()</b>	Modifica lo stato della pensione completa
Input	Stato (valore true o false)
Output	Nessuno
<b>GetNAdulti()</b>	Ritorna il numero di adulti disponibile nella camera
Input	Nessuno
Output	Ritorna numero adulti
<b>SetNAdulti()</b>	Modifica il numero di adulti disponibile nella camera
Input	Numero adulti
Output	Nessuno
<b>GetNBambini()</b>	Ritorna il numero di bambini disponibile nella camera
Input	Nessuno
Output	Ritorna numero bambini
<b>SetNBambini()</b>	Modifica il numero di bambini disponibile nella camera
Input	Numero bambini
Output	Nessuno

### ATTRIBUTI

<b>Saldo</b>	Contiene il saldo della camera
<b>PensioneCompleta</b>	Riferimento alla pensione completa
<b>Occupata</b>	Riferimento sullo stato della camera
<b>NAdulti</b>	Contiene il numero di adulti contenuti nella camera
<b>NBambini</b>	Contiene il numero di bambini contenuti nella camera

<b>NOME</b>	<b>Prenotazione</b>
DESCRIZIONE	Consente ad un cliente di effettuare le prenotazioni di: attivita extra, bar, ristorante
<b>OPERAZIONI</b>	
<b>LoadPrenotazione()</b>	Carica i dati della prenotazione dal database
Input	Codice Prenotazione
Output	Indicatore di successo
<b>CreaPrenotazione()</b>	Crea nuova prenotazione
Input	Dati della prenotazione
Output	Indicatore di successo
<b>DeletePrenotazione()</b>	Cancella una Prenotazione
Input	Codice Prenotazione
Output	Indicatore di successo
<b>StoragePrenotazione()</b>	Memorizza nel database una nuova prenotazione
Input	Codice della prenotazione
Output	Indicatore di successo
<b>UpdatePrenotazione()</b>	Aggiorna i dati di una prenotazione
Input	Codice Prenotazione
Output	Indicatore di successo
<b>GetPrenotazione()</b>	Mostra i dati di una prenotazione
Input	Nessuno
Output	Dati prenotazione
<b>SetPrenotazione()</b>	Modifica i dati di una prenotazione
Input	Dati da modificare
Output	Nessuno
<b>GetCostoTotale()</b>	Mostra il costo totale di una prenotazione
Input	Nessuno
Output	Dati prenotazione
<b>SetCostoTotale()</b>	Modifica il costo di una prenotazione
Input	Nuovo costo
Output	Nessuno
<b>GetData()</b>	Ritorna la data della prenotazione
Input	Nessuno
Output	Data
<b>SetData()</b>	Modifica la data della prenotazione
Input	Data
Output	Nessuno
<b>GetOra()</b>	Ritorna l'ora della prenotazione
Input	Nessuno
Output	l'ora
<b>SetOra()</b>	Modifica l'ora della prenotazione
Input	l'ora
Output	Nessuno
<b>NewOperation()</b>	Inserisce una nuova operazione
Input	Dati operazione
Output	Indicatore di successo

ATTRIBUTI	
<b>Data</b>	Memorizza la data della prenotazione
<b>Costo</b>	Memorizza il costo della prenotazione
<b>Stato</b>	Memorizza lo stato della prenotazione ed è un attributo specificato solo per le ordinazione al bar; vale 1 (Attiva) se lo stato è Attivo e quindi ancora deve essere passato al bar per essere soddisfatta l'ordinazione; vale 2 (Passiva) se è già passata al bar e quindi sta per essere soddisfatta; vale 3 (Modifica) se il cliente la sta modificando.
<b>Ora</b>	Memorizza l'ora della prenotazione

<b>NOME</b>	<b>Menu</b>
DESCRIZIONE	Serve per mostrare un menu formato dalla data, dal tipo (cena o pranzo)
<b>OPERAZIONI</b>	
<b>LoadMenu()</b>	Carica i dati del menù dal database
Input	Codice del menù
Output	Indicatore di successo
<b>CreaMenu()</b>	Crea un nuovo menu
Input	Dati da memorizzare
Output	Indicatore di successo
<b>DeleteMenu()</b>	Cancella un menù
Input	Codice del menù
Output	Indicatore di successo
<b>StorageMenu()</b>	Memorizza i dati del menù nel database
Input	Codice menù
Output	Indicatore di successo
<b>UpdateMenu()</b>	Aggiorna il menù
Input	Codice menu
Output	Indicatore di successo
<b>GetMenu()</b>	Ritorna i dati del menù
Input	Nessuno
Output	Dati del menù
<b>SetMenu()</b>	Modifica i dati del menu
Input	Dati da modificare
Output	Nessuno
<b>GetData()</b>	Ritorna la data del menù
Input	Nessuno
Output	Data
<b>SetData()</b>	Modifica la data del menù
Input	Data
Output	Nessuno
<b>GetTipo()</b>	Ritorna il tipo di menù
Input	Nessuno
Output	Tipo di menù
<b>SetTipo()</b>	Modifica il valore che riguarda il tipo di menù
Input	Tipo di menù
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Date</b>	Data del menù
<b>Tipo</b>	Valore che indica il tipo di menù

<b>NOME</b>	<b>PortateMenu</b>
<b>DESCRIZIONE</b>	Collegato alla classe menù, serve per mostrare le singole portate che formano il menù
<b>OPERAZIONI</b>	
<b>LoadPortateMenu()</b>	Carica i dati delle portate dal database
Input	Codice della portata
Output	Indicatore di successo
<b>CreaPortateMenu()</b>	Crea una nuova portata
Input	Codice e riferimento al tipo di menù
Output	Indicatore di successo
<b>DeletePortateMenu()</b>	Cancella una portata
Input	Codice della portata
Output	Indicatore di successo
<b>StoragePortateMenu()</b>	Memorizza le portate nel database
Input	Dati da memorizzare
Output	Indicatore di successo
<b>UpdatePortateMenu()</b>	Aggiorna le portate
Input	Codice portata
Output	Indicatore di successo
<b>GetDisponibilità()</b>	Ritorna un valore che indica la disponibilità del prodotto
Input	Nessuno
Output	Valore true o false che indica la disponibilità
<b>SetDisponibilità()</b>	Modifica il valore che indica la disponibilità
Input	Valore true o false
Output	Nessuno
<b>GetPortateMenu()</b>	Mostra i dati riguardanti le portate
Input	Nessuno
Output	Dati portate
<b>SetPortateMenu()</b>	Modifica i dati delle portate
Input	Dati da modificare
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Descrizione</b>	Memorizza la descrizione della portata
<b>Prezzo</b>	Prezzo della portata
<b>Tipo</b>	Tipologia della portata
<b>Disponibilità</b>	Valore booleano che indica la disponibilità della portata

<b>NOME</b>	<b>DettaglioPrenotazioneAttivitaExtra</b>
<b>DESCRIZIONE</b>	Serve per mostrare i dettagli di una singola attività includendo dati come la disponibilità dei posti
<b>OPERAZIONI</b>	
<b>LoadDettaglioExtra()</b>	Carica dal database i dati delle attività extra
Input	Codice attività
Output	Indicatore di successo
<b>CreaDettaglioExtra()</b>	Crea una nuova attività
Input	Dati da memorizzare
Output	Indicatore di successo
<b>DeleteDettaglioExtra()</b>	Cancella un attività
Input	Codice
Output	Indicatore di successo
<b>StorageDettaglioExtra()</b>	Memorizza nel database l'attività
Input	Codice Attività
Output	Indicatore di successo
<b>UpdateDettaglioExtra()</b>	Aggiorna l'attività
Input	Codice attività
Output	Indicatore di successo
<b>GetDettaglioExtra()</b>	Ritorna le informazioni sull'attività
Input	Nessuno
Output	Dati attività
<b>SetDettaglioExtra()</b>	Modifica i dati sull'attività
Input	Dati da modificare
Output	Nessuno
<b>GetNAdulti()</b>	Ritorna il numero degli adulti partecipanti
Input	Nessuno
Output	Numero adulti
<b>SetNAdulti()</b>	Modifica il numero degli adulti
Input	Numero adulti
Output	Nessuno
<b>GetNBambini()</b>	Ritorna il numero di bambini partecipanti
Input	nessuno
Output	Numero bambini
<b>SetNBambini()</b>	Modifica il numero di bambini partecipanti
Input	Numero bambini
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Nadulti</b>	Numero di adulti partecipanti all'attività
<b>NBambini</b>	Numero di bambini partecipanti all'attività
<b>Note</b>	Note sul tipo di attività

<b>NOME</b>	<b>DettaglioPrenotazioneRistorante</b>
<b>DESCRIZIONE</b>	Gestisce i dettagli riguardanti le prenotazioni del ristorante
<b>OPERAZIONI</b>	
<b>LoadDettaglioRistorante()</b>	Carica il dettaglio delle prenotazioni al ristorante da DataBase
Input	Codice dettaglio
Output	Indicatore di successo
<b>CreaDettaglioRistorante()</b>	
Input	Dati del dettaglio prenotazione
Output	Indicatore di successo
<b>DeleteDettaglioRistorante()</b>	Cancella dettaglio di prenotazione
Input	Codice
Output	Indicatore di successo
<b>StorageDettaglioRistorante()</b>	Memorizza nel database i dati del dettaglio prenotazione ristorante
Input	Codice
Output	Indicatore di successo
<b>UpdateDettaglioRistorante()</b>	Aggiorna un dettaglio di prenotazione
Input	Codice
Output	Indicatore di successo
<b>GetDettaglioRistorante()</b>	Ritorna dati di prenotazione
Input	Nessuno
Output	Dati da visualizzare
<b>SetDettaglioRistorante()</b>	Modifica dati di prenotazione
Input	Dati da memorizzare
Output	Nessuno
<b>GetQuantita()</b>	Ritorna la quantità
Input	Nessuno
Output	Dati da visualizzare
<b>SetQuantita()</b>	Modifica la quantità
Input	Dati da memorizzare
Output	Nessuno
<b>GetData()</b>	Ritorna la data
Input	Nessuno
Output	Dati da visualizzare
<b>SetData()</b>	Modifica la data
Input	Dati da memorizzare
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Quantita</b>	Memorizza la quantità delle portate
<b>Note</b>	Memorizza varie note

<b>NOME</b>	<b>DettaglioPrenotazioneBar</b>
<b>DESCRIZIONE</b>	Mostra e gestisce le varie prenotazioni che si possono effettuare al bar
<b>OPERAZIONI</b>	
<b>LoadDettaglioBar()</b>	Carica le prenotazioni del bar dal Database
Input	Codice della prenotazione del bar
Output	Indicatore di successo
<b>CreaDettaglioBar()</b>	Crea una nuova prenotazione del bar
Input	Dati da memorizzare
Output	Indicatore di successo
<b>DeleteDettaglioBar()</b>	Cancella una prenotazione del bar
Input	Codice
Output	Indicatore di successo
<b>StorageDettaglioBar()</b>	Memorizza una prenotazione del bar direttamente nel DataBase
Input	Codice
Output	Indicatore di successo
<b>UpdateDettaglioBar()</b>	Aggiorna i dati delle prenotazioni del bar
Input	Codice
Output	Indicatore di successo
<b>GetDettaglioBar()</b>	Ritorna i valori della prenotazione
Input	Nessuno
Output	Dati da visualizzare
<b>SetDettaglioBar()</b>	modifica i valori della prenotazione
Input	Dati da memorizzare
Output	Nessuno
<b>GetQuantita()</b>	Ritorna la quantità
Input	Nessuno
Output	Dati da visualizzare
<b>SetQuantita()</b>	Modifica la quantità
Input	Dati da memorizzare
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Note</b>	Memorizza eventuali note
<b>Quantità</b>	Memorizza la quantità

<b>NOME</b>	<b>ProdottoBar</b>
<b>DESCRIZIONE</b>	Mostra i dettagli dei singoli prodotti del bar con la disponibilità e il prezzo
<b>OPERAZIONI</b>	
<b>LoadProdotto()</b>	Carica il prodotto del bar dal DataBase
Input	Codice della prenotazione del bar
Output	Indicatore di successo
<b>CreaProdotto()</b>	Crea una nuovo il prodotto del bar
Input	Dati da memorizzare
Output	Indicatore di successo
<b>DeleteProdotto()</b>	Cancella una il prodotto del bar
Input	Codice
Output	Indicatore di successo
<b>StorageProdotto()</b>	Memorizza un prodotto del bar direttamente nel DataBase
Input	Codice
Output	Indicatore di successo
<b>UpdateProdotto()</b>	Aggiorna i dati del prodotto del bar
Input	Codice
Output	Indicatore di successo
<b>GetProdotto()</b>	Ritorna i dati del prodotto del bar
Input	Nessuno
Output	Dati da visualizzare
<b>SetProdotto()</b>	Modifica i dati del prodotto del bar
Input	Dati da memorizzare
Output	Nessuno
<b>GetDisponibilità()</b>	Ritorna la disponibilità del prodotto
Input	Nessuno
Output	Dati da visualizzare
<b>SetDisponibilità()</b>	Modifica la disponibilità del prodotto
Input	Dati da memorizzare
Output	Nessuno
<b>GetPrezzo()</b>	Ritorna il prezzo del prodotto
Input	Nessuno
Output	Dati da visualizzare
<b>SetPrezzo()</b>	Modifica il prezzo del prodotto
Input	Dati da memorizzare
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Nome</b>	Memorizza il nome del prodotto del bar
<b>Prezzo</b>	Memorizza il prezzo
<b>Disponibilità</b>	Memorizza lo stato della disponibilità

<b>NOME</b>	<b>Categoria</b>
DESCRIZIONE	Serve per elencare le varie categorie che formano il bar (per esempio: bibite, liquori, ecc.)
<b>OPERAZIONI</b>	
<b>LoadCategoria()</b>	Carica una categoria dal database
Input	Codice della categoria
Output	Indicatore di successo
<b>CreaCategoria()</b>	Crea una nuova categoria
Input	Dati da memorizzare
Output	Indicatore di successo
<b>DeleteCategoria()</b>	Cancella una categoria
Input	Codice
Output	Indicatore di successo
<b>StorageCategoria()</b>	Memorizza una categoria nel database
Input	Codice
Output	Indicatore di successo
<b>UpdateCategoria()</b>	Aggiorna una categoria
Input	Codice
Output	Indicatore di successo
<b>GetCategoria()</b>	Ritorna il nome della categoria
Input	Nessuno
Output	Categoria
<b>SetCategoria()</b>	Modifica il nome della categoria
Input	Nome categoria
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Descrizione</b>	Memorizza la descrizione della categoria

<b>NOME</b>	<b>Utente</b>
<b>DESCRIZIONE</b>	Serve per memorizzare le registrazioni di vari addetti che gestiscono il sistema di prenotazione
<b>OPERAZIONI</b>	
<b>LoadUtente()</b>	Carica i dati di accesso di un utente
Input	Login e Password dell'utente
Output	Indicatore di successo
<b>GetNome()</b>	Ritorna il nome dell'utente
Input	Nessuno
Output	Dati da visualizzare
<b>GetCognome()</b>	Ritorna il cognome dell'utente
Input	Nessuno
Output	Dati da visualizzare
<b>ATTRIBUTI</b>	
<b>Login</b>	Username dell'utente
<b>Password</b>	Password dell'utente
<b>Tipo</b>	Indica il tipo di utente cioè: gestore, addetto al ristorante, addetto al bar
<b>Nome</b>	Nome utente
<b>Cognome</b>	Cognome utente

<b>NOME</b>	<b>AttivitàExtra</b>
DESCRIZIONE	Mostra le varie attività extra dell'albergo come per esempio gite escursioni ecc.
<b>OPERAZIONI</b>	
<b>LoadAttivita()</b>	Carica i dati di un attività extra dal database
Input	Codice dell'attività
Output	Indicatore di successo
<b>CreaAttivita()</b>	Crea una nuova attività
Input	Dati da memorizzare
Output	Indicatore di successo
<b>DeleteAttivita()</b>	Cancella una attività
Input	Codice
Output	Indicatore di successo
<b>StorageAttivita()</b>	Memorizza un'attività nel database
Input	Codice
Output	Indicatore di successo
<b>UpdateAttivita()</b>	Aggiorna un'attività
Input	Codice
Output	Indicatore di successo
<b>GetAttivita()</b>	Ritorna i valori dell'attività
Input	Nessuno
Output	Dati da visualizzare
<b>SetAttivita()</b>	Modifica i valori dell'attività
Input	Dati da modificare
Output	Nessuno
<b>GetPrezzo()</b>	Ritorna il prezzo
Input	Nessuno
Output	Dati da visualizzare
<b>SetPrezzo()</b>	Modifica il prezzo
Input	Dati da modificare
Output	Nessuno
<b>GetPosti()</b>	Ritorna i posti disponibili
Input	Nessuno
Output	Dati da visualizzare
<b>SetPosti()</b>	modifica i posti disponibili
Input	Dati da modificare
Output	Nessuno
<b>IncreasePosti()</b>	Incrementa i posti Disponibili
Input	Numero di incremento
Output	Nessuno
<b>DecreasePosti()</b>	decrementa i posti Disponibili
Input	Numero di decremento
Output	Nessuno
<b>ATTRIBUTI</b>	
<b>Nome</b>	Memorizza il nome dell'attività
<b>Descrizione</b>	Memorizza la descrizione dell'attività
<b>Data</b>	Memorizza la data in cui si svolgerà l'attività
<b>Prezzo</b>	Memorizza il prezzo

<b>PostiDisponibili</b>	Mantiene il numero di posti disponibili per poter partecipare
<b>OpzioniSpeciali</b>	Serve per memorizzare le eventuali opzioni speciali secondo il tipo di attività

## Servizio ristorante

- **Entity:**

Cliente: contiene le informazioni sui clienti che accedono al sistema per effettuare la prenotazione al ristorante.

Addetto ristorante: permette all'utente di controllare, inserire e modificare menu.

- **Boundary:**

Annullamento ordinazione: serve per disdire una ordinazione effettuata al ristorante in precedenza.

Visualizza ordinazione: serve a visualizzare tutte le ordinazioni effettuate.

Modifica ordinazione: serve per modificare una ordinazione al ristorante effettuata in precedenza.

Prenotazione menù: serve per prenotare un menù.

Visualizza saldo: permette di visualizzare il saldo attuale.

Visualizza menù: fornisce una lista relativa ai menù disponibili.

Modifica menù: modifica un menù già presente in elenco.

Inserimento menù: permette di aggiungere o eliminare menù.

Consultazione prenotazione: fornisce una lista di tutte le prenotazioni effettuate al ristorante.

- **Control:**

Valida: questo oggetto è creato quando il gestore richiede l'inserimento, aggiornamento e cancellazione dei menù e gestisce i form d'immissione dei dati.

## Attività extra:

- **Entity:**

Gestore: permette all'utente di gestire le attività extra.

Cliente: permette al cliente di visualizzare le attività extra.

- **Boundary:**

Annullamento prenotazioni: permette al cliente di annullare un'attività prenotata.

Visualizza prenotazione: permette al cliente di visualizzare le attività prenotate.

Modifica prenotazione: permette al cliente di modificare una prenotazione.

Nuova prenotazione: permette al cliente di prenotare l'attività desiderata.

Resoconto prenotazione: permette al gestore di visualizzare il resoconto di tutti i prenotati ad una certa attività.

Modifica attività: permette al gestore di modificare le attività.

Cancellazione attività: permette al gestore di eliminare un'attività.

Inserimento attività: permette al gestore di inserire una nuova attività.

- **Control:**

Valida: questo oggetto è creato quando il gestore richiede l'inserimento, aggiornamento e cancellazione dell'attività extra e gestisce i form d'immissione dei dati.

## Servizio bar

- **Entity:**

Cliente: contiene le informazioni sui clienti che accedono al sistema per effettuare la prenotazione al bar.

Addetto bar: permette all'utente di controllare, inserire e modificare i prodotti.

- **Boundary:**

Visualizza ordinazione: serve a visualizzare tutte le ordinazioni effettuate.

Modifica ordinazione: serve per modificare una ordinazione al bar effettuata in precedenza.

Nuovo ordine: serve per gestire un nuovo ordine.

Cancella ordine: serve per disdire una ordinazione effettuata al bar in precedenza.

Visualizza listino: contiene i prezzi relativi ad ogni prodotto.

Inserimento prodotti: serve per inserire dei nuovi prodotti.

Modifica prodotti: serve per modificare prodotti già esistenti.

Cancella prodotti: serve per eliminare un prodotto esistente.

- **Control:**

Valida: questo oggetto è creato quando il gestore richiede l'inserimento, aggiornamento e cancellazione dei prodotti e gestisce i form d'immissione dei dati.

## Servizio camere

- **Entity:**

Gestore: permette la gestione completa sulle camere dell' albergo.

- **Boundary:**

Libera camera: permette di liberare le camere.

Visualizza camera: visualizzare le informazioni riguardanti le camere dell'albergo.

Modifica camera: permette al gestore di modificare la prenotazione delle camere.

Assegna camera: permette al gestore di assegnare le camere.

- **Control:**

Valida: questo oggetto è creato quando il gestore richiede l'inserimento, aggiornamento e cancellazione delle camere e gestisce i form d'immissione dei dati.

### 3.5.4 Modello dinamico

#### Diagramma di attività

##### Bar : Nuova Ordinazione

Questa funzione permette al cliente di effettuare una nuova ordinazione al bar. Effettuare una ordinazione significa selezionare un prodotto e definire la quantità e le eventuali note del prodotto da ordinare.

Il diagramma ci mostra la comunicazione tra cliente, sistema e addetto bar. Dopo aver invocato la funzione, il sistema risponde con la visualizzazione del listino. Il cliente seleziona i prodotti da ordinare e invia l'ordinazione. Il sistema salva i dati, invia l'ordine all'addetto bar e visualizza un messaggio di avvenuta ordinazione.

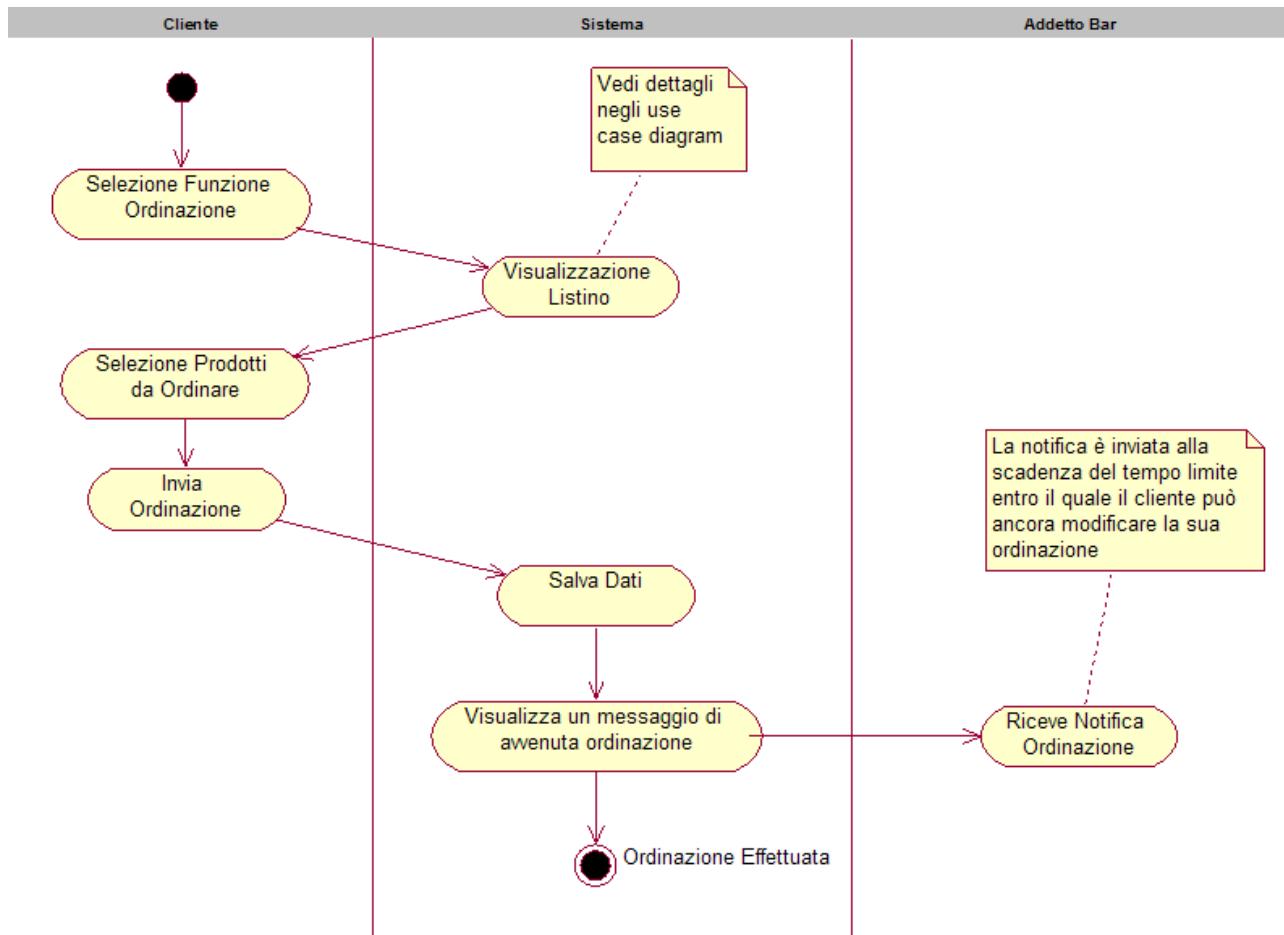


Figura 16: Diagramma di attività: Bar: Nuova Ordinazione

## Ristorante : Inserimento Menù

Questa funzione permette al gestore del ristorante di inserire un nuovo menù. Per effettuare tale operazione è necessario predisporre dei permessi di accesso alla gestione del ristorante, quindi loggarsi.

Il diagramma ci mostra la comunicazione tra gestore e sistema. Dopo aver invocato la funzione, il sistema risponde con la visualizzazione della maschera di login. Viene poi chiesto al gestore di scegliere la data del menù da inserire. Vengono effettuati i controlli sulla validità della data inserita, dopodiché viene visualizzata la form per l'inserimento dei dati del menù. Il gestore compila i campi, il sistema verifica la correttezza e la completezza dei dati dopodiché li salva.

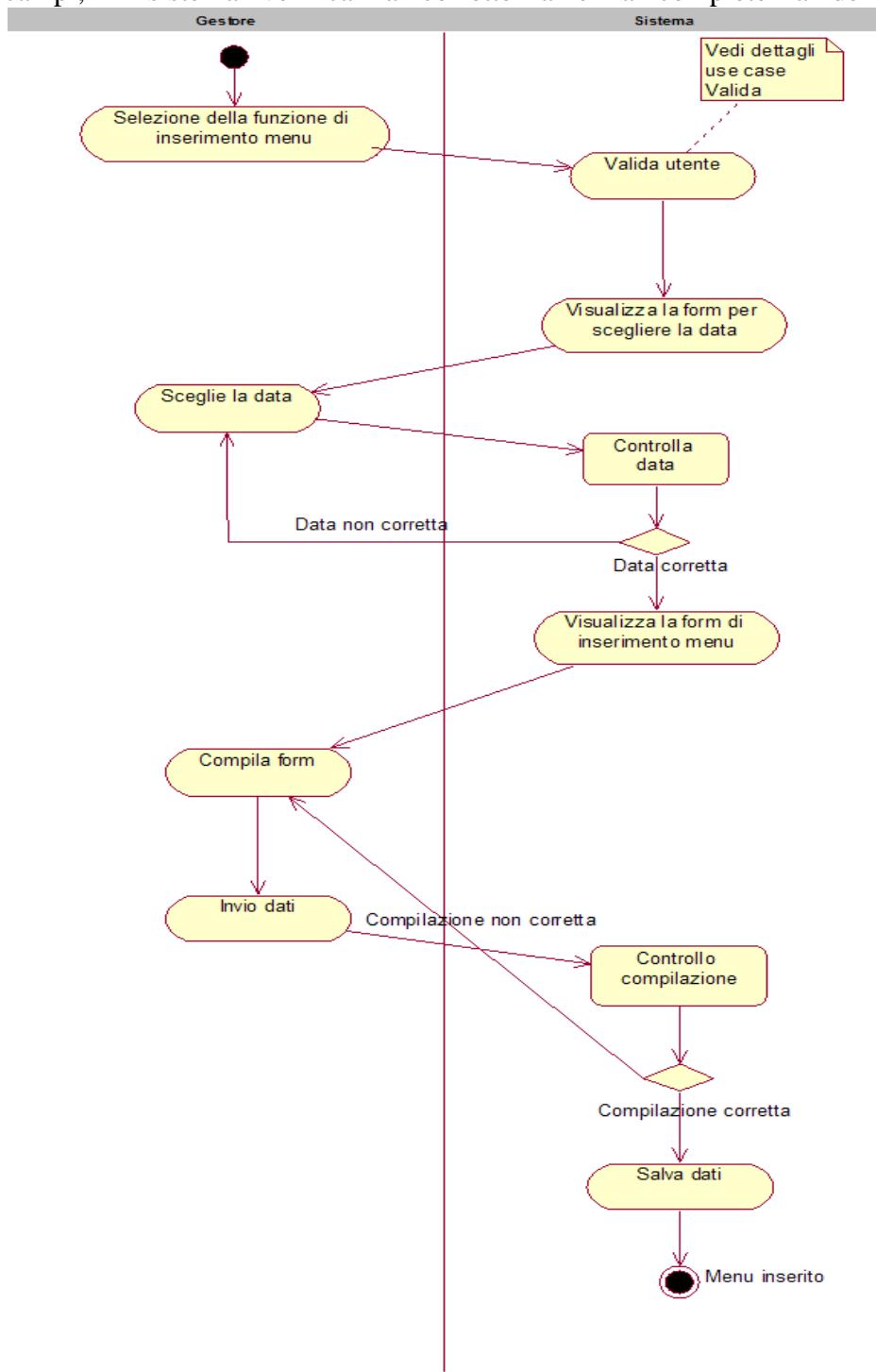


Figura 17: Diagramma di attività: Ristorante: Inserimento Menù

## Gestione Camere : Assegna Camera

Questa funzione permette al gestore delle camere di assegnare una camera ad un cliente. Per effettuare tale operazione è necessario predisporre dei permessi di accesso alla gestione delle camere, quindi il gestore deve essersi in precedenza loggato.

Il diagramma ci mostra la comunicazione tra gestore e sistema. Dopo aver invocato la funzione, il sistema risponde con la visualizzazione delle camere. Nel caso che la struttura non abbia camere libere, il sistema mostra un messaggio al gestore e termina il caso d'uso. Nel caso opposto viene invece visualizzato un elenco delle camere libere, tra cui il gestore effettua la scelta della camera. Viene poi chiesto al gestore di compilare il form con le informazioni del cliente. Vengono controllati i dati e di conseguenza salvati.

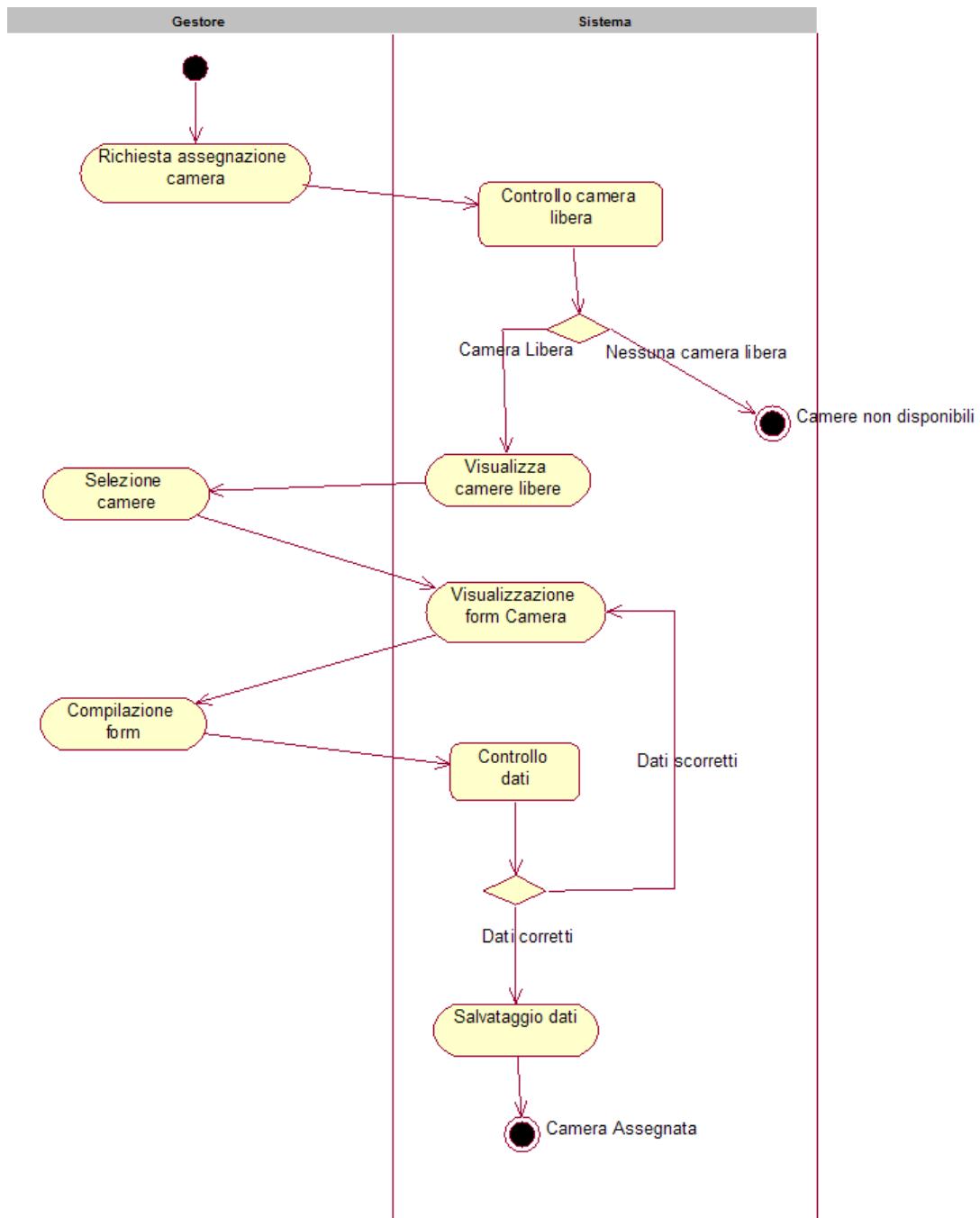


Figura 18: Diagramma di attività: Gestione Camere - Assegna Camera

## Gestione Attività Extra : Cancellazione Attività

Questa funzione permette al gestore delle attività extra di cancellare una attività dall'elenco delle attività disponibili. Per effettuare tale operazione è necessario predisporre dei permessi di accesso alla gestione del ristorante, quindi il gestore deve essersi in precedenza loggato.

Il diagramma ci mostra la comunicazione tra gestore e sistema. Dopo aver invocato la funzione, il sistema risponde con la visualizzazione delle attività disponibili. Viene poi chiesto al gestore di scegliere e confermare l'attività da cancellare. Il sistema effettua la cancellazione dell'attività, salva i dati e visualizza un messaggio al gestore di avvenuta cancellazione.

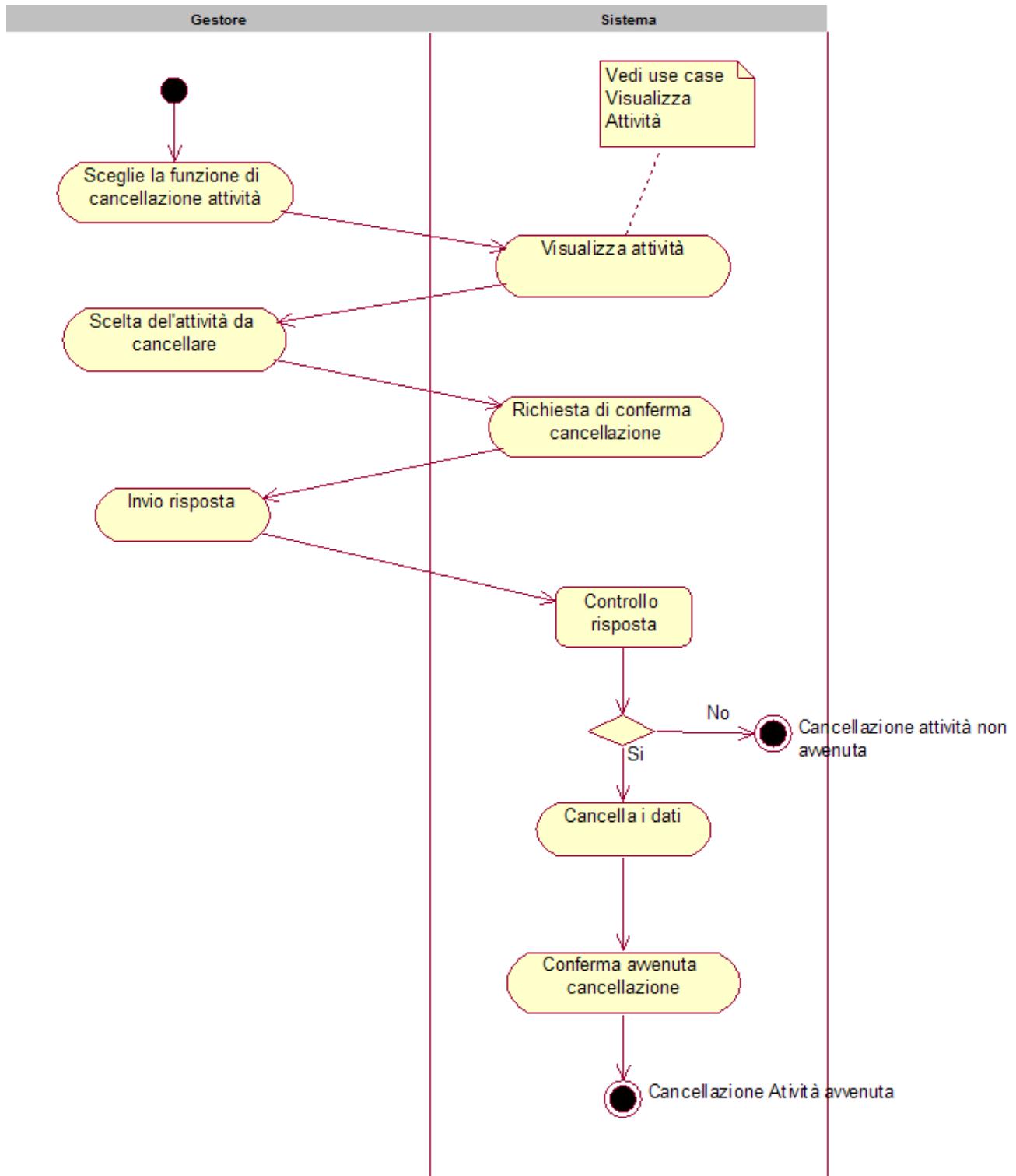


Figura 19: Diagramma di attività: Attività Extra - Cancellazione Attività

## Bar : Modifica Ordinazione

Questa funzione permette al cliente di modificare una ordinazione precedentemente effettuata. Per effettuare tale operazione è necessario aver effettuato l'ordinazione di almeno un prodotto.

Il diagramma ci mostra la comunicazione tra cliente e sistema. Dopo aver invocato la funzione, il sistema in caso che siano state effettuate delle prenotazioni, risponde con la visualizzazione della maschera delle prenotazioni fatte dal cliente. Nel caso contrario, viene terminato il caso d'uso. Tra le prenotazioni effettuate il cliente sceglie l'ordinazione da modificare, il sistema controlla se il tempo utile alla modifica dell'ordinazione non è ancora scaduto dopodiché mostra al cliente l'ordinazione fatta in precedenza. Il cliente modifica l'ordinazione e conferma la modifica. Il sistema salva i dati ed invia un messaggio di avvenuta modifica al cliente.

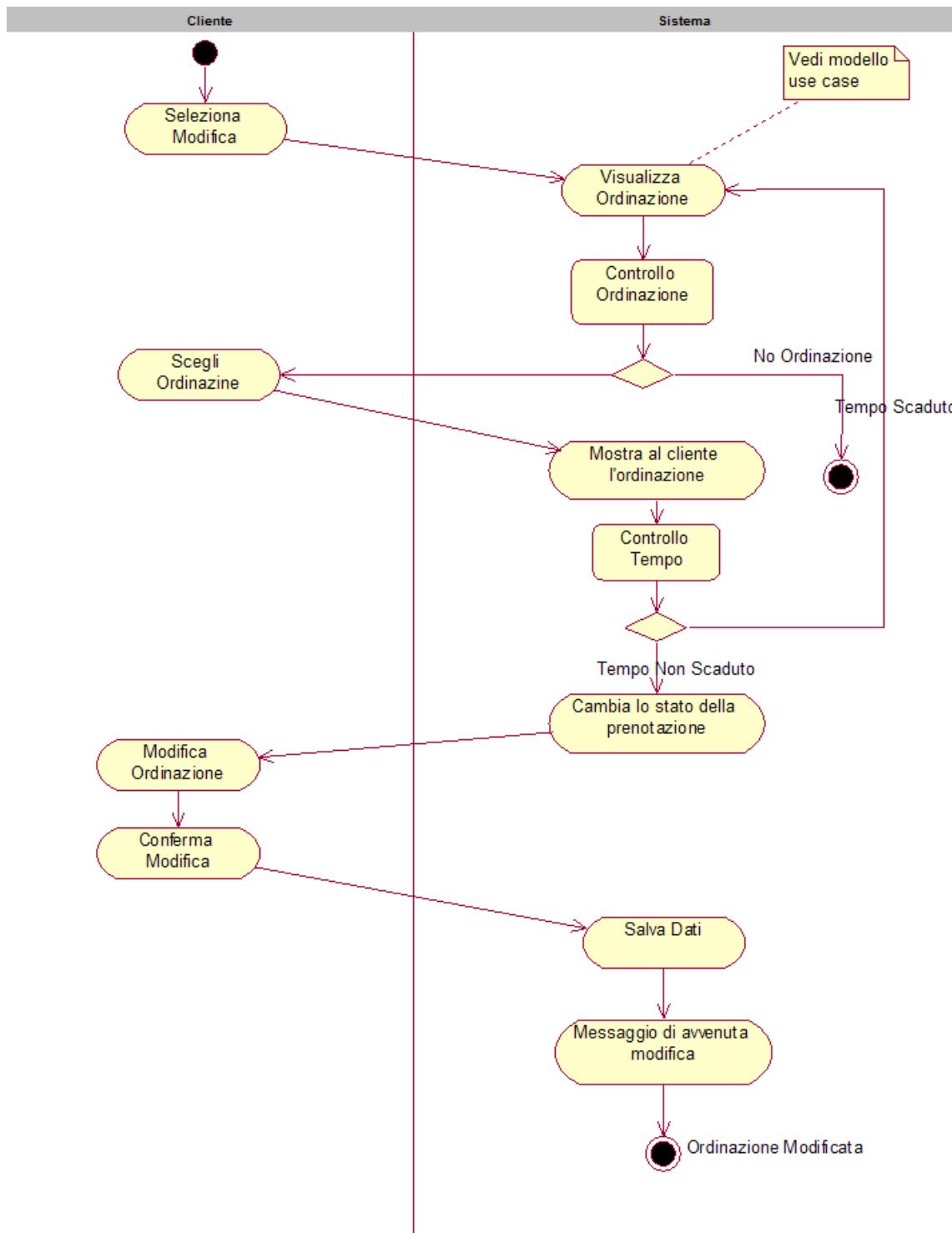


Figura 20: Diagramma di attività: Bar – Modifica Ordinazione

### Bar : Inserimento Prodotto (nel Listino)

Questa funzione permette al gestore del bar di inserire un nuovo prodotto in listino. Per effettuare tale operazione è necessario predisporre dei permessi di accesso alla gestione del bar, quindi il gestore deve essersi in precedenza loggato.

Il diagramma ci mostra la comunicazione tra gestore e sistema. Dopo aver invocato la funzione, il sistema mostra una form per l'inserimento del nuovo prodotto. Il sistema controlla i campi inseriti dal gestore (se i dati sono corretti, completi e se già esiste un prodotto con la descrizione inserita). Dopo ciò il sistema aggiunge il prodotto e salva i dati, in caso di dati errati consente al gestore di inserire di nuovo i dati.

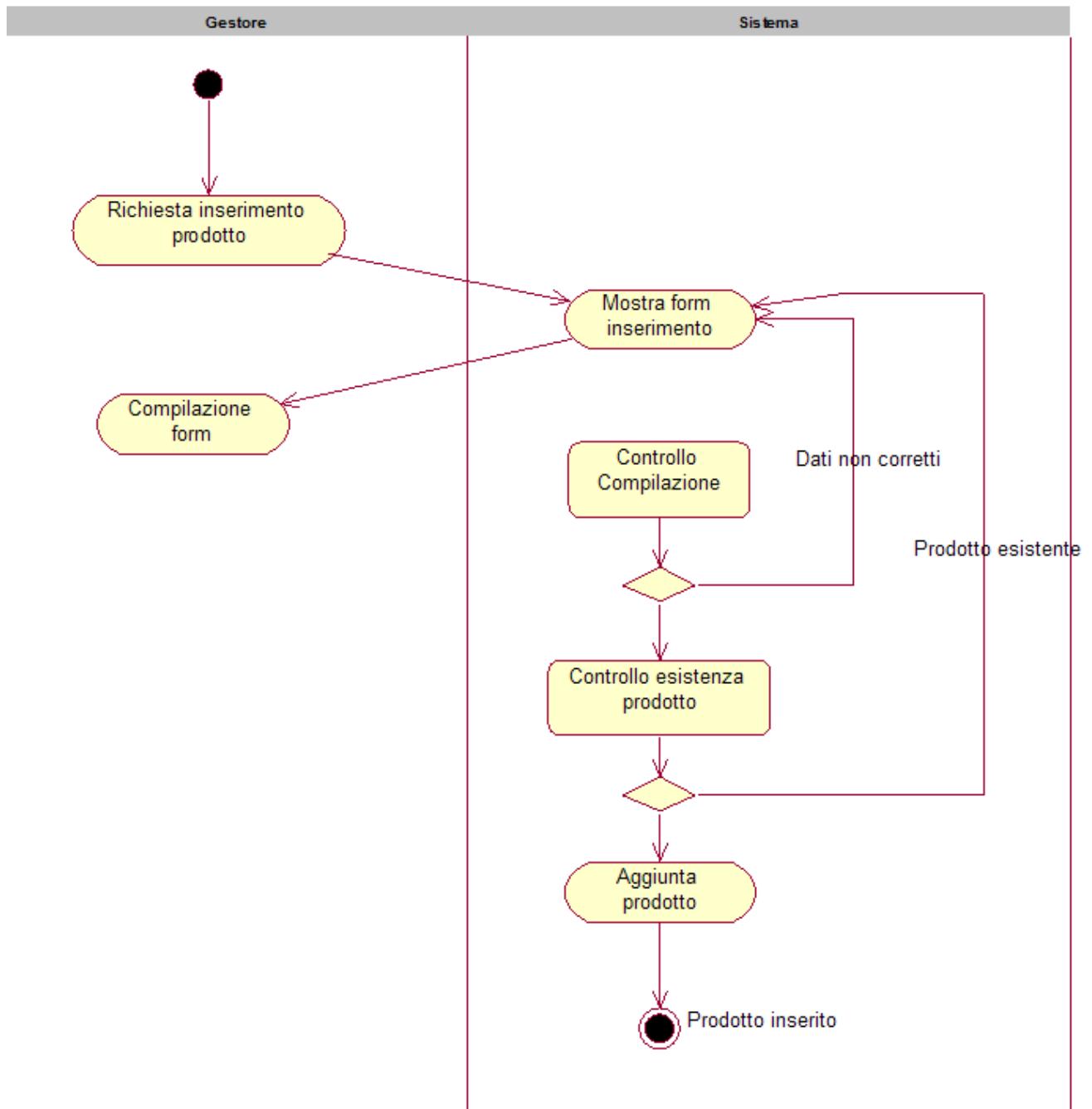


Figura 21: Diagramma di attività: Bar – Inserimento Prodotto

### Gestione Camere : Libera Camera

Questa funzione permette al gestore delle camere di liberare una camera (fine del soggiorno del cliente). Per effettuare tale operazione è necessario predisporre dei permessi di accesso alla gestione delle camere, quindi il gestore deve essersi in precedenza loggato.

Il diagramma ci mostra la comunicazione tra gestore e sistema. Dopo aver invocato la funzione, il sistema visualizza le camere occupate e chiede al gestore di selezionare la camera da liberare. Il sistema visualizza un messaggio di richiesta di conferma della liberazione, stampando il saldo della camera e salvando i dati.

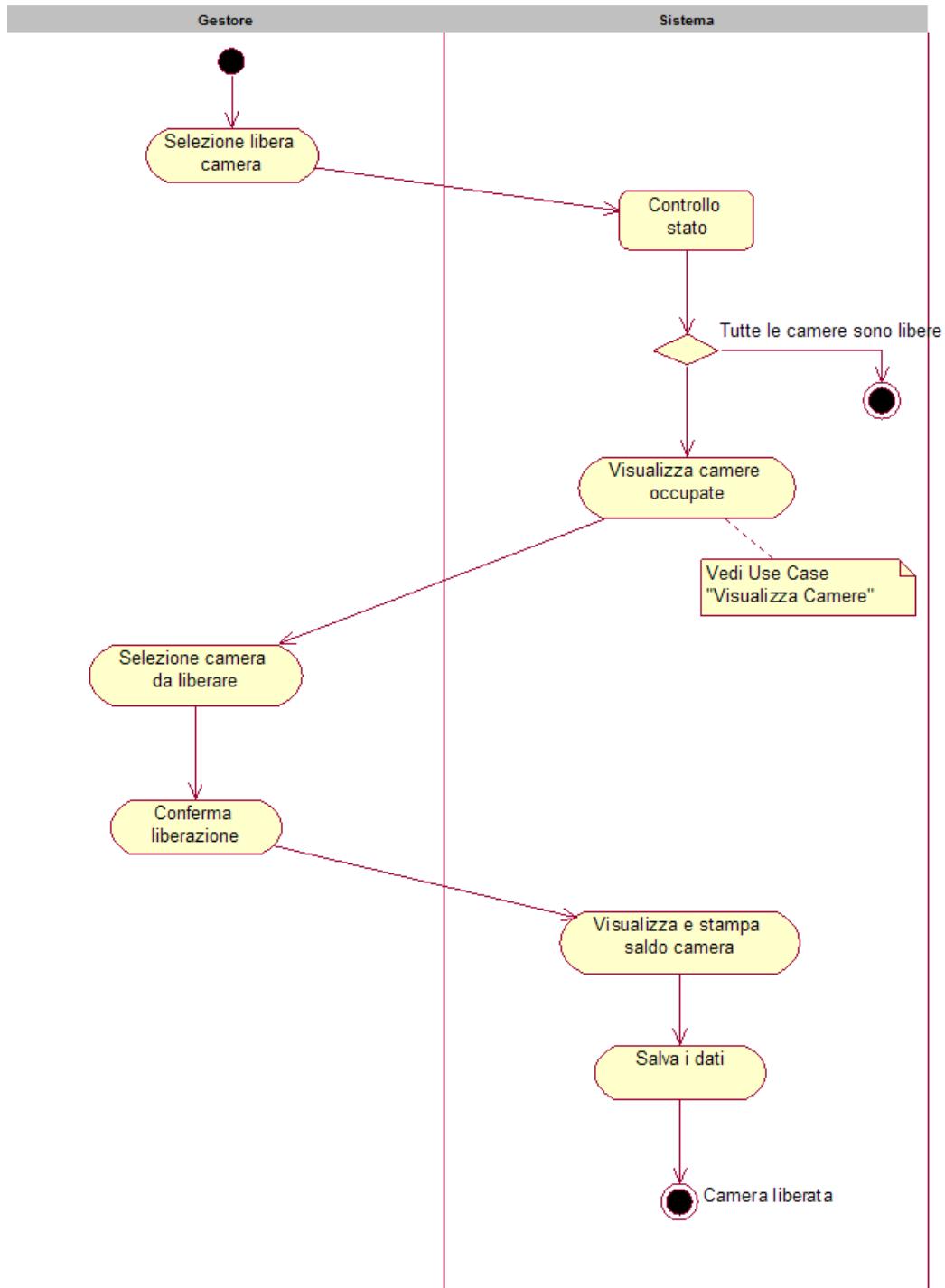


Figura 22: Diagramma di attività: Gestione Camere– Libera Camera

## Ristorante : Prenotazione Menù

Questa funzione permette al cliente di effettuare una prenotazione al ristorante. Effettuare una prenotazione al ristorante significa selezionare un primo, un secondo e un contorno e definire le quantità e le eventuali note del menù che si vuole ordinare.

Il diagramma ci mostra la comunicazione tra cliente e sistema. Dopo aver invocato la funzione, il sistema mostra una schermata che richiede al cliente se vuole prenotare per il pranzo o per la cena e la data della prenotazione. Il sistema controlla la correttezza della data, dopodiché controlla se è ancora possibile prenotare per quel tipo di menù. Il cliente sceglie le portate, il sistema effettua il controllo dei dati inseriti, salva i dati e mostra un messaggio al cliente di avvenuta prenotazione.

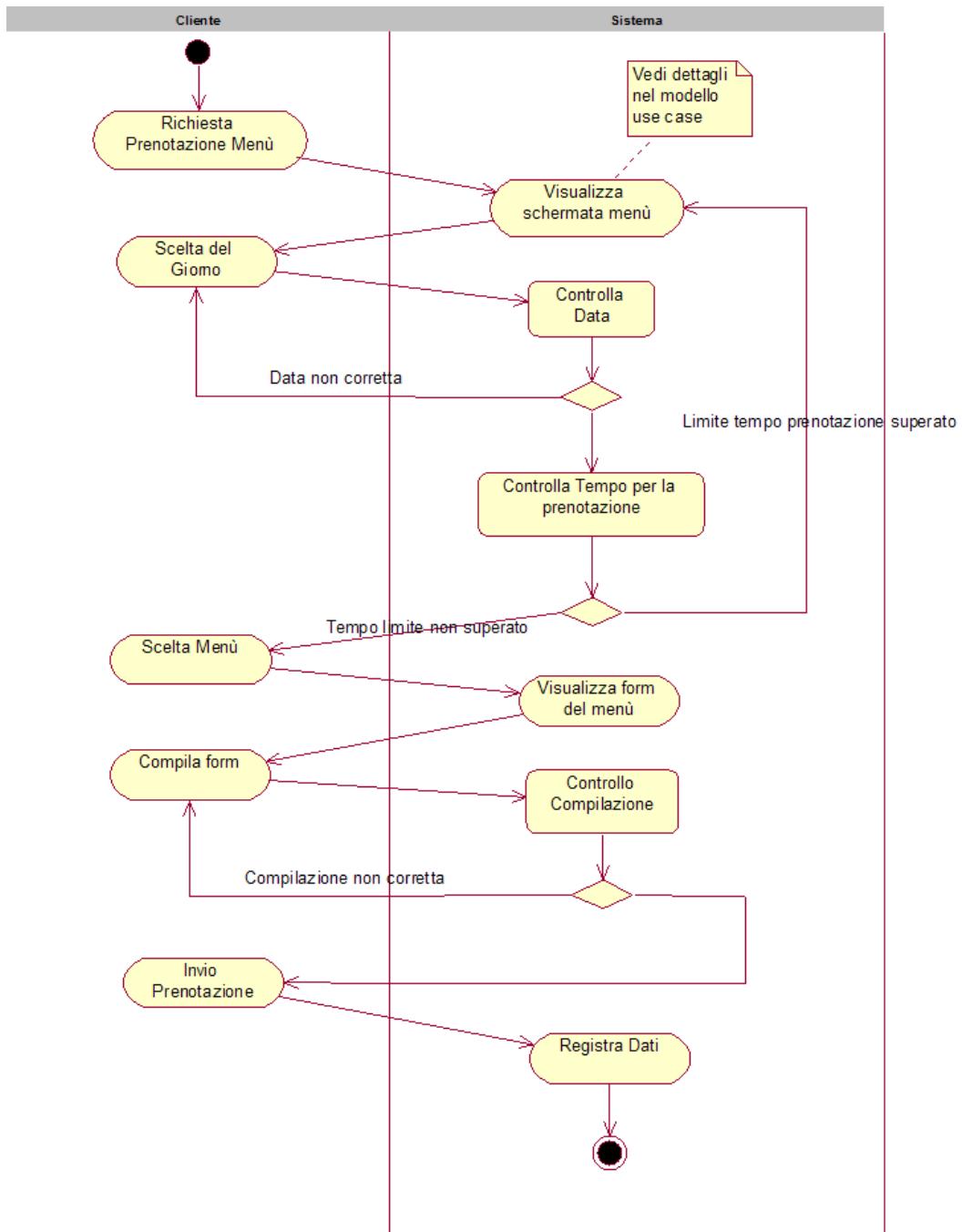


Figura 23: Diagramma di attività: Ristorante - Prenotazione Menù

## Gestione Attività Extra : Nuova Prenotazione

Questa funzione permette al cliente di prenotare una attività extra. Effettuare tale prenotazione significa selezionare una attività, il numero di posti per adulti e bambini e le note che si richiedono. Il diagramma ci mostra la comunicazione tra cliente e sistema. Dopo aver invocato la funzione, il sistema mostra una schermata che visualizza le attività che sono disponibili, il cliente seleziona l’attività che vuole svolgere e compila la form. Il sistema controlla la correttezza dei dati salva i dati e invia un messaggio al cliente di avvenuta prenotazione.

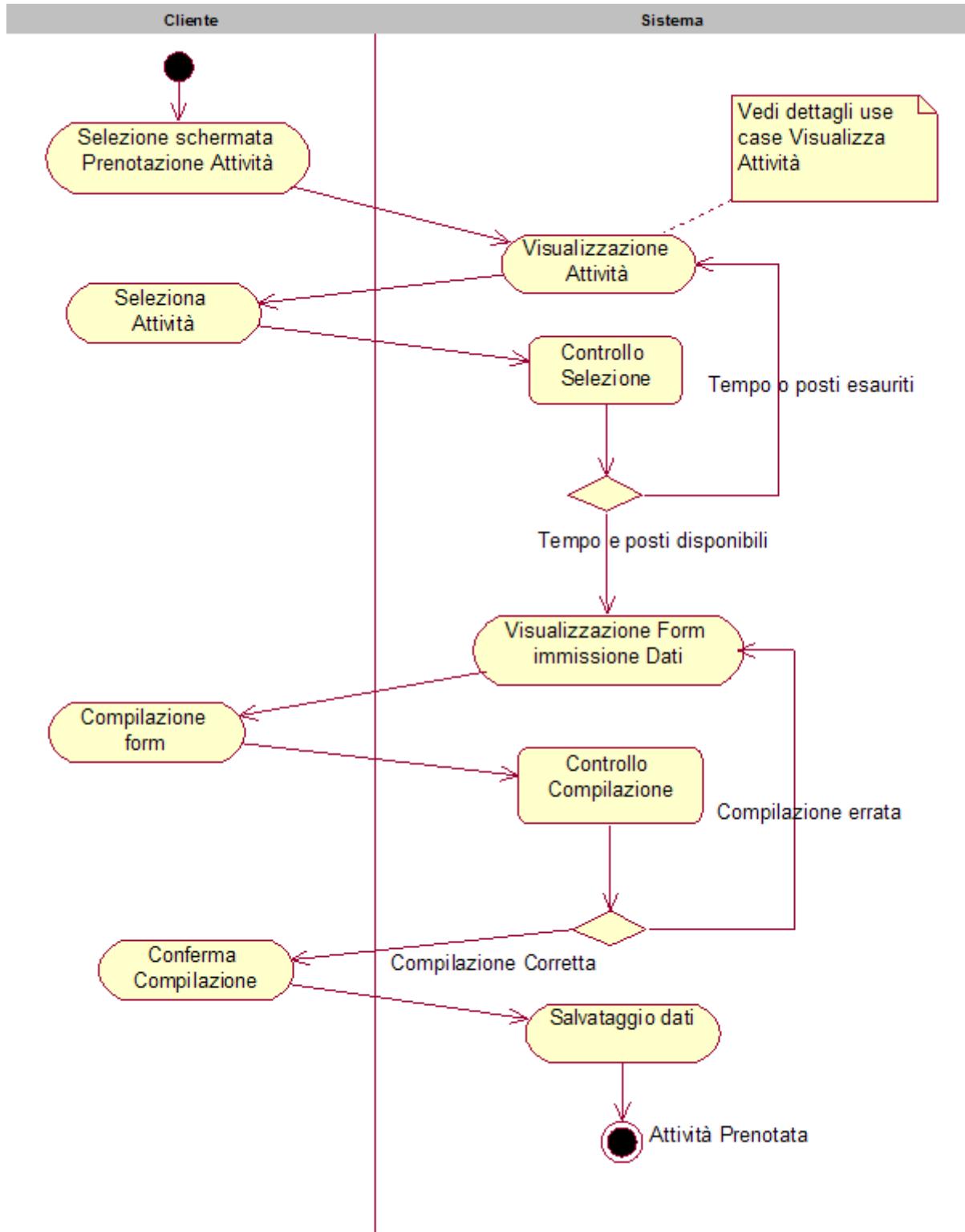


Figura 24: Diagramma di attività: Attività Extra – Nuova Prenotazione

## Diagramma di sequenza

Nei diagrammi che seguono la classe Livello dati è una classe fittizia che offre dei metodi per accedere al risultato di interrogazioni verso il database che restituiscono grosse quantità di dati.

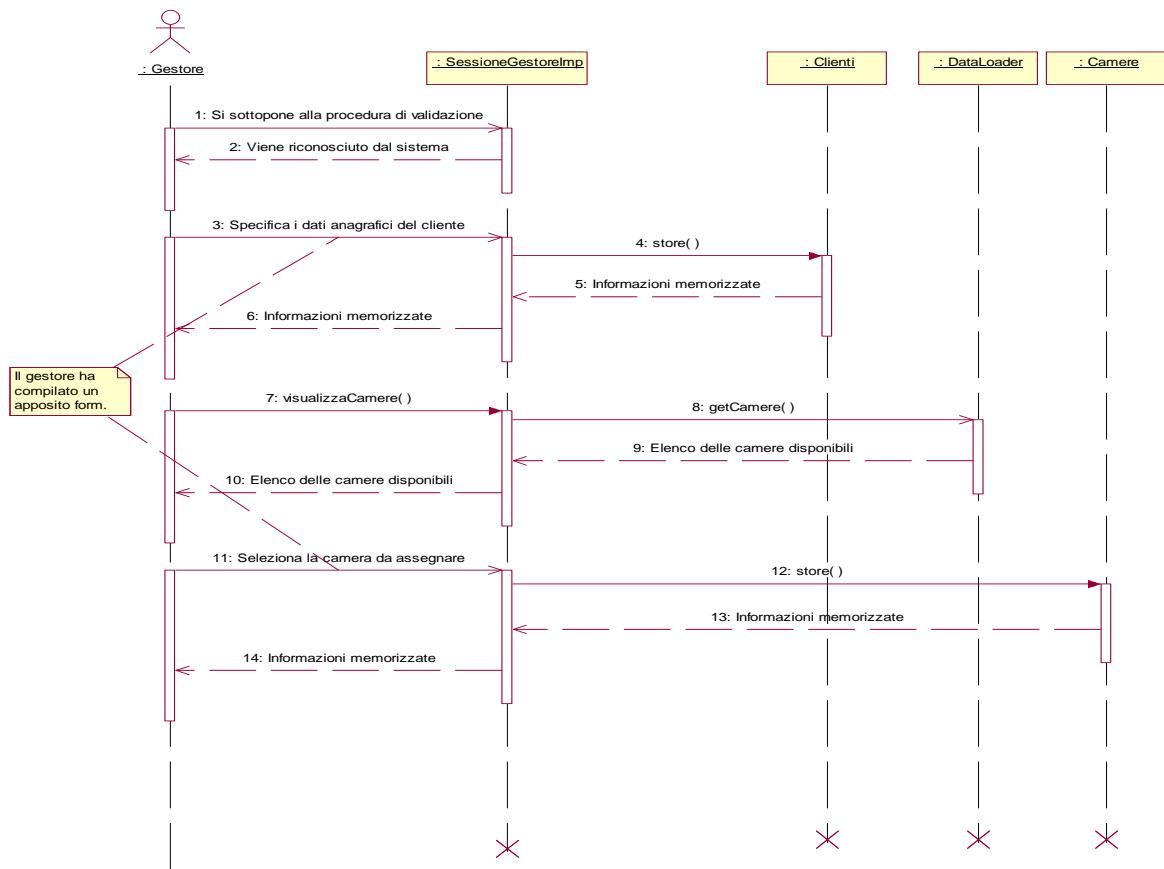
### Gestione Camere: Assegna Camere

Questa procedura consente al responsabile della reception dell'albergo di assegnare le camere ai clienti. Essendo una procedura gestionale è richiesta l'autenticazione per poterne usufruire. Il responsabile può visualizzare l'elenco completo delle camere non ancora occupate e, se tra quelle camere disponibili ce n'è una che soddisfa le esigenze del cliente, procedere alla compilazione di un form dove vanno specificati i dati anagrafici del cliente e la modalità di assegnazione (mezza pensione o pensione completa) e l'eventuale durata.

L'istanza della classe SessioneGestore consente al responsabile di aprire una nuova sessione nel sistema e accedere alle funzionalità riservate alla classe di utenti registrati come "Gestore". Per aprire una nuova sessione è necessario sottoporsi alla procedura di autenticazione fornendo il nome utente e la password per tale account. La procedura di autenticazione è realizzata dalla routine Valida() che prende in input le informazioni sopra citate e ne controlla l'autenticità.

Se l'autenticazione va a buon fine il gestore può aggiungere un nuovo cliente al database specificando le sue informazioni anagrafiche. L'istanza della classe Cliente, creata invocando il comando CreaCliente(), si occuperà della memorizzazione effettiva di tali informazioni mediante i comandi SetCliente() e StorageCliente() e restituirà il codice che identifica univocamente il nuovo cliente nel sistema.

L'istanza della classe camera conterrà tutte le informazioni relative alla camera oggetto della prenotazione caricate dal database. L'assegnazione della camera avverrà invocando il comando AssegnaCamera() che provvederà a modificare le informazioni precedentemente caricate. Infine l'istanza della classe Camera provvederà a memorizzare le nuove informazioni nel database.



### Ristorante: Prenotazione menu

Questa procedura consente ai clienti dell'albergo, dai terminali delle loro camere, di prenotare un tavolo al ristorante per la colazione, il pranzo o la cena. I clienti possono scegliere ciò che desiderano consultando il menù giornaliero e compilando un apposito form dove vanno specificate le portate (primo, secondo, contorno, ecc.) e le quantità di ciascuna portata nel caso in cui stiano ordinando per più persone.

L'istanza della classe **SessioneUtente** consente ai clienti dell'albergo di aprire una nuova sessione e di accedere a tutti i servizi e le funzionalità riservate alla classe di utenti registrati come "Cliente". Invocando il comando **ScegliMenu()** sarà creata una nuova istanza della classe **Menu** che si occuperà di leggere il menu giornaliero dal database per consentirne la visualizzazione sui terminali dei clienti.

Tutto ciò che l'utente desidera prenotare sarà dato in input a un'istanza della classe **Prenotazione** che provvederà a memorizzare sia le informazioni relative alla prenotazione (data, costo, ecc.) sia i dettagli, cioè le portate, creando un'istanza della classe **DettagliPrenotazioneRistorante** per ciascuna portata. Queste ultime saranno responsabili per la memorizzazione effettiva delle informazioni relative alle portate ordinate.

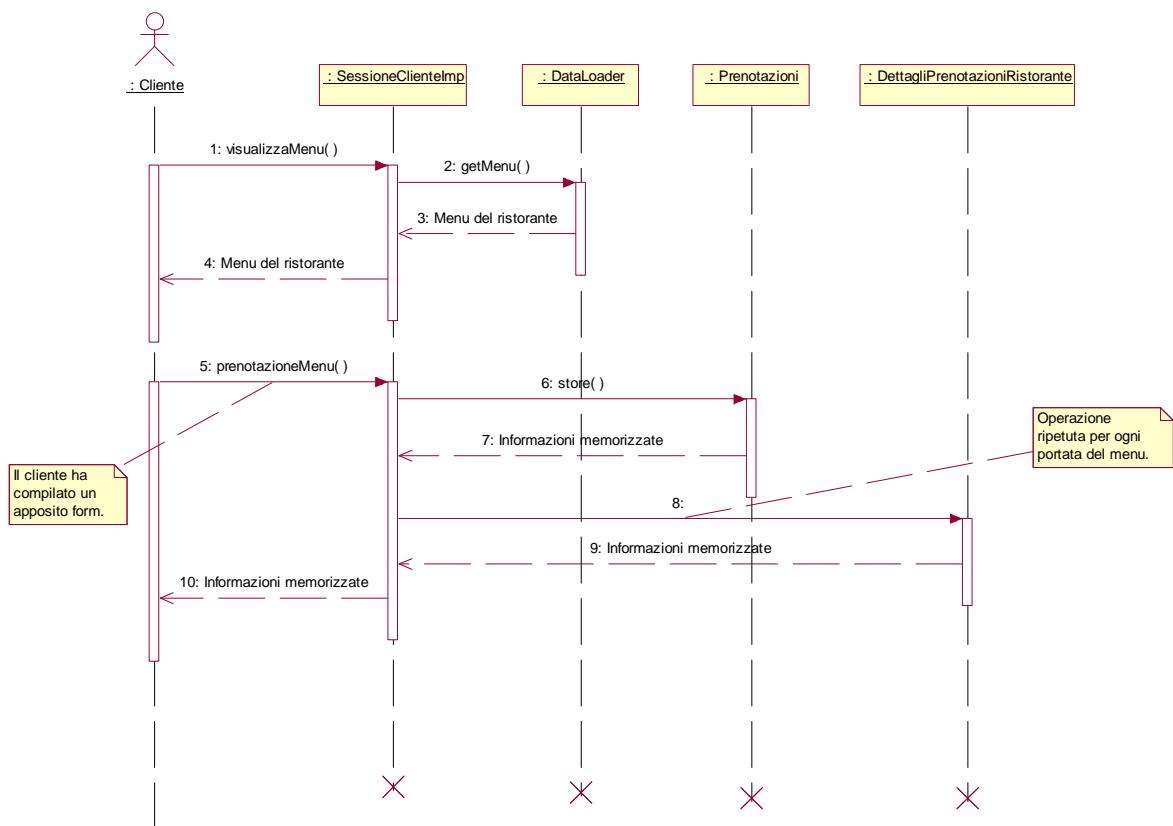


Figura 26: Sequence Diagram: Gestione Ristorante – Prenota Menu

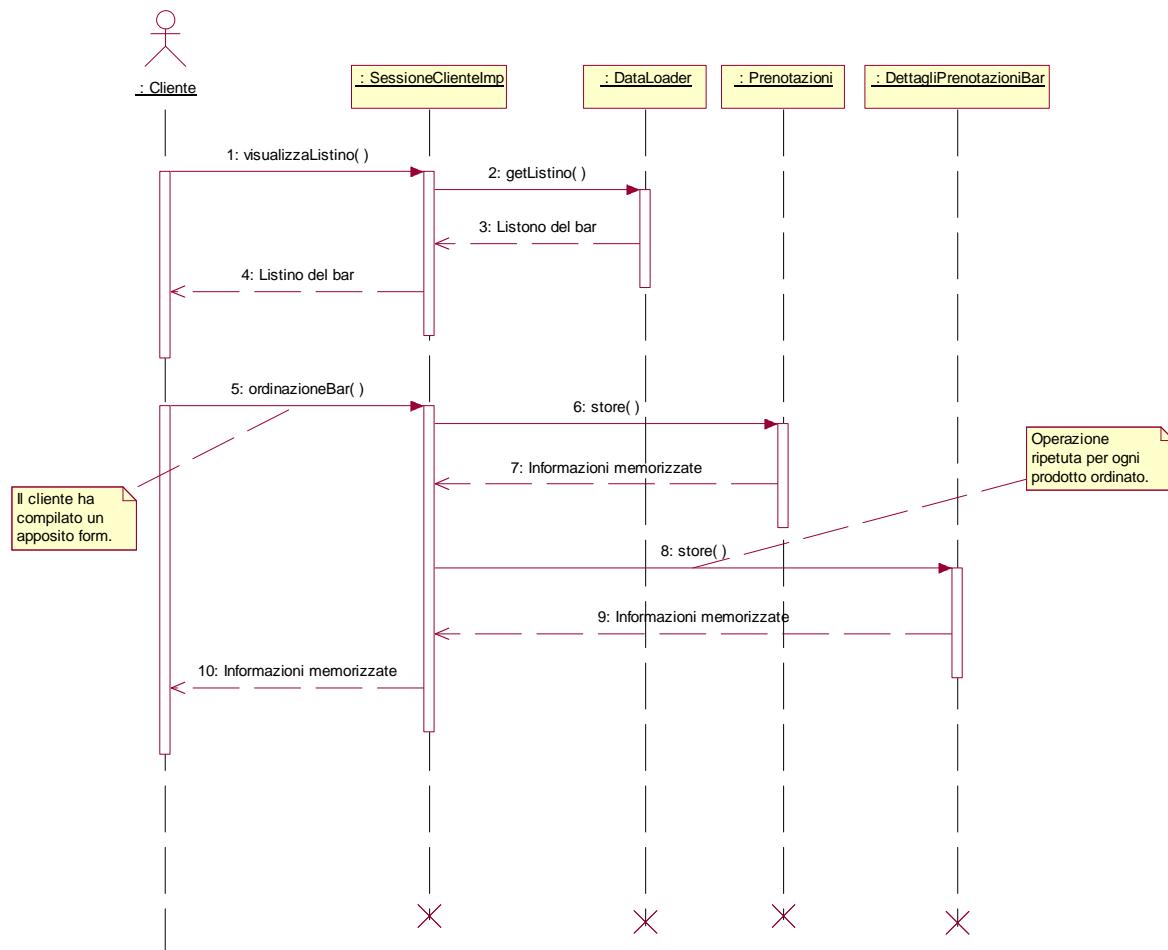
## Bar: Nuova Ordinazione

Questa procedura consente ai clienti dell'albergo, dai terminali delle loro camere, di usufruire del servizio in camera consultando il listino del bar e compilando un apposito form dove vanno specificati i prodotti da ordinare e le relative quantità.

L'istanza della classe SessioneUtente consente ai clienti dell'albergo di aprire una nuova sessione e di accedere a tutti i servizi e le funzionalità riservate alla classe di utenti registrati come "Cliente".

Il comando VisualizzaListino() creerà una nuova istanza della classe Listino che si occuperà di leggere il listino del bar dal database e di consentirne la visualizzazione sui terminali dei clienti.

Tutto ciò che l'utente desidera prenotare sarà dato in input a un'istanza della classe Prenotazione che provvederà a memorizzare sia le informazioni relative alla prenotazione (data, costo, ecc.) sia i dettagli, cioè i prodotti, creando un'istanza della classe DettagliPrenotazioneBar per ciascun prodotto. Queste ultime saranno responsabili per la memorizzazione effettiva delle informazioni relative ai prodotti ordinati.



**Figura 27: Diagramma di sequenza: Gestione Bar – Nuova Ordinazione**

## Attività Extra: Nuova Prenotazione

Questa procedura consente ai clienti dell'albergo di prenotare la loro partecipazione a una particolare attività extra. La prenotazione avviene consultando il relativo elenco e compilando un apposito form dove vanno specificate tutte le informazioni richieste (numero di posti da prenotare, ecc.).

L'istanza della classe SessioneUtente consente ai clienti dell'albergo di aprire una nuova sessione e di accedere a tutti i servizi e le funzionalità riservate alla classe di utenti registrati come "Cliente". Il comando PrenotaAttivita() creerà una nuova istanza della classe AttivitaExtra() che si occuperà di leggere la lista delle attività extra dal database e di consentirne la visualizzazione sui terminali dei clienti.

Tutto ciò che l'utente desidera prenotare sarà dato in input a un'istanza della classe Prenotazione che provvederà a memorizzare sia le informazioni relative alla prenotazione (data, costo, ecc.) sia i dettagli, cioè le singole attività, creando un'istanza della classe DettagliPrenotazioneAttivita per ciascuna attività. Queste ultime saranno responsabili per la memorizzazione effettiva delle informazioni relative alle attività prenotate.

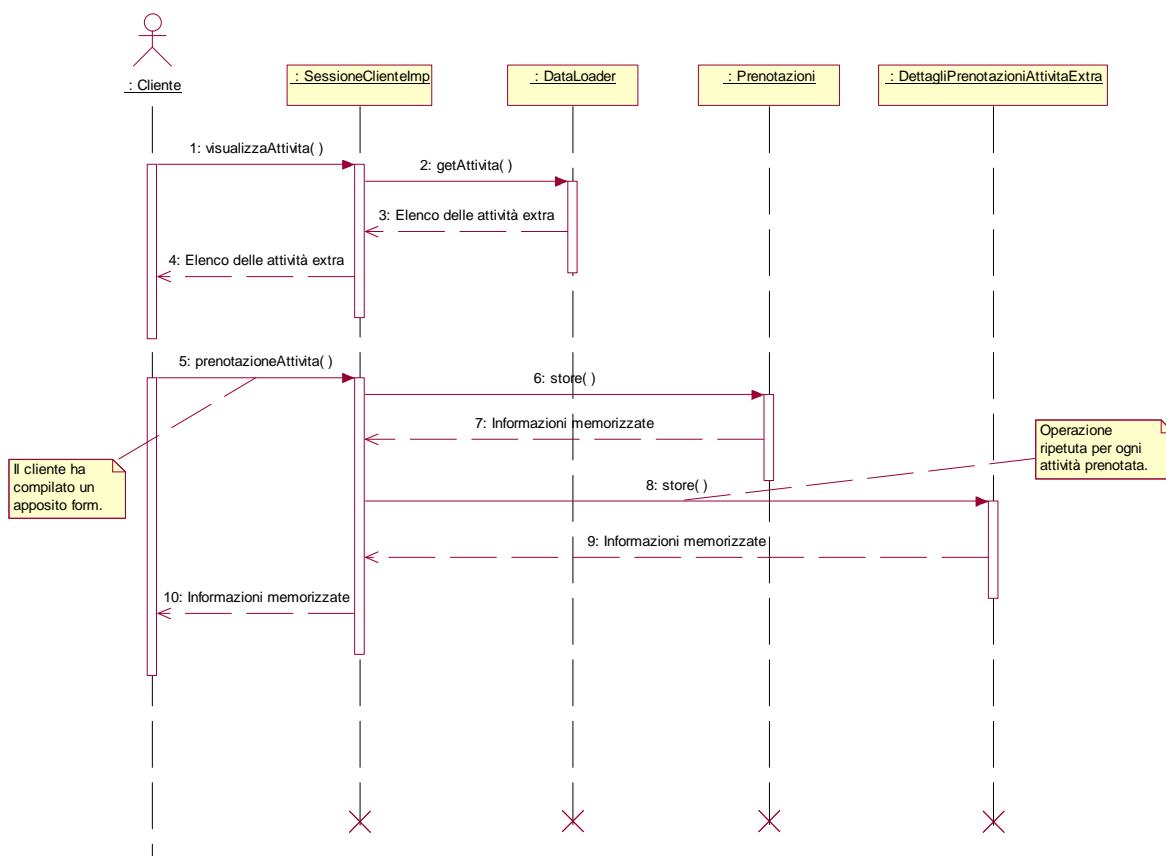


Figura 28: Diagramma di sequenza: Gestione Attività – Nuova Prenotazione

## Ristorante: Inserimento Menu

Questa procedura consente la stesura del menù giornaliero al gestore del ristorante. Essendo una procedura gestionale è richiesta l'autenticazione per poterne usufruire. L'inserimento del menù avviene compilando un apposito form dove vanno specificate le pietanze per le varie portate (primo, secondo, contorno, ecc.). Per ciascuna portata è possibile specificare più pietanze.

L'istanza della classe SessioneRistorante consente al gestore di aprire una nuova sessione nel sistema e accedere alle funzionalità riservate alla classe di utenti registrati come "Gestore del Ristorante". Per aprire una nuova sessione è necessario sottoporsi alla procedura di autenticazione fornendo il nome utente e la password per tale account. La procedura di autenticazione è realizzata dalla routine Valida() che prende in input le informazioni sopra citate e ne controlla l'autenticità.

Invocando il comando inserimentoMenu() sarà creata una nuova istanza della classe Menu che si occuperà della memorizzazione nel database di tutte le informazioni immesse dal gestore. L'invocazione del comando CreaPortata() per ogni portata del menu provvederà alla creazione di un'istanza della classe PortateMenu. L'invocazione dei comandi SetPortata() e StorePortata() delle istanze create si occuperanno della memorizzazione effettiva delle informazioni sulle portate nel database.

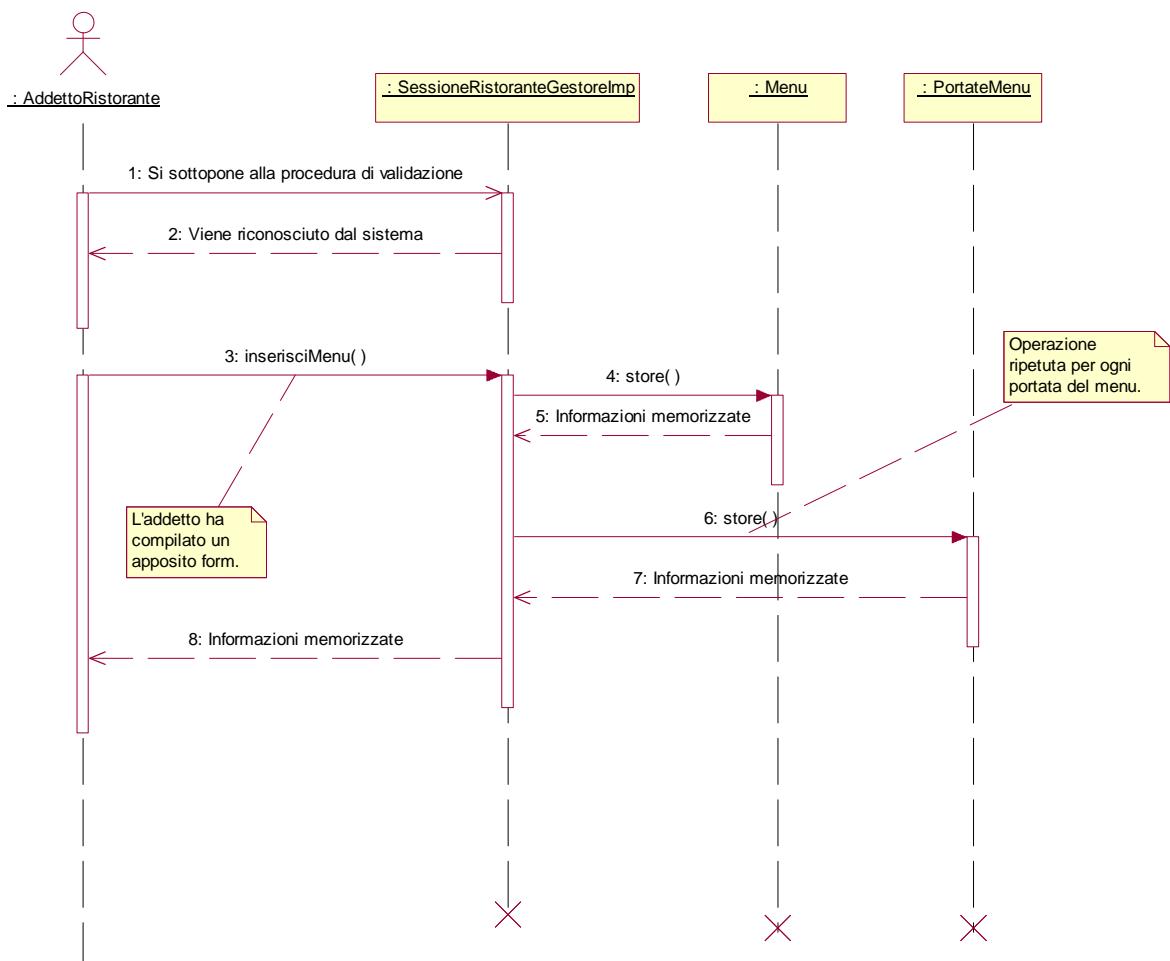


Figura 29: Diagramma di sequenza: Gestione Ristorante – Inserimento Menu

## Bar: Inserimento Prodotto

Questa procedura consente all'addetto del bar di inserire un nuovo prodotto nel listino. Essendo una procedura gestionale è richiesta l'autenticazione per poterne usufruire. L'inserimento avviene compilando un apposito form.

L'istanza della classe SessioneBar consente al responsabile di aprire una nuova sessione nel sistema e accedere alle funzionalità riservate alla classe di utenti registrati come "Gestore del Bar". Per aprire una nuova sessione è necessario sottoporsi alla procedura di autenticazione fornendo il nome utente e la password per tale account. La procedura di autenticazione è realizzata dalla routine Valida() che prende in input le informazioni sopra citate e ne controlla l'autenticità.

Il comando CreaProdotto() creerà un'istanza della classe ProdottoBar che si occuperà di memorizzare le informazioni immesse dall'addetto.

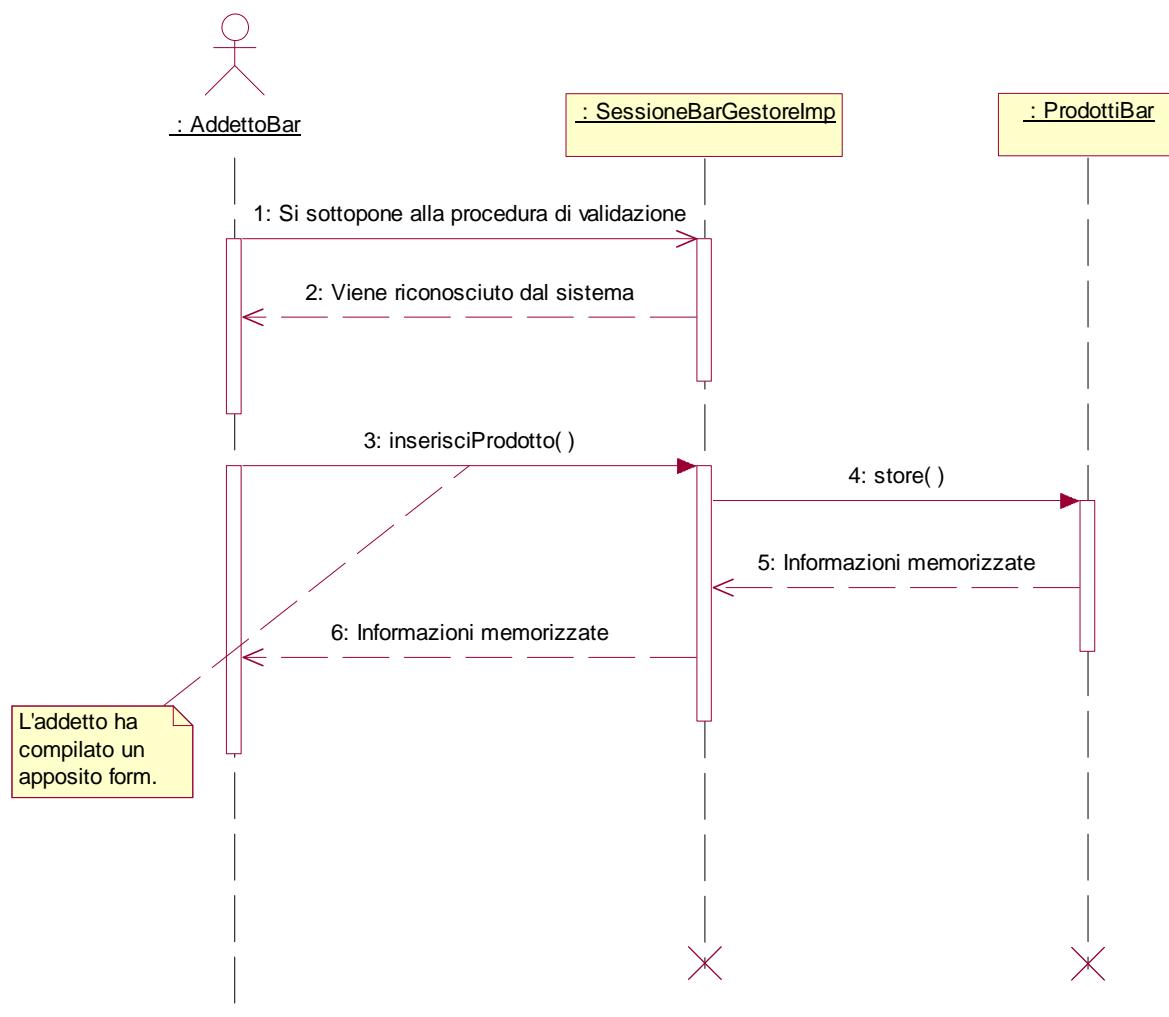


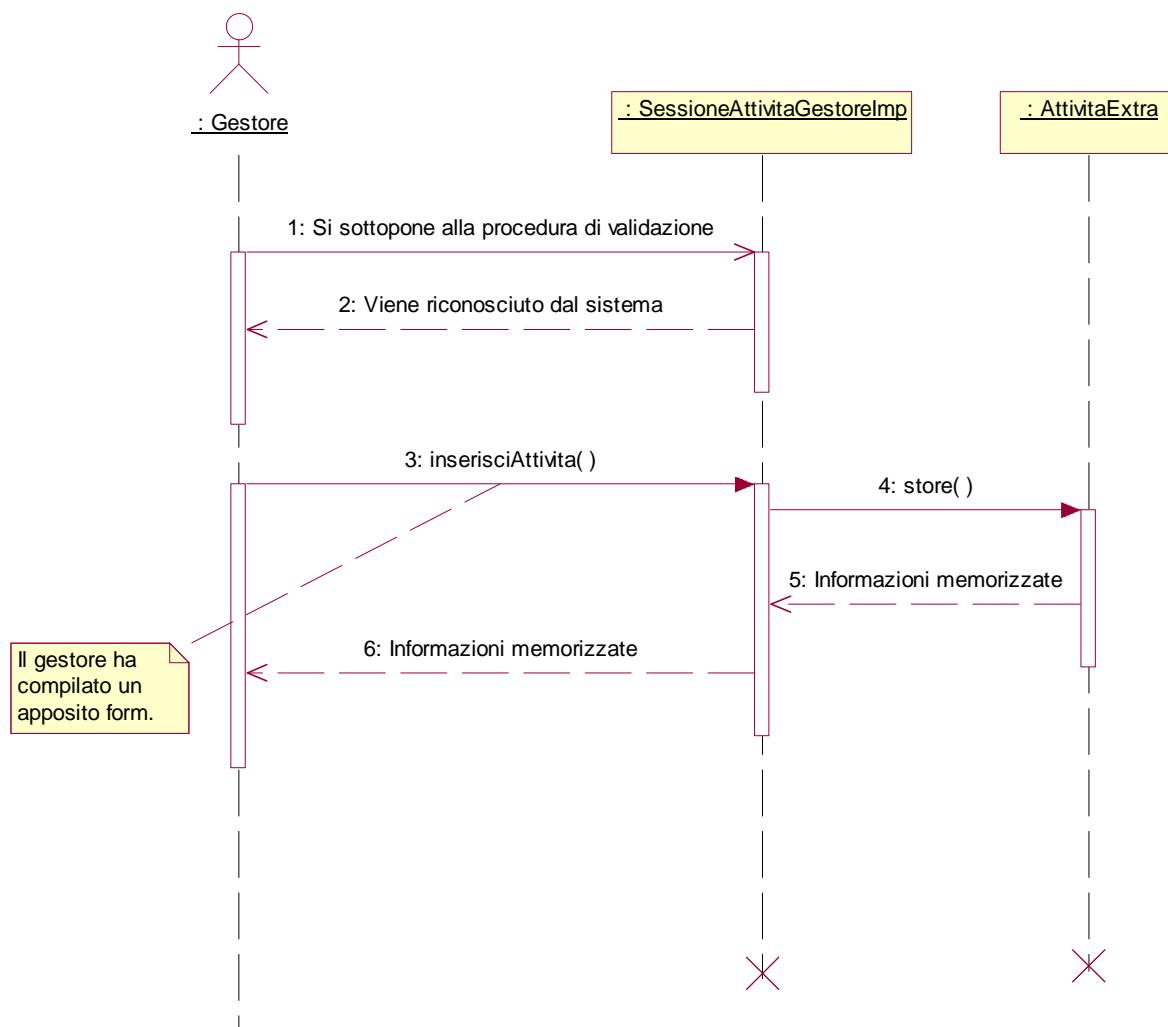
Figura 30: Diagramma di sequenza: Gestione Bar – Inserimento Prodotto

## Attività Extra: Inserimento Attività

Questa procedura consente al gestore dell'albergo di pianificare delle attività extra e di renderle disponibili ai clienti attraverso l'omonimo elenco. Essendo una procedura gestionale è richiesta l'autenticazione per poterne usufruire. L'inserimento di una nuova attività avviene compilando un apposito form dove vanno specificate tutte le informazioni relative (data e ora, eventuali posti a disposizione, data in cui si svolgerà l'attività, ecc.).

L'istanza della classe SessioneGestore consente al gestore di aprire una nuova sessione nel sistema e accedere alle funzionalità riservate alla classe di utenti registrati come "Gestore". Per aprire una nuova sessione è necessario sottoporsi alla procedura di autenticazione fornendo il nome utente e la password per tale account. La procedura di autenticazione è realizzata dalla routine Valida() che prende in input le informazioni sopra citate e ne controlla l'autenticità.

Invocando il comando CreaAttivita() sarà creata una nuova istanza della classe AttivitaExtra che consentirà di memorizzare tutte le informazioni relative alla nuova attività appena creata nel database. La memorizzazione effettiva avviene invocando i comandi SetAttivita() e StorageAttivita() della classe.



**Figura 31: Diagramma di sequenza: Gestione Attività – Inserimento Attività**

### 3.5.5 Interfaccia Utente - Navigational Paths and Screen Mockups

#### SERVIZIO ATTIVITA' EXTRA - Interfaccia lato gestore

(Login)



Quando il gestore vuole entrare nell'area di servizio deve effettuare la Login inserire login e password negli appositi campi e cliccare su “**Entra**”.

(Visualizzazione principale delle attività)

Questa è la visualizzazione principale che si presenta dopo aver effettuato il login.

Essa presenta l'elenco delle varie attività sotto forma di opzioni (per selezionarne solo una per volta), una breve descrizione sotto forma di etichetta di testo, le varie caselle di testo non modificabili contenenti: data ed ora di inizio, numero di posti disponibili e prezzo dell'attività.

Cliccando su “**Inserisci Attività**” si aprirà una nuova form:

*(Inserimento Attività)*

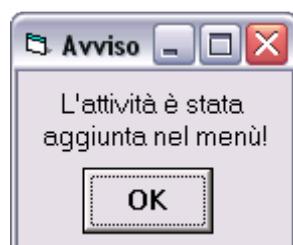


Con la quale il gestore potrà inserire una nuova attività, inserendo nome dell’attività, numero dei posti disponibili, la data, il prezzo, la descrizione dell’attività, spuntare le eventuali opzioni speciali e cliccare su “Conferma”.

Le “Opzioni Speciali” sono importanti in quanto si può con queste decidere di chiedere o meno al cliente alcune informazioni aggiuntive.

Cliccando su “Conferma” il programma ci avviserà che l’attività è stata aggiunta nel menù:

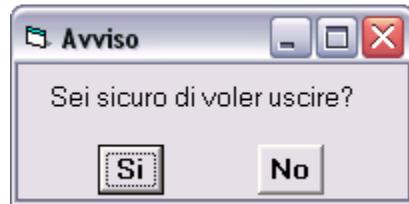
*(Avviso aggiunta menù)*



In questo caso possiamo dare solo risposta affermativa, che chiuderà la finestra e ci riporterà alla visualizzazione principale.

Cliccando su “X” il programma chiederà conferma di uscita

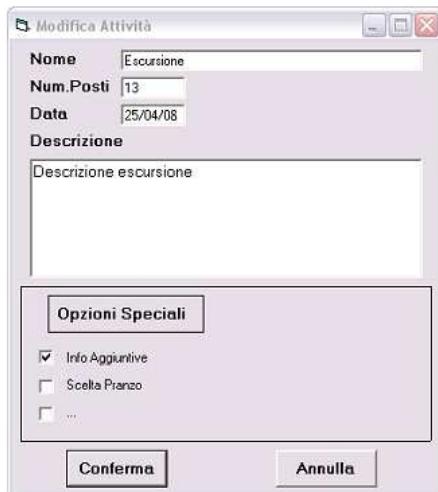
*(Avviso di uscita)*



“**Si**” ci farà tornare alla visualizzazione principale, “**No**” chiuderà l’avviso e ci riporterà davanti al form di inserimento attività.

Nella visualizzazione principale scegliendo una delle opzioni e premendo in seguito su “**Modifica Attività**”, si aprirà una form simile a quella di inserimento di una nuova attività, ma con i dati già inseriti precedentemente.

*(Modifica attività)*



Qui si possono modificare tutti i dati.

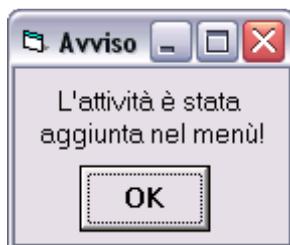
“**Annulla**” ha lo stesso comportamento che ha nella form di inserimento di un’attività, mentre il tasto conferma lancerà un diverso avviso:

*(Avviso modifica attività)*



“No” chiuderà l'avviso e ci riporterà alla form di modifica, mentre “Si” aggiungerà la nuova attività e ci informerà del successo dell'operazione con una nuova finestra:

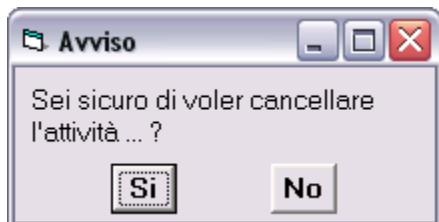
*(Avviso di aggiunta al menù)*



cliccando su “OK” torneremo alla visualizzazione principale.

Cliccando prima su una delle attività e poi su “Cancella Attività” nel menù di visualizzazione principale, il programma chiederà conferma di cancellazione di quell'elemento:

*(Avviso cancellazione attività)*



“Si” cancellerà l'attività, “No” non porterà nessun effetto alle attività, entrambi chiuderanno l'avviso e ci riporteranno alla visualizzazione principale.

Cliccando su un'attività e poi sul pulsante “Resoconto” nella visualizzazione principale, potremo vedere tutti gli iscritti a quella attività ed in guadagno totale fatto con essa:

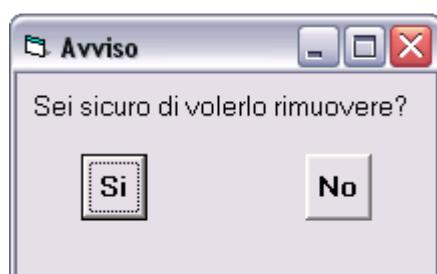
*(Resoconto)*



Il nome dell'attività e il guadagno sono racchiusi in caselle di testo non modificabili, mentre i nomi dei clienti prenotati sono racchiusi in una lista.

Cliccando prima su uno o più nomi e poi su "**Rimuovi Utente**", possiamo rimuovere manualmente una prenotazione, ma prima di farlo il programma ci chiederà conferma con un avviso:

*(Avviso rimozione)*



con "**Si**" rimuoveremo l'utente dalla lista, con "**No**" non faremo niente.

Entrambi chiuderanno la finestra di avviso e ci riporteranno alla finestra di resoconto.

Invece cliccando su "**Chiudi**" nella finestra di resoconto, ritorneremo alla visualizzazione principale.

Premendo "**Chiudi**" nella visualizzazione principale invece, ritorneremo alla finestra di login.

## SERVIZIO ATTIVITA' EXTRA - Interfaccia cliente

(Visualizzazione principale attività)

The screenshot shows a Windows-style application window titled "Menù Attività Extra". It displays three activity options:

	Data	Ora	Num.Posti	Prezzo
Nome Attività	25/04/08	15.00	13	30 €
Attività2	28/04/08	10.00	5	50 €
Attività3	30/04/08	9.00	7	100 €

Below the table are three radio buttons labeled "Nome Attività", "Attività2", and "Attività3", each with a corresponding brief description below it. At the bottom are three buttons: "Visualizza Attività" (highlighted), "Saldo", and "Chiudi".

Questa è la prima interfaccia che il cliente vede nel momento in cui entra nelle attività extra.

Per prenotare o modificare una prenotazione (se l'attività è stata già prenotata), basta attivare una sola delle Opzioni e premere su **Visualizza Attività**. L'uso dei tasti di opzione è comodo in quanto il cliente può selezionare una sola attività per volta.

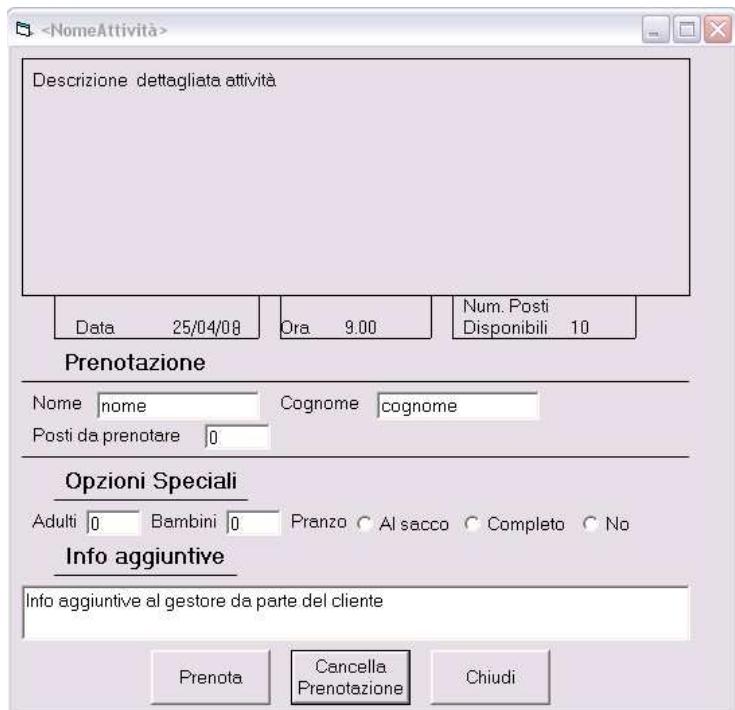
Ogni attività comprende una casella di opzione in cui è scritto il nome dell'attività, con sotto una breve descrizione di ciò che si tratta e con vicino caselle di testo contenenti informazioni su data di svolgimento, ora di inizio, numero di posti ancora disponibili e costo di partecipazione, racchiusi tutti in caselle di testo non modificabili. L'uso delle caselle di testo è stato necessario per evidenziare alcune informazioni più importanti.

Il tasto di comando "**Saldo**" visualizza il saldo del cliente corrente.

"**Chiudi**" per tornare al menù principale.

Come detto per prenotare un'attività dobbiamo selezionarne una e cliccare su "**Visualizza Attività**", se il cliente non ha già prenotato questa attività si aprirà la form di prenotazione:

*(Form di prenotazione)*



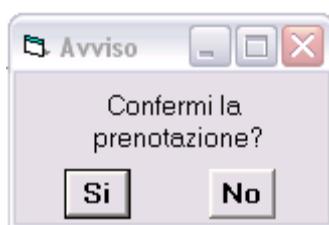
Nella quale abbiamo una etichetta di testo in cui c’è una descrizione dettagliata dell’attività, con sotto la data l’ora e il numero di posti disponibili.

Per prenotare basta inserire **Nome** e **Cognome** nelle apposite caselle di testo, specificare il numero di adulti o bambini, attivare le eventuali opzioni e cliccare su “**Prenota**”.

Le “**Info aggiuntive**” sono attivate o disattivate dal gestore e sono utili nel caso in cui il cliente voglia comunicare qualcosa agli organizzatori dell’attività.

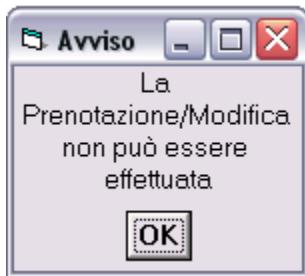
Cliccando su “**Prenota**” il programma chiederà conferma:

*(Conferma prenotazione)*



cliccando “**No**” l’avviso si chiuderà e potremo ancora modificare la prenotazione prima di inviarla; cliccando “**Si**” si invierà la prenotazione, ma nel caso in cui ci sia stato qualche problema nell’effettuare la prenotazione il programma lancerà un nuovo avviso:

*(Avviso Errore)*



Al quale l'unica risposta è “**OK**” che farà chiudere l'avviso e quindi tornare al form di prenotazione.

Cliccando su “**Cancella Prenotazione**” (nella form principale) il programma chiederà conferma per la cancellazione:

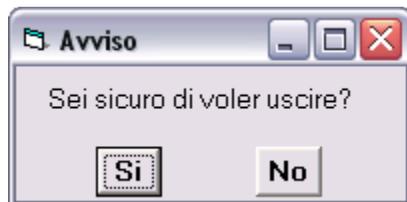
*(Conferma cancellazione prenotazione)*



cliccando “**No**” l'avviso si chiuderà e torneremo alla form di prenotazione;  
cliccando “**Si**” si invierà la cancellazione della prenotazione.

Cliccando su “**Chiudi**” (nella form principale) il programma chiederà conferma:

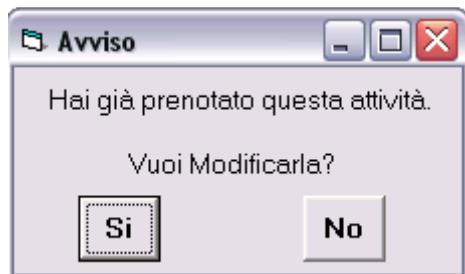
*(Conferma di uscita dal form di prenotazione)*



cliccando “**No**” l'avviso si chiuderà e torneremo alla form di prenotazione;  
cliccando “**Si**” farà tornare alla visualizzazione principale delle attività.

Quando nella visualizzazione principale si clicca su “**Visualizza Attività**” se il cliente ha già prenotato quella attività il programma lancerà un avviso:

*(Avviso modifica)*

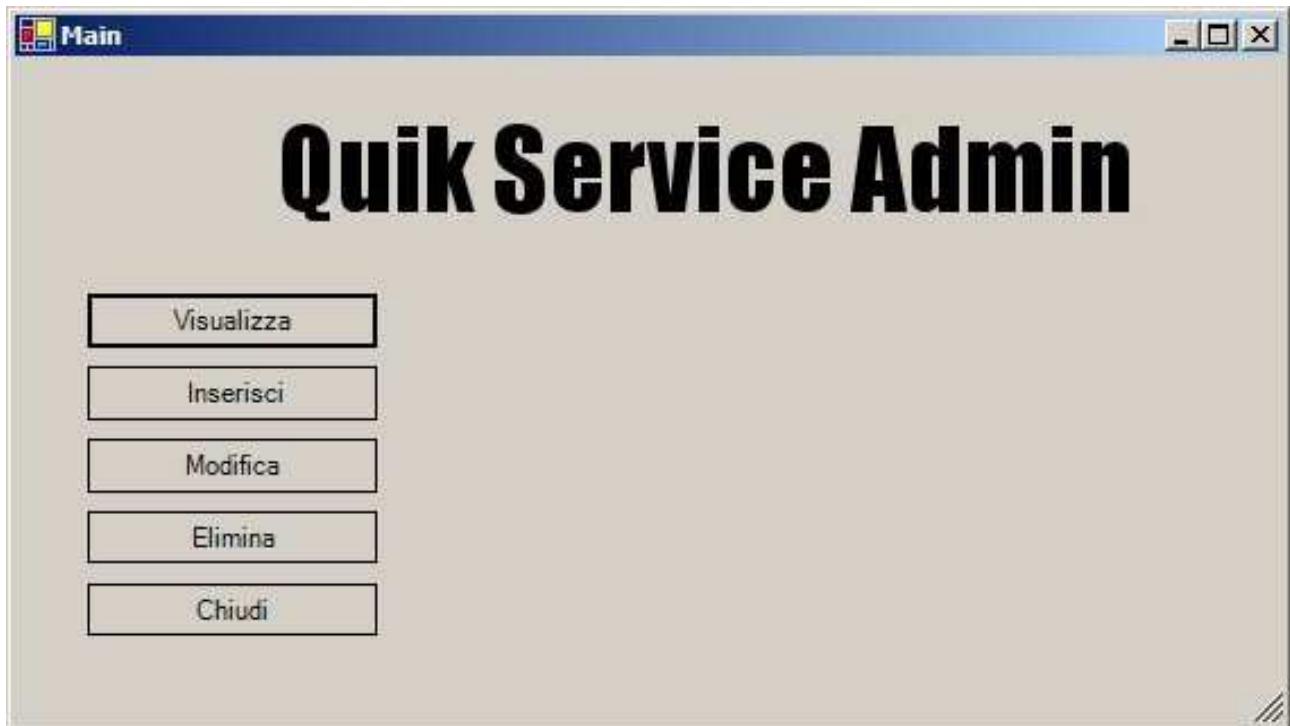


Cliccando su “**No**” la finestra di avviso si chiuderà e il cliente si troverà di fronte alla visualizzazione delle attività.

Cliccando su “**Si**” si aprirà il form di Prenotazione già compilato con i dati inseriti in precedenza.

## SERVIZIO BAR - Interfaccia lato gestore

(Main)



Interfaccia principale per la gestione del bar composta dai pulsanti che permettono di sfruttare le funzionalità del sistema.

Il pulsante **Visualizza** mostra il listino dei prodotti in una nuova finestra.

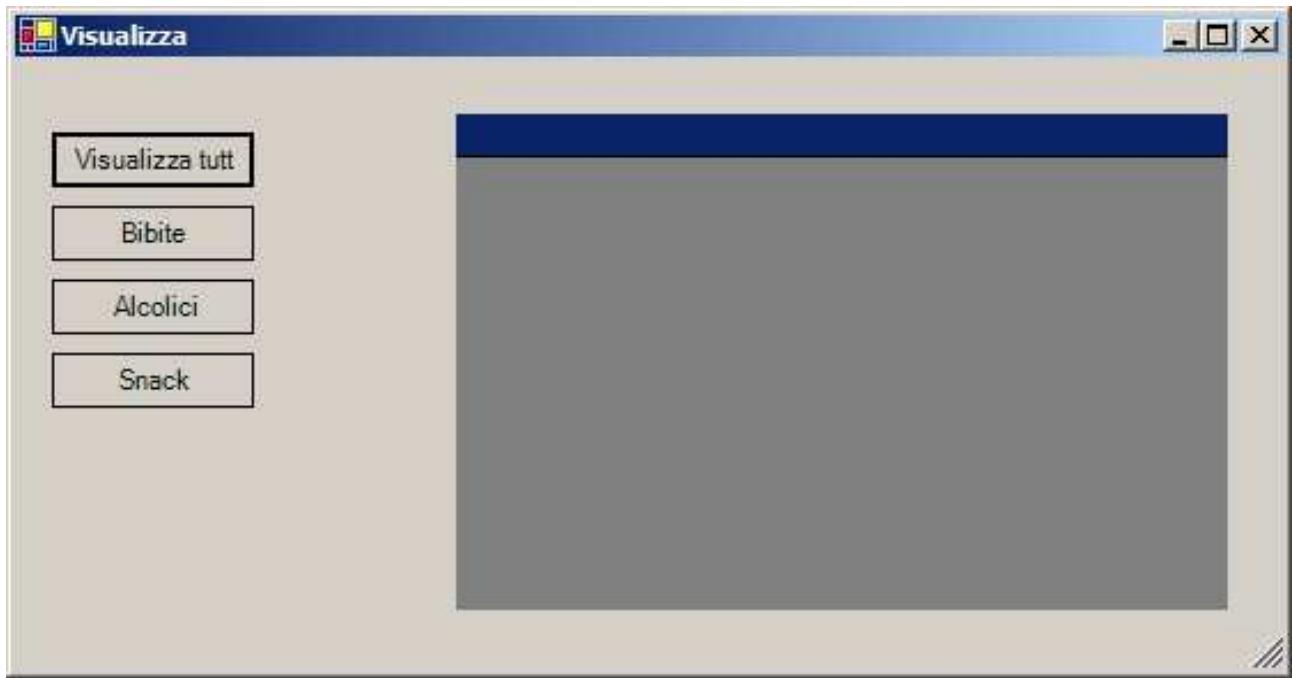
Il pulsante **Inserisci** mostra la form per l'inserimento di un nuovo prodotto.

Il pulsante **Modifica** mostra una finestra per in cui compare l'elenco dei prodotti da selezionare per potere essere modificati.

Il pulsante **Elimina** mostra una finestra per in cui compare l'elenco dei prodotti da selezionare per potere essere eliminati.

Il pulsante **Chiudi** chiude l'applicazione

(Visualizza prodotto)



L'interfaccia mostra a video il listino, sulla sinistra vengono mostrati i pulsanti per la selezione dei vari listini sulla destra la tabella mostra il listino in base al pulsante premuto, di default viene mostrato il listino completo.

Il pulsante **Visualizza** tutti mostra il listino completo.

Il pulsante **Bibite** mostra il listino delle bibite.

Il pulsante **Alcolici** mostra il listino specifico per gli alcolici.

Il pulsante **Snack** mostra il listino specifico per gli snack.

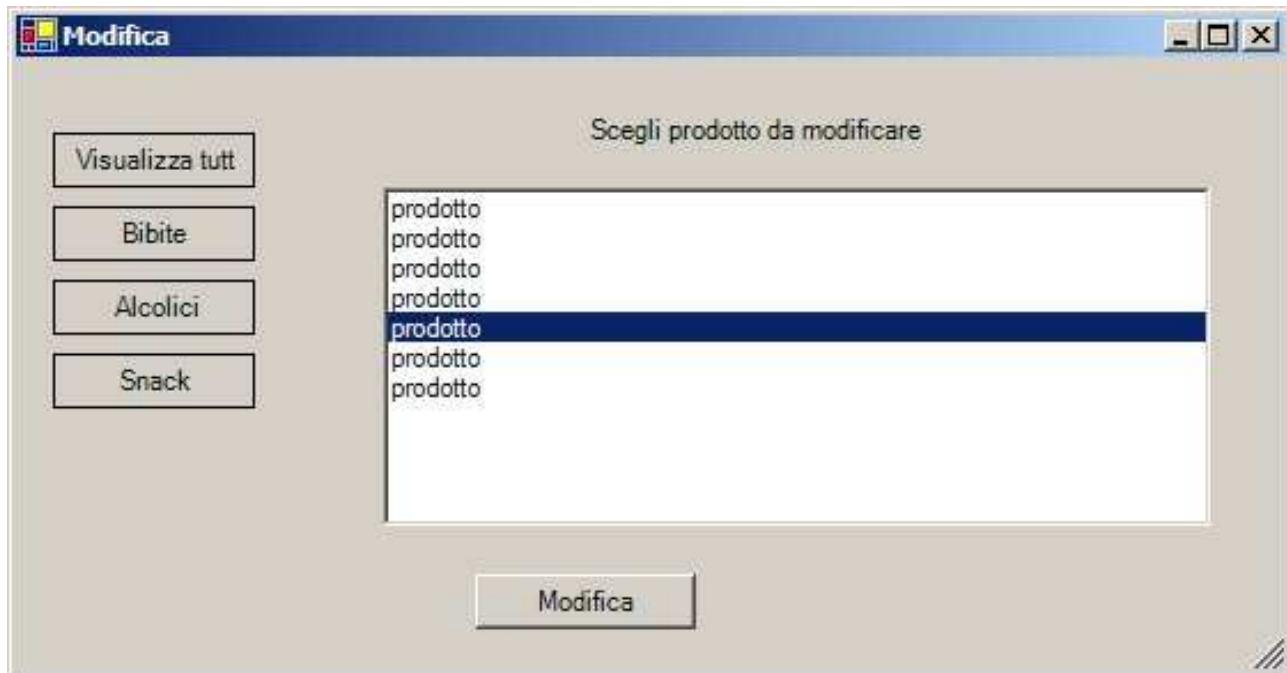
Il pulsante **Chiudi** chiude la finestra e ritorna alla finestra principale.

*(Inserimento prodotto)*

The screenshot shows a Windows application window titled "Inserisci". The window has a blue header bar with the title and standard window controls (minimize, maximize, close). Inside, there are five text input fields arranged vertically. Each field has a label to its left: "Codice", "Nome", "Quantità", "Prezzo", and "Tipo". To the right of the "Tipo" label is a "ComboBox1" control with a dropdown arrow. At the bottom of the window are two buttons: "Inserisci" on the left and "Resetta" on the right.

Interfaccia per l'inserimento di nuovi prodotti nel sistema, sono previsti i campi testo per l'inserimento del **Codice**, **Nome**, **Prezzo** mentre per semplificare la selezione del **Tipo** di prodotto viene usata una combo box dalla quale è possibile selezionare la tipologia del prodotto da quelli disponibili nel sistema, i due pulsanti in basso permettono di agire sul form. Il pulsante **Inserisci** valida i dati, li inserisce all'interno del sistema. Il pulsante **Resetta** inserisce in tutti i campi dell'interfaccia i valori di default.

*(Modifica prodotto)*

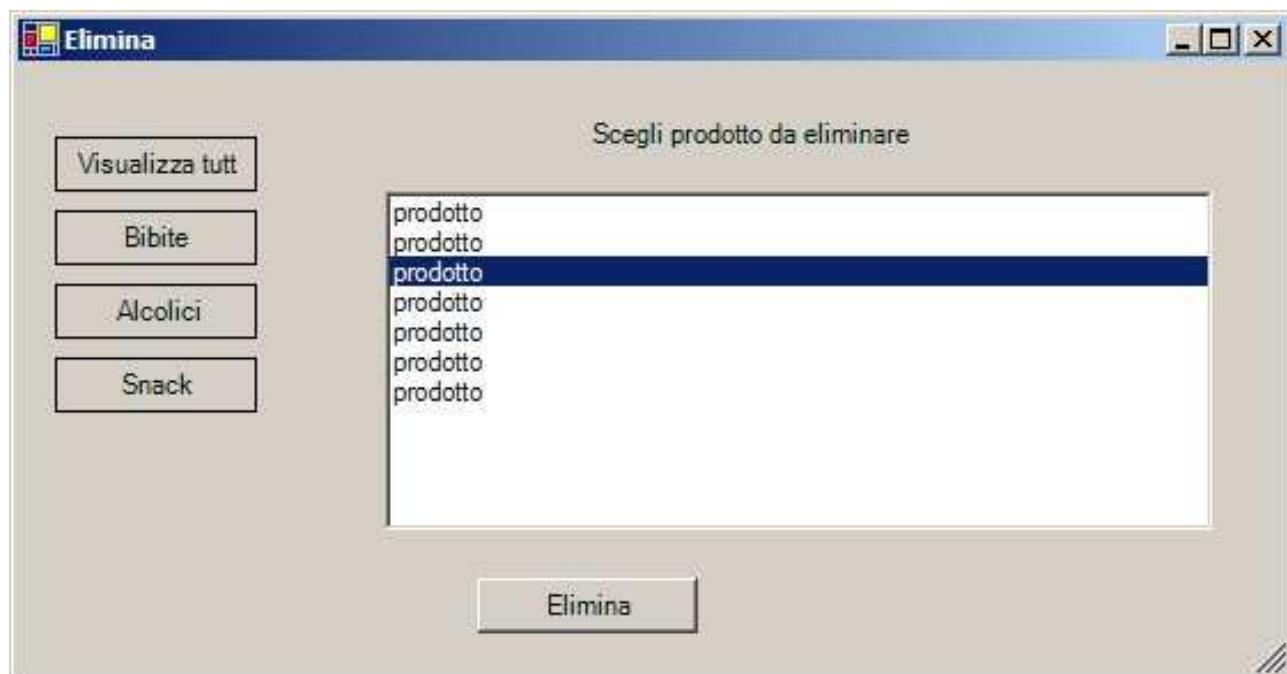


Interfaccia composta dai pulsanti di selezione del listino sulla sinistra, e da una lista dove sono elencati i prodotti del listino selezionato (vedi *visualizza*).

Dalla lista è possibile selezionare un prodotto per volta per modificarlo.

In basso il pulsante **Modifica** consente di aprire la finestra con la form per la modifica del prodotto selezionato (vedi maschera di inserimento).

*(Elimina prodotto)*



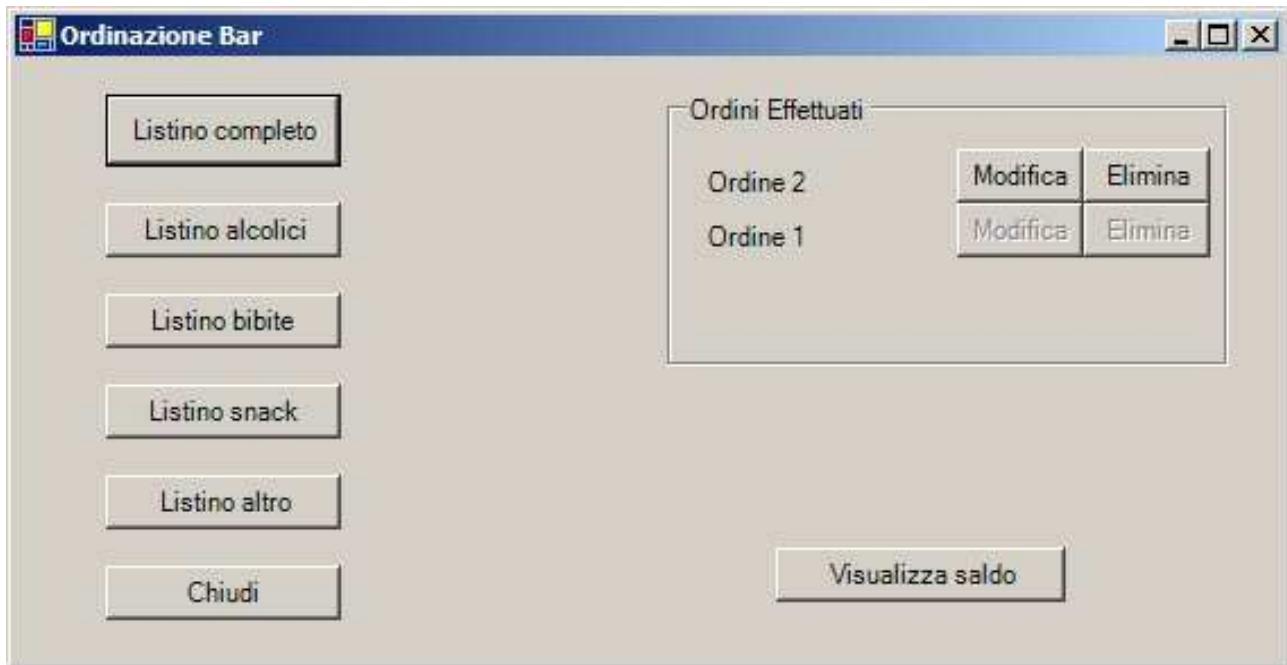
Interfaccia composta dai pulsanti di selezione del listino sulla sinistra, e da una lista dove sono elencati i prodotti del listino selezionato (vedi *visualizza*).

Dalla lista è possibile selezionare più di un prodotti da eliminare.

In basso il pulsante **Elimina** alla pressione del pulsante viene richiesta la conferma per l'eliminazione degli elementi selezionati .

## SERVIZIO BAR - Interfaccia lato cliente

(Main)



Questa interfaccia mostra al cliente le azioni che può effettuare, sulla sinistra ci sono i pulsanti che permettono la scelta del listino da visualizzare, a destra invece viene mostrato il riepilogo degli ordini effettuati al bar con i pulsanti per modificare o cancellare l'ordine, in basso c'è il pulsante per accedere al saldo.

I pulsanti che permettono la selezione dei listini (**Listino completo**, **Listino alcolici**,etc...) aprono una nuova finestra in cui compare il listino con i controlli necessari per poter effettuare un ordine.

Nel riepilogo degli ordini effettuati dal cliente compaiono per ogni ordine i pulsanti per eliminare e modificare l'ordine, questi pulsanti sono abilitati fin quando non scade il tempo massimo in cui è possibile modificare/eliminare un ordine.

Il pulsante **Modifica** apre una nuova finestra che permette di modificare l'ordinazione.

Il pulsante **Elimina** cancella l'ordine richiedendo prima la conferma.

Il pulsante **Saldo** apre una nuova finestra dove compare il dettaglio del saldo presso la struttura.

Il pulsante **Chiudi** chiude la finestra e ritorna alla schermata principale.

*(Visualizza listino)*

The screenshot shows a Windows application window titled "Listino selezionato". The window contains a table with three rows of data. The columns are labeled "Prodotto", "Prezzo", "Ordina", and "Quantità". Each row has a "Prodotto" entry ("Prodotto", "Prodotto", "Prodotto"), a "Prezzo" entry ("10,00€", "10,00€", "10,00€"), an empty "Ordina" checkbox, and an empty "Quantità" text input field. Below the table, the total values are displayed: "00,00 €" for the sum of prices and "0" for the total quantity. At the bottom are two buttons: "Ordina" and "Resetta".

Prodotto	Prezzo	Ordina	Quantità
Prodotto	10,00€	<input type="checkbox"/>	<input type="text"/>
Prodotto	10,00€	<input type="checkbox"/>	<input type="text"/>
Prodotto	10,00€	<input type="checkbox"/>	<input type="text"/>

00,00 €      0,00 €      0      0

Ordina      Resetta

L'interfaccia mostra la lista dei prodotti con affianco per ogni voce il prezzo unitario un check per la selezione del prodotto da ordinare e una casella di testo per inserirne la quantità, in basso alla tabella è riportato il totale della spesa.

In basso sono posizionati i pulsanti per l'ordinazione e per resettare il listino.

Per la selezione dei prodotti da ordinare vengono usati i check box che permettono la selezione di più elementi.

Il pulsante **Ordina** richiede la conferma dell'ordine, chiude la finestra e ritorna alla schermata principale.

Il pulsante **Resetta** riporta il listino ai valori di default.

*(Modifica Ordinazione)*

**Ordine n° x**

Listino			
Prodotto	Prezzo	Ordina	Quantità
Prodotto	10,00€	<input type="checkbox"/>	<input type="text"/>
Prodotto	10,00€	<input type="checkbox"/>	<input type="text"/>
Prodotto	10,00€	<input type="checkbox"/>	<input type="text"/>

Total 30,00 €      n      xx

**Conferma**    **Aggiungi...**    **Elimina**

L'interfaccia permette al cliente di modificare l'ordine precedentemente effettuato, questa presenta la lista dei prodotti scelti dal cliente (vedi visualizza).

In basso vediamo tre pulsanti che permettono all'utente di confermare,aggiungere ed eliminare l'ordine.

Il pulsante **Conferma** chiede la conferma da parte del cliente per le modifiche apportate, valida le scelte effettuate dall'utente applica le modifiche all'ordine le invia al sistema e ritorna alla finestra principale.

Il pulsante **Aggiungi...** apre una nuova finestra con il listino per la selezione di nuovi prodotti.

Il pulsante **Elimina** chiede la conferma per l'eliminazione dell'ordine, lo cancella dal sistema e ritorna alla finestra principale del programma.

*(Visualizza Saldo)*



L'interfaccia mostra il riepilogo del saldo, ogni ordine/servizio acquistati corrisponde il prezzo e un pulsante per vederne i dettagli .

I pulsanti **Vedi** consentono di visualizzare i dettagli per ogni ordine/servizio acquistato.

Il pulsante **Chiudi** chiude la finestra e ritorna alla finestra principale.

*(Dettaglio saldo)*

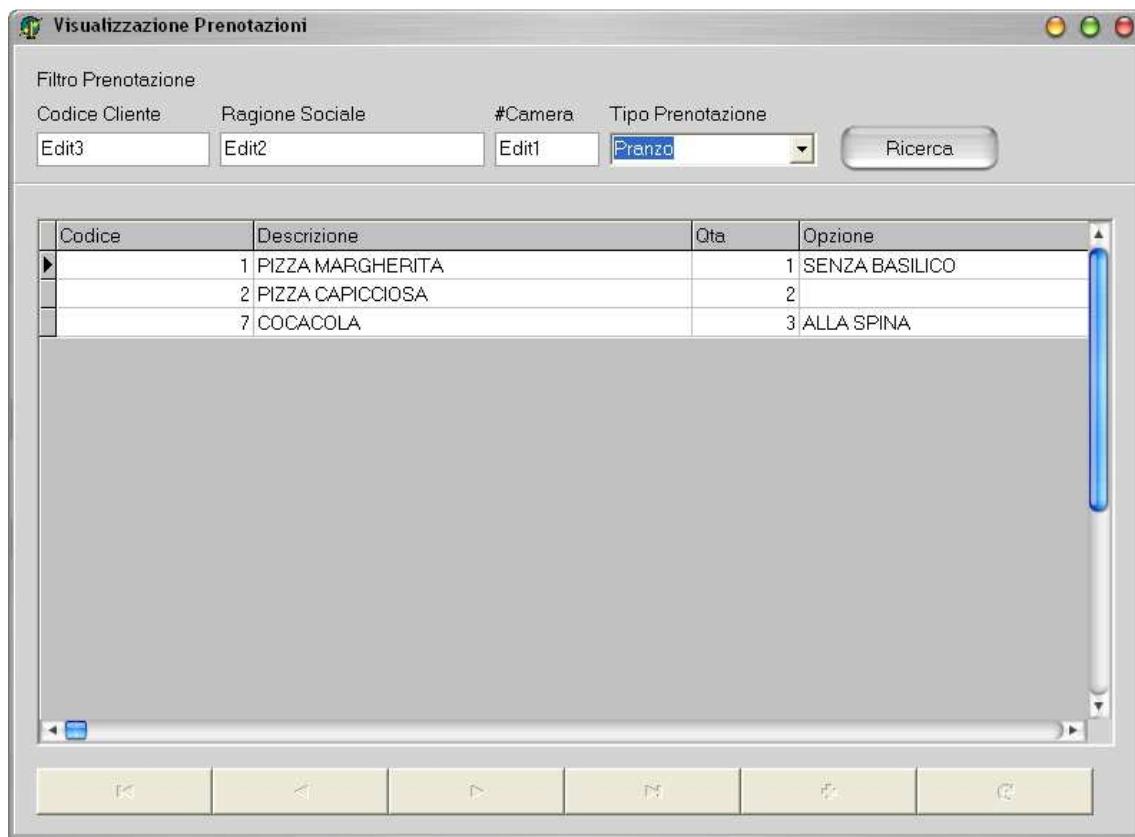
The screenshot shows a Windows application window titled "Ordine n° x". The main area is a table titled "Listino" (Bill) with three columns: "Prodotto" (Product), "Prezzo" (Price), and "Quantità" (Quantity). There are three rows of data, each showing a product name, a price of "10,00€", and a quantity of "1". At the bottom of the table, there is a summary row labeled "Totale" (Total) with a value of "30,00 €" and a note "xx" next to it. An "OK" button is located at the bottom center of the window.

Prodotto	Prezzo	Quantità
Prodotto	10,00€	1
Prodotto	10,00€	1
Prodotto	10,00€	1
<b>Totale</b>	<b>30,00 €</b>	<b>xx</b>

L'interfaccia mostra la lista dei prodotti acquistati con relativi prezzi e quantità lo stile è conforme al resto dell'applicazione le caselle di testo sono disabilitate e quindi non è possibile modificare i valori. In basso il pulsante di conferma ritorna alla schermata precedente.

## SERVIZIO RISTORANTE - Interfaccia lato gestore

(visualizza prenotazioni)



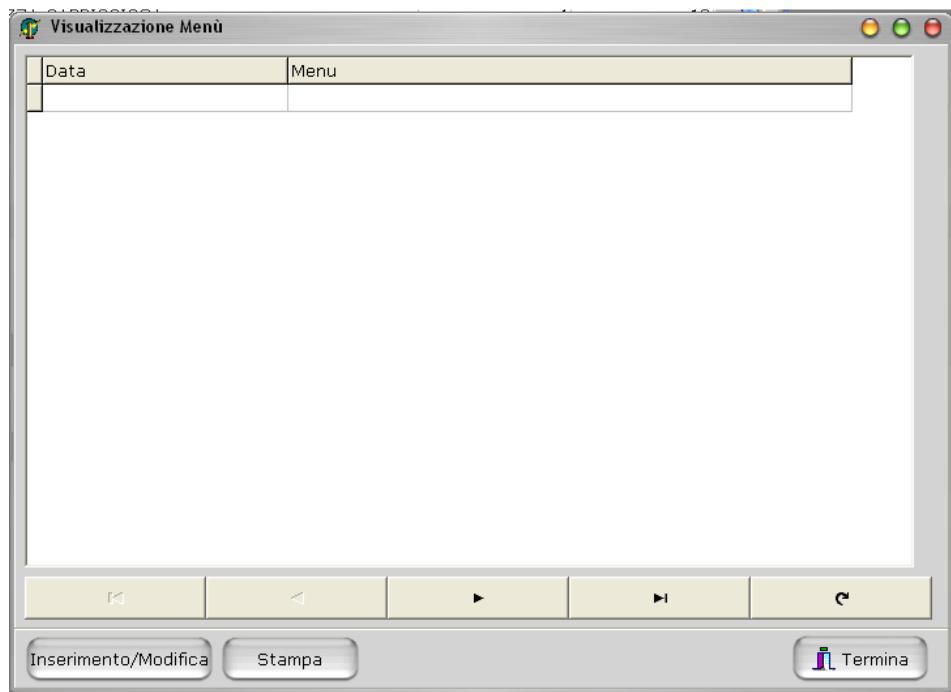
Questa Interfaccia permette al Gestore del Ristorante, o a qualsiasi altro utente di visualizzare le prenotazioni effettuate dai clienti delle camere.

In alto Possiamo notare un set di Campi di testo e una ComboBox, I campi di testo rappresentano valori da utilizzare come filtro nella ricerca di una prenotazione, ad esempio una ricerca per camera, per codice cliente, o per Ragione Sociale del Cliente, La ComboBox fornisce un filtro aggiuntivo che permette di scindere le prenotazioni tra pranzo e cena, potrebbe essere inserito un ulteriore filtro per dividere le prenotazioni per data.

Il Pulsante a Seguire “Ricerca” Permette di applicare i filtri stabiliti ed eseguire la ricerca desiderata.

I risultati Della ricerca saranno presentati nella griglia in basso, interfacciata con un navigatore che permette in modo automatico di spostarsi di record in record.

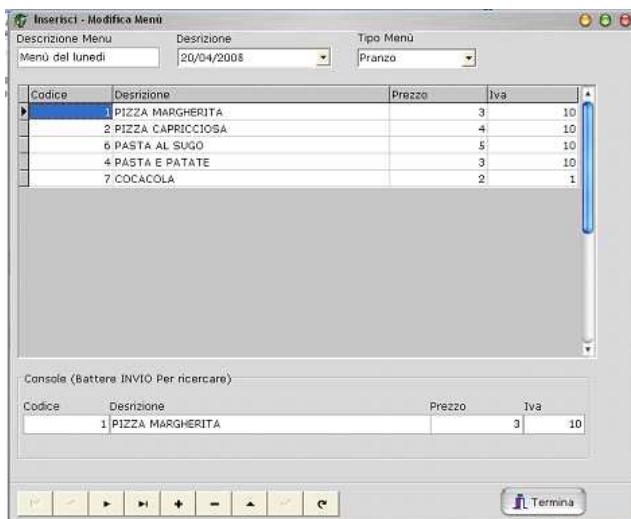
**(Visualizza menu)**



Questa schermata contiene tutti i menù disponibili che una volta selezionati con la pressione del tasto “Inserimento/Modifica” possono essere editati una volta aver eseguito il Log – in.

Il tasto “Stampa” ovviamente permette la stampa del menù selezionato, e il tasto “Termina” Chiude l’applicazione. Il navigatore in basso subito dopo la griglia permette di spostarsi tra i record senza utilizzare la griglia, in modo automatico permette di posizionarsi sul primo, l’ultimo, il successivo e il precedente.

*(inserimento e modifica delle portate nei menu)*



Questa interfaccia è il fulcro della gestione del ristorante, una volta validato l'addetto può inserire e modificare un menù, in alto abbiamo i dati base del menù, una eventuale descrizione, una data di validità, e un tipo di menù ovvero Tipo pranzo/Cena.

Subito a seguire vediamo una griglia che contiene i prodotti abbinati al menù appena creato o da modificare, queste righe potranno essere editate e cancellate, rispettando ovviamente i vincoli definiti in fase di progetto.

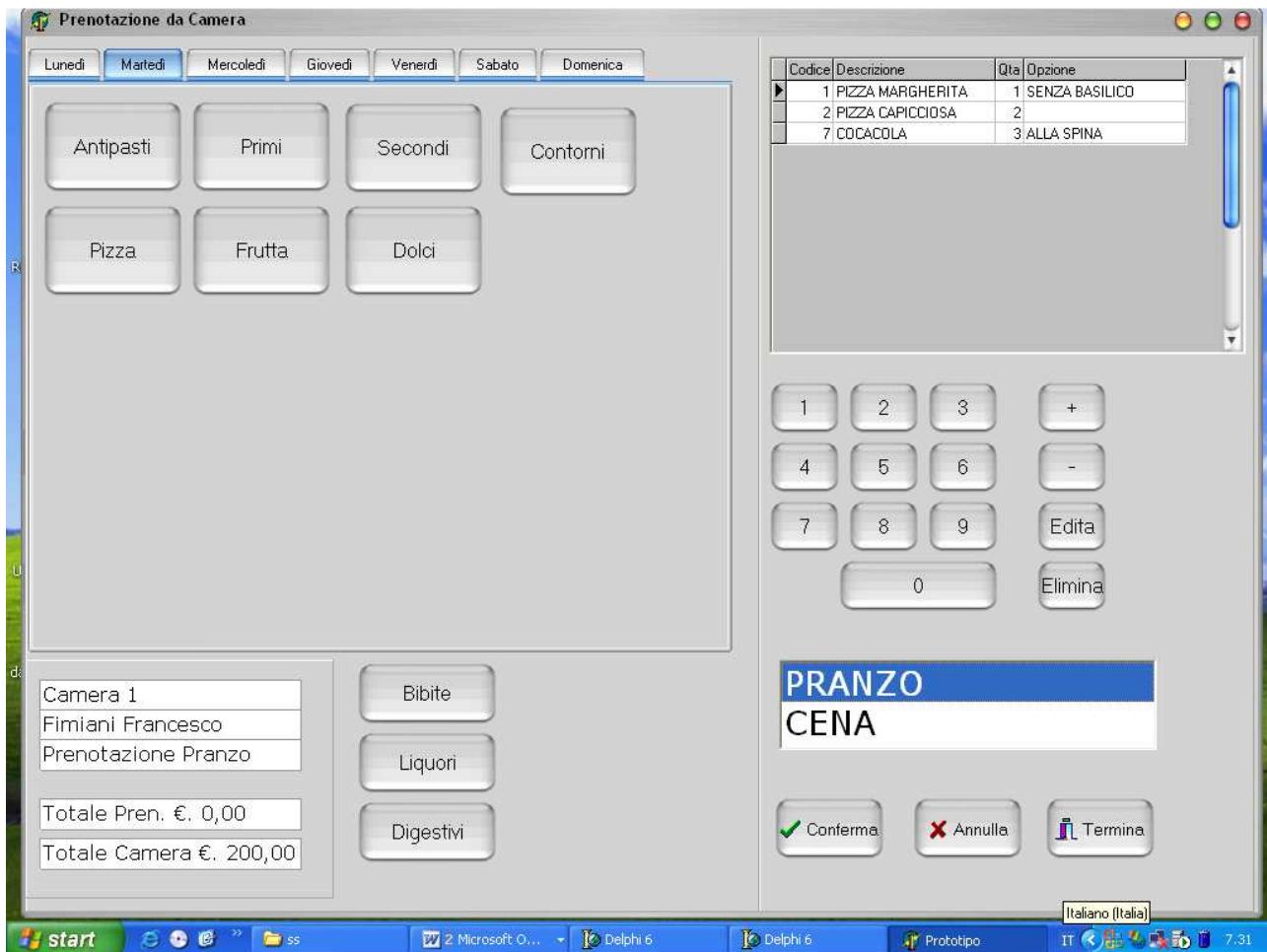
Nella GroupBox Console Abbiamo dei campi di testo, un codice, una descrizione ed un prezzo, questi campi serviranno sia per la ricerca del prodotto da inserire, sia per una eventuale modifica del prodotto inserito, infatti digitando il codice o parte della descrizione e battendo invio, si ha il completamento automatico dei campi, che possono essere modificati o lasciati invariati, premendo il tasto + il prodotto viene abbinato al menù scelto, altrimenti utilizzando le frecce di navigazione o il mouse sulla griglia, ci si sposta da prodotto a prodotto e i campi vengono immediatamente ricompilati permettendo la modifica della riga selezionata, con il tasto – il prodotti viene eliminato, con la freccetta ricurva viene fatto il refresh e con il tasto ^ si annullano le modifiche effettuate.

Infine il Tasto termina chiude l'applicazione di inserimento e modifica.

Questa schermata potrebbe essere ulteriormente migliorata inserendo un'altra griglia contenente tutti i prodotti che potranno essere inseriti mediante la selezione multipla e il Drag & Drop nella griglia contenente i prodotti abbinati al menù.

## SERVIZIO RISTORANTE - Interfaccia lato cliente

(Scelta menù e visualizzazione prenotazioni)



Questa interfaccia infine rappresenta il lato Client del Quick Touch Service – sezione Ristorante. Il cliente in camera si trova di fronte ad un’interfaccia facile ed intuitiva, in alto si vede una serie di pulsanti raffiguranti i giorni della settimana, ogni pagina contiene un set di pulsanti che una volta premuti visualizzano le portate della categoria scelta, ad esempio la pressione del pulsante “Primi” comporta la visualizzazione di tutti i primi piatti disponibili per quel giorno sempre sotto forma di pulsanti, al cliente basta premere il pulsante per aggiungere alla sua prenotazione e contestualmente alla griglia in alto a destra il piatto selezionato e premendo i pulsanti del tastierino numerico immediatamente può variarne la quantità desiderata, con i tasti + e – la quantità invece aumenta e diminuisce gradualmente, il pulsante elimina toglie la portata selezionata dalla prenotazione e il tasto edita permette di inserire eventuali opzioni alla portata selezionata.

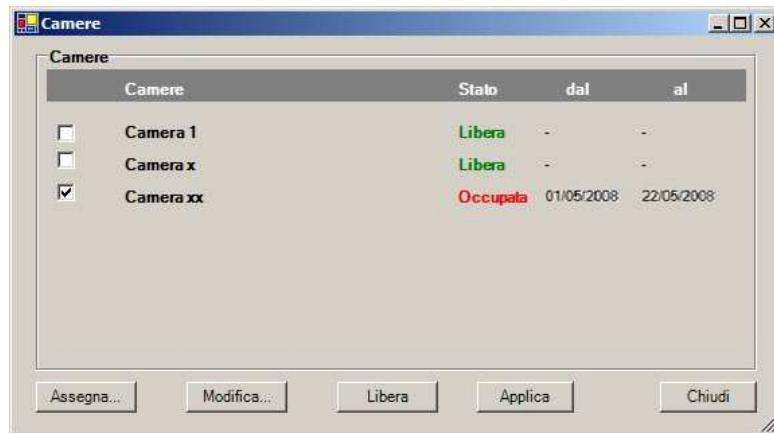
Nella Lista in basso a Destra è possibile selezionare i piatti scelti per il pranzo o per la cena, mentre nel riquadro in basso a sinistra è disponibile un breve promemoria che mostra il tipo di prenotazione in corso, il nome del cliente, la camera e il totale sia parziale che totale.

Per confermare la prenotazione basterà premere il tasto “Conferma”, mentre per annullarla nei limiti descritti lato progetto basterà la pressione del tasto “Annulla”, mentre “Termina” chiude l’applicazione.

Con questa interfaccia il cliente può immediatamente tenere sotto controllo le prenotazioni effettuate in tutta la settimana e nei giorni a seguire, inoltre i tasti opportunamente selezionati e dimensionati permettono un corretto uso dell’interfaccia sia con telecomando, che con mouse, che con touch screen.

## SERVIZIO CAMERE

(Main)



L'interfaccia presenta una tabella nella quale sono inserite i dati relativi alle **camere** e al loro **stato** e le **date** riferite all'arco di tempo in cui la camera risulta occupata, tramite i check box sulla sinistra è possibile selezionare una o più camere per effettuare le operazioni tramite i pulsanti in basso.

Il pulsante **Assegna...** mostra una nuova form nella quale è possibile assegnare la camera selezionata (o le camere selezionate) al cliente.

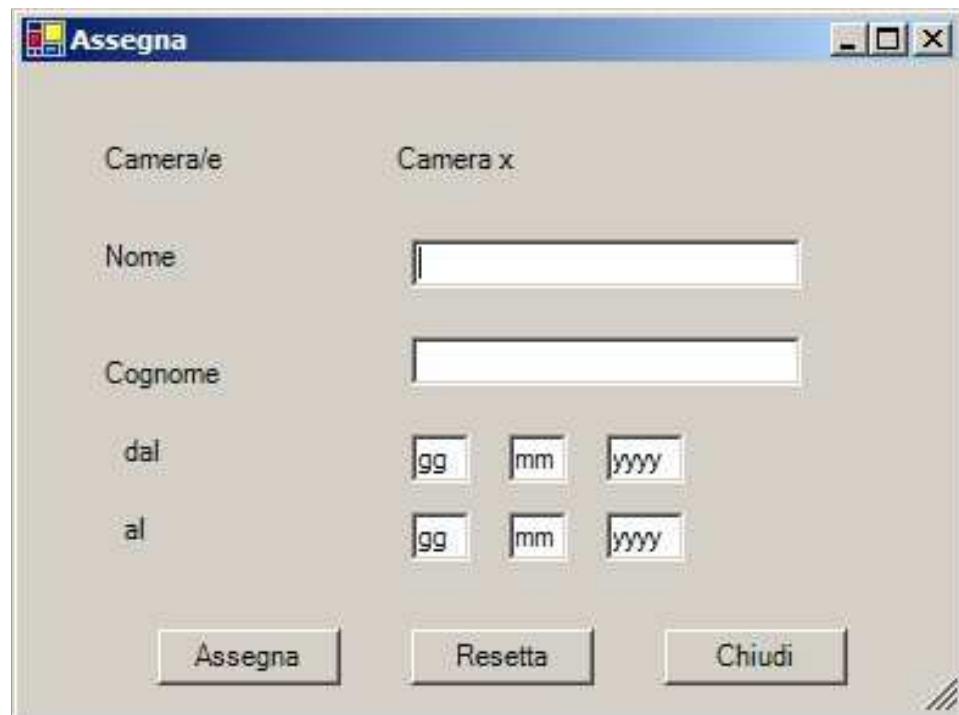
Il pulsante **Modifica...** mostra una nuova form che permette di modificare la prenotazione del cliente.

Il pulsante **Libera** serve per impostare a “libero” lo stato di una o più camere selezionate, di default è disabilitato, viene abilitato quando viene selezionata una o più stanze con stato “occupato”. Alla sua pressione apparirà un messaggio per avere la conferma da parte dell’utente di cambiare lo stato delle camere selezionate.

Il pulsante **Applica** consente di applicare le modifiche apportate, di default è disabilitato, viene abilitato ad ogni cambiamento apportato al sistema; alla pressione apparirà un messaggio che chiede la conferma per applicare tutte le modifiche.

Il pulsante **Chiudi** permette di uscire dall'applicazione, la sua posizione è quella comune alle applicazioni più diffuse.

*(Assegna camera)*



L'interfaccia mostra i controlli necessari per assegnare una camera ad un cliente, in alto in corrispondenza dell'etichetta **Camera/e** vengono mostrate la/e stanza/e , selezionate in precedenza, da assegnare.

Nei campi **Nome** e **Cognome** viene inserito il nome e il cognome del cliente mentre dei campi **“dal”**, **“al”** viene inserita la data nel formato giorno/mese/anno (gg/mm/aaaa) per impostare l'arco di tempo per cui la camera è prenotata

In basso vengono posizionati i pulsanti di controllo, **Assegna** convalida i dati impostati nel form e li inserisce nel sistema, **Resetta** cancella il contenuto dai componenti della form e **Chiudi** chiude la finestra e ritorna alla finestra principale

*(Modifica camera)*



L'interfaccia mostra i controlli necessari per modificare i dati relativi ad una camera , in alto il campo **Camera/e** permette di cambiare la camera assegnata al cliente.

I campi **Nome** e **Cognome** permettono la modifica del nome e del cognome del cliente mentre i campi **dal** e **al** permettono di reimpostare i valori per l'arco di tempo in cui la camera avrà lo stato “occupato”.

In basso vengono posizionati i pulsanti di controllo, **Conferma** convalida i dati impostati nel form e li inserisce nel sistema, **Ripristina** imposta i dati nella form precedenti alla modifica effettuata, **Cancella** elimina i dati relativi alla camera dal sistema e imposta lo stato della camera a “libera”. **Chiudi** chiude la finestra e ritorna alla finestra principale.

## Appendice A – indice delle figure

---

1. Use Case Attività Extra	35
2. Use Case Servizio Bar	57
3. Use Case Servizio Ristorante	74
4. Use Case Servizio Camere	96
5. Package Diagram dei tre livelli	106
6. Presentation Layer Addetto Bar	108
7. Presentation Layer Addetto Ristorante	109
8. Presentation Layer Gestore	110
9. Presentation Layer Cliente	111
10. Package dell'Aplication Layer	112
11. Class diagram: Package Gestore	113
12. Class diagram: Package Addetto Bar	116
13. Class diagram: Package Addetto Ristorante	119
14. Class diagram: Package Cliente	122
15. Data Layer	126
16. Activity Diagram: Bar – Nuova Ordinazione	142
17. Activity Diagram: Ristorante – Inserimento Menù	143
18. Activity Diagram: Gestione Camere – Assegna Camera	144
19. Activity Diagram: Attività Extra – Cancellazione Attività	145
20. Activity Diagram: Bar – Modifica Ordinazione	146
21. Activity Diagram: Bar – Inserimento Prodotto	147
22. Activity Diagram: Gestione Camere – Libera Camera	148
23. Activity Diagram: Ristorante – Prenotazione Menù	149
24. Activity Diagram: Attività Extra – Nuova Prenotazione	150
25. Sequence Diagram: Gestione Camere – Assegna Camera	151

26. Sequence Diagram: Ristorante - Prenotazione Menu	152
27. Sequence Diagram: Bar – Nuova Ordinazione	153
28. Sequence Diagram: Attività Extra – Nuova Prenotazione	154
29. Sequence Diagram: Ristorante - Inserimento Menu	155
30. Sequence Diagram: Bar – Inserimento Prodotto	156
31. Sequence Diagram: Attività Extra – Inserimento Attività	157

## 5. Glossario

**Activity Diagram:** Diagramma che illustra un comportamento focalizzandosi sul lavoro eseguito, mostra una sequenza di azioni e gli oggetti coinvolti nell'esecuzione del lavoro.

**Attore:** Un insieme coerente di ruoli che gli utenti di un caso d'uso possono impersonare interagendo con il caso d'uso stesso. Un attore ha un ruolo per ciascun caso d'uso con il quale comunica.

**Autenticazione:** Verifica di autorizzazione all'accesso, identificazione mediante nome utente e password usata per la connessione a server con limitazioni d'accesso.

**Boundary object:** Una classe stereotipata i cui oggetti vengono usati per comunicare con gli attori usata per modellare le interazioni tra il sistema e i suoi attori.

**Caso d'uso:** descrizione di un'interazione tra il sistema e uno o più attori assieme alle attività eseguite dal sistema stesso.

**Class Diagram:** diagramma che presenta una visione statica di un insieme di classi e delle loro relazioni, illustra le operazioni e gli attributi di ciascuna classe così come i vincoli sulle relazioni tra oggetti.

**Classe:** descrizione di un gruppo di oggetti con proprietà simili e comportamenti comuni.

**Control object:** oggetto che ha il compito di controllare e realizzare le operazioni.

**Entity object:** oggetto usato per modellare dati persistenti del sistema.

**Form:** finestra di dialogo incorporata in una pagina Web che consente all'utente di inserire informazioni destinate ad un server. Generalmente richiede un programma sul server che si occupi di esaminare le informazioni inviate. E' composto da spazi (campi) predefiniti, ad esempio menù a tendina, elenchi puntati o caselle di testo libero.

**Interfaccia utente:** Insieme delle procedure e dei comandi che consentono ad un utente di interagire facilmente con un programma. I comandi possono essere impartiti attraverso la tastiera per la digitazione o mediante dispositivi di puntamento come il mouse nelle interfacce grafiche.

**Login:** Procedura attraverso la quale ci si collega con un qualsiasi servizio in linea. All'utente viene assegnato un nome di login ed una password che vengono richiesti dal sistema ogni volta che ci si collega.

**Logout:** Procedura che consente ad un utente di terminare la propria attività in un'area riservata di un certo sito.

**Mock-up:** Rappresentazione di esempio per alcune pagine tipiche dell'applicazione.

**Modello:** Una descrizione astratta di un sistema. Un modello viene espresso attraverso diagrammi e i diagrammi attraverso un linguaggio di modellazione.

**Navigation Path:** Il processo del muoversi da un nodo ad un altro in un ipertesto (o web). La navigazione avviene seguendo i collegamenti (links). I comandi disponibili in un particolare

browser possono rendere più facile la navigazione. In particolare ciò avviene mantenendo la storia dei collegamenti attivati (in alcuni casi indicando graficamente una mappa delle relazioni tra essi).

**Oggetto:** Un concetto, astrazione o cosa che può essere identificata individualmente e ha significato per una applicazione. Un oggetto è una istanza di una classe.

**Package:** Un raggruppamento di elementi (classi, associazioni, generalizzazioni e altri package) basato su un tema comune.

**Password:** metodo di sicurezza che, mediante una stringa di caratteri, permette di identificare un utente specifico. Generalmente le password sono formate da una sequenza di lettere e numeri; digitando correttamente questi caratteri, si può avere accesso al computer o alla rete.

**R.A.D.:** acronimo di Requirements Analysis Document (Documento di Analisi dei Requisiti)

**Requisiti Funzionali:** descrizione dell'interazione tra il sistema e l'ambiente esterno.

**Requisiti Non Funzionali:** descrizione di proprietà e vincoli di sistema.

**Scenario:** interazione possibile tra il sistema e un attore. L'interazione viene descritta come un insieme di messaggi, rappresenta un' "istanza" di un caso d'uso.

**Sequence Diagram:** diagramma che descrive di un'interazione tra oggetti come sequenza temporale di azioni, In ascissa sono disposti vari oggetti (istanze specifiche) e l'ordinata rappresenta il tempo.

**Statechart Diagram:** diagramma che contiene informazioni sullo stato e le transizioni di stato di un oggetto, descrive o rappresenta graficamente i cambiamenti di stato di un oggetto.

**Unified Modeling Language (UML):** insieme di linguaggi che, utilizzati congiuntamente, consentono di descrivere/modellare aspetti diversi di un sistema con un approccio Object oriented.

**Use case diagram:** Un modello dei casi d'uso descritto da un diagramma che contiene elementi del sistema, attori e i loro casi d'uso, e mostra le diverse relazioni tra questi elementi.

**Username:** Nome dell'utente, viene richiesta all'atto di autenticazione al sistema in combinazione con una password.



# Università degli Studi di Salerno

*Corso di Ingegneria del Software*

---

## Quick Service

*System Design Document*

*quick* -  
*Service*

*Data <5/05/2008>*

## **MEMBRI DEL QUICK SERVICE:**

### **Team Members:**

*Vicidomini Vincenzo*

*Mercurio Antonio*

*Iacoletti Alessandro*

*Pacifico Marta*

*Pizza Luca Ernesto*

# INDICE DEI CONTENUTI

<b>1. INTRODUZIONE .....</b>	<b>4</b>
1.1 SCOPO DEL SISTEMA .....	4
1.2 OBIETTIVI DI DESIGN.....	4
1.3 DEFINIZIONI, ACRONIMI ED ABBREVIAZIONI.....	6
1.4 RIFERIMENTI .....	6
1.5 OVERVIEW .....	7
<b>2. SISTEMA SOFTWARE CORRENTE .....</b>	<b>7</b>
<b>3. SISTEMA SOFTWARE PROPOSTO .....</b>	<b>8</b>
3.1 OVERVIEW .....	8
3.2 DECOMPOSIZIONE IN SOTTOSISTEMI .....	11
<b>ERRORE. IL SEGNALIBRO NON È DEFINITO.</b>	
3.3 HARDWARE/SOFTWARE MAPPING .....	12
<b>ERRORE. IL SEGNALIBRO NON È DEFINITO.</b>	
3.4 GESTIONE DATI PERSISTENTI .....	18
3.5 CONTROLLO DEGLI ACCESSI E SICUREZZA .....	18
3.6 CONTROLLO GLOBALE DEL SOFTWARE.....	19
3.7 CONDIZIONI BOUNDARY .....	20
<b>4 SERVIZI DEI SOTTOSISTEMI.....</b>	<b>22</b>
<b>5. GLOSSARIO .....</b>	

# **1. Introduzione**

## **1.1 Scopo del sistema**

Gli obiettivi del sistema sono di realizzare un software per la gestione del servizio in camera presso una struttura.

Il progetto QUICK TOUCH SERVICE si adatta per le strutture che vogliono consentire una gestione facile, comoda e completa dei servizi volti alla clientela. Con un semplice click del mouse, o meglio ancora con una semplice pressione del monitor (touch screen), e con la rappresentazione grafica dei prodotti.

Il sistema sarà in grado di incrementare l'efficienza di un albergo, mediante meccanismi di gestione che si basano sull'utilizzo di un archivio centrale dove sono organizzate le informazioni.

In fase di design del software è stato necessario compiere delle scelte riguardanti i programmi da utilizzare per lo sviluppo del progetto.

## **1.2 Obiettivi di Design**

I punti fondamentali che abbiamo preso in considerazione sono:

- Il linguaggio di programmazione;
- Portabilità;
- Interfaccia grafica;
- Base di dati;
- Prestazioni;

### **Il Linguaggio di Programmazione**

In funzione della fase di sviluppo sarebbero preferibili una sintassi semplice e flessibile, semplicità e potenza nell'elaborazione di testo, disponibilità di documentazione e possibilità d'estensione ed interfacciamento verso altri linguaggi (JAVA).

## **Portabilità**

Trattandosi di un sistema software gestionale di alto livello è desiderabile una marcata indipendenza dal sistema operativo. Peraltro non vi sono necessità imprescindibili che pongano ostacoli nel design di un sistema software portabile. Inoltre, sebbene il design e l'implementazione di un software multi-piattaforma possano risultare più lunghi e complessi, è a nostro avviso importante compiere subito un deciso passo verso un sistema di questo genere.

## **Interfaccia Grafica**

Il sistema sarà semplificato da interfacce amichevoli, intuitive e di semplice utilizzo.

L'interfaccia utente sarà realizzata in modo da renderne immediato l'utilizzo anche da parte degli utenti meno esperti. Tutte le informazioni vengono presentate in maniera chiara ed intuitiva.

## **Base di Dati**

Sia l'implementazione di un sistema proprio sia l'interfacciamento ad un DBMS presentano ovvi vantaggi e svantaggi. Considerando condizioni operative normali, con svariati record di diverso tipo memorizzati e la necessità di condividere la base di dati fra più client l'unica scelta possibile risulta l'interfacciamento con un DBMS.

## **Prestazioni**

Ovviamente trattandosi di un software interattivo è necessaria una buona velocità di risposta alle richieste dell'utente. D'altra parte ipotizzando un ambiente di utilizzo tipico che coinvolge una base di dati condivisa, accessibile via rete, non bisogna trascurare la latenza ad essa dovuta, quindi è inutile esasperare l'aspetto prestazionale. E' comunque desiderabile che l'utilizzo del software non sia precluso alle macchine di fascia più bassa.

## **1.3 Definizioni, Acronimi ed Abbreviazioni**

**QUICK TOUCH SERVICE:**Sistema software per la gestione del servizio in camera presso una struttura.

**JDBC:** Java DataBase Connection;

**DBMS:** Database Management System, Sistema di gestione del database.

**Java:** linguaggio di programmazione orientato agli oggetti.

## **1.4 Riferimenti**

- **Object Software Engineering – Using UML, Patterns and Java** *B. Bruegge, A. H. Dutoit.*
- **Rad**
- **SoftwareProjectManagementPlan**

## 1.5 Overview

Nei successivi paragrafi del documento di System Design verranno descritti:

- **Sistema Software Corrente:** rappresenta l'architettura del software che è stato sostituito o, se esso non esiste, descrive l'architettura dei software per sistemi simili esistenti.
- **Sistema Software proposto:** documenta il modello di System Design del nuovo sistema,in cui vengono descritti i seguenti elementi:
  - Decomposizione in sottosistemi, nella quale viene descritta la suddivisione del sistema in vari sottosistemi. Le operazioni che essi svolgono vengono descritte ad alto livello senza entrare troppo nello specifico.
  - Hardware/Software mapping, in cui vengono prese alcune decisioni per quanto riguarda le piattaforme hardware e software su cui il sistema dovrà girare,una volta decise le piattaforme è necessario mappare le componenti su di esse. Questa operazione potrebbe portare all'introduzione di nuove componenti per interfacciare i sottosistemi su diverse piattaforme.
  - Gestione Dati Persistenti, in cui sono identificati gli oggetti persistenti e scelto il tipo di infrastruttura da usare per memorizzarli.
  - Controllo degli Accessi e Sicurezza, che descrive il modello utente del sistema in termini di matrici di accesso e i problemi di sicurezza, come la scelta di un meccanismo di autenticazione.
  - Controllo Globale del Software, che descrive come il controllo globale del software è implementato, come le procedure di richiesta sono avviate e come si sincronizzano i sottosistemi.
  - Condizioni Boundary in cui sono descritte le condizioni limite del sistema,le quali includono lo start-up, lo shutdown, l'inizializzazione e la gestione di fallimenti come corruzione di dati, caduta di connessione e caduta di componenti;a tale scopo saranno elaborati dei casi d'uso che specificano la sequenza di operazioni in ciascuno dei casi sopra elencati.
- **Servizi dei Sottosistemi:** descrive i servizi forniti da ciascun sottosistema in termini di operazioni.

## 2. Sistema Software Corrente

Attualmente esistono sistemi software con caratteristiche simili all'applicazione QUICK TOUCH SERVICE, ma il nostro scopo è quello di mirare alla creazione di un software che faciliti ulteriormente la gestione di strutture e infrastrutture strettamente connesse con le attività turistiche.

## **3. Sistema Software Proposto**

### **3.1 Overview**

Il tipo di architettura software che si è scelta è legata (e vincolata) all'analisi del sistema. Infatti, il sistema che si vuole realizzare è un sistema distribuito nel quale esiste un elemento centrale contenente l'archivio dati della libreria a cui i vari attori dalle loro postazioni vogliono accedere. Quindi esistono dei moduli client che richiedono dati ad un modulo centrale che rappresenta il server. Fra loro i vari moduli non comunicano: l'unica comunicazione è rivolta da e verso il server. Naturalmente tali moduli risiedono in macchine differenti.

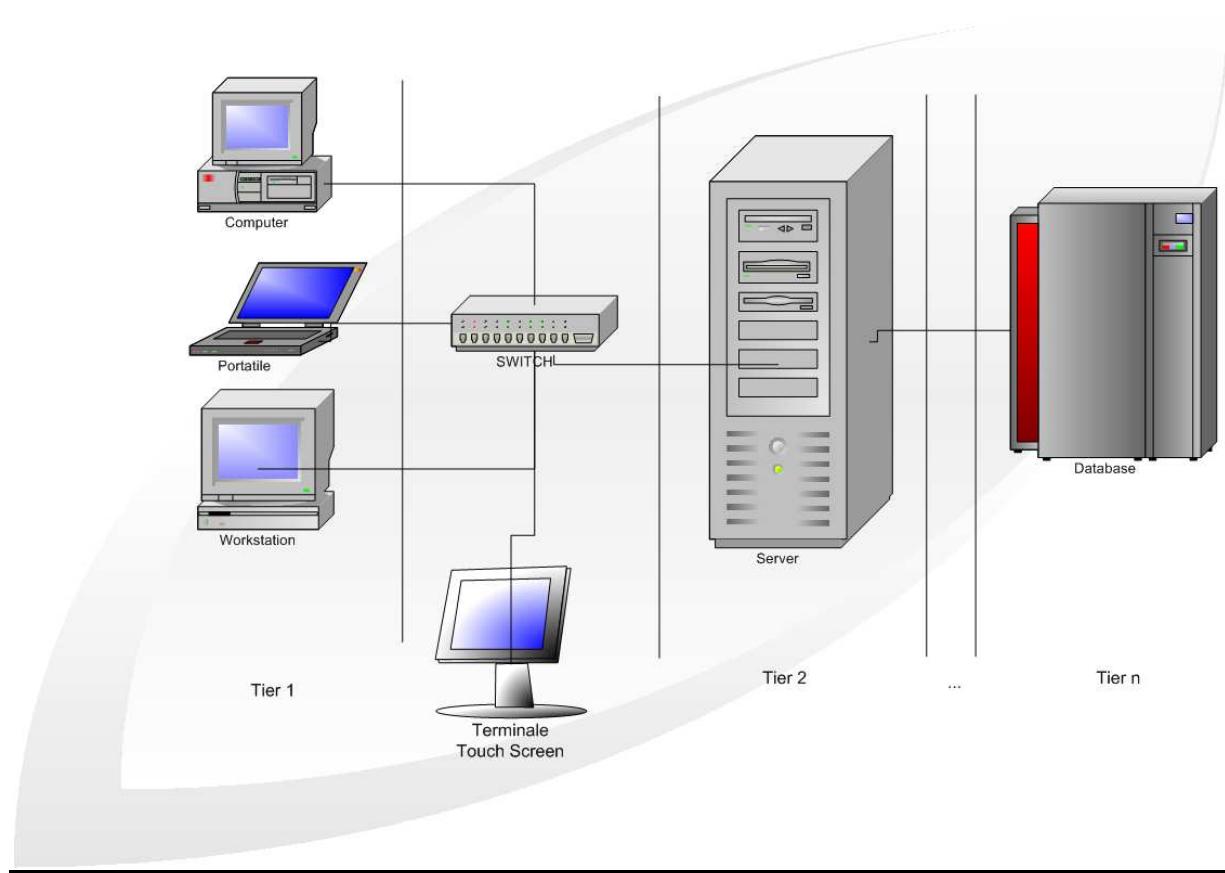
L'architettura scelta è pertanto quella client - server. Il server a sua volta fornisce la comunicazione dei dati con il database.

La comunicazione tra client e server avviene utilizzando l'architettura Three-Tier, evoluzione dell'architettura client/server.

Con riferimento a tali sistemi, infatti, possiamo osservare che esse bene si adattano al nostro sistema di accesso ai dati.

La necessità di accesso da parte del personale della libreria al sistema informativo sta alla base dell'organizzazione n-tier. Un'architettura *n-tier* è costituita da *n* sistemi che contribuiscono in successione, a portare a termine una computazione.

Nella Figura seguente è schematizzata tale struttura.



Tra ogni coppia di livelli esiste una relazione *client - server* e l'interazione viene propagata dal primo all'ultimo livello dell'architettura a viceversa. In ogni passo sono coinvolti programmi che si occupano di rispondere alle richieste eventualmente appoggiandosi ai livelli successivi. In ogni *tier* possono essere coinvolti più calcolatori. Le richieste vengono elaborate dai vari *tier* fino al loro completamento.

Il modello *n-tier* si è affermato proprio perché riesce a identificare "isole" di calcolo che possono essere associate alle reti su cui risiedono le risorse su cui un programma opera.

Le architetture *n-tier* riguardano la struttura di un sistema che è composto di elementi che nel loro piccolo sono del tutto standard. La ricchezza che deriva dalla struttura è principalmente una migliore ingegnerizzazione del sistema che porta ad una sua maggiore manutenibilità. Inoltre la conoscenza delle interfacce tra un livello e l'altro aiuta a rendere più flessibile il sistema, ad esempio accedendo più DB invece che uno solo per distribuire il carico senza che il livello intermedio si accorga del cambiamento. Le applicazioni *three-tier* ( $n=3$ ) prevedono un primo livello

costituito dai clients, questi si connettono al secondo livello in cui vengono elaborate le richieste. Nel livello intermedio si gestisce l'elaborazione della richiesta accedendo a dati che risiedono sul terzo ed ultimo livello.

## **Client + Front-end + Back-end = three-tier**

È importante osservare come l'interfaccia tra i livelli sia ben definita: questo aiuta a effettuare verifiche di sicurezza sulle comunicazioni. Inoltre è possibile separare il livello dei clients dai restanti due utilizzando un *firewall*.

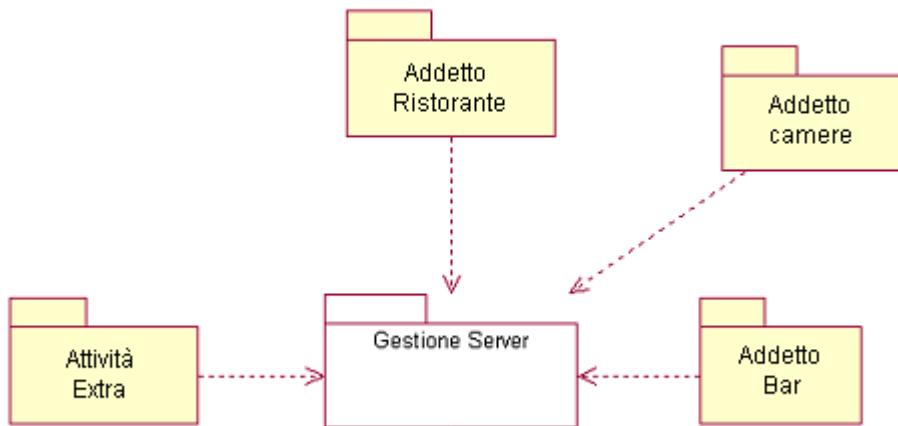
In questo progetto abbiamo realizzato una struttura 3-tier in cui c'è un database, un client ed un server che interagisce sia con il database sia con il client e che svolge il compito di modulo intermedio. Tale modulo intermedio, (rappresentato per noi dalla classe Server e dai driver JDBC-ODBC di Java), si occupa di inoltrare le richieste dell'utente al livello archivio dati e di trasmettere i relativi risultati.

Il principale vantaggio di tale scelta è di rendere il client del tutto indipendente dalle problematiche di accesso ai dati, rendendo il sistema aggiornabile e mantenibile soprattutto per quel che riguarda la parte del sistema che si occupa dell'accesso all'archivio dei dati.

Resta da definire il tipo di protocollo utilizzato per la comunicazione client - server. Si è scelto l'uso del protocollo UDP. Questo protocollo offre un servizio di comunicazione fra client e server senza connessione evitando così la fase di handshaking.

Il sistema è composto da sei componenti principali. Riportiamo il diagramma dei package principali del sistema ed una breve descrizione di ogni package.

### 3.2 Decomposizione dei sottosistemi



I sottosistemi indicati sono in esecuzione su macchine diverse (La parte cliente è replicata su più terminali disposti nelle camere, mentre le altre parti sono disposte nei vari settori dell'albergo) e comunicano (tutti col sottosistema server) in maniera remota.

Il sistema è stato diviso *orizzontalmente* in tre strati:

- Sottosistema **Presentation Layer**: visualizza all'utente gli oggetti del dominio applicativo. Fanno parte di questo strato gli oggetti Boundary.
- Sottosistema **Application Layer**: responsabile della sequenza di interazioni con l'utente e di notificare ai Presentation Layer i cambiamenti nel modello. Fanno parte di questo strato gli oggetti Control.
- Sottosistema **Data Layer**: mantiene la conoscenza del dominio applicativo. Fanno parte di questo strato gli oggetti Entity.

Il sottosistema **Presentation Layer** è stato diviso in quattro sottosistemi principali. Le partizioni fanno riferimento ai quattro principali attori che accedono al sistema:

- Sottosistema **AddettoBar**: Include tutte le interfacce grafiche a cui l'Addetto Bar può accedere e cioè: visualizza\_ordinazione, modifica\_ordinazione, nuovo\_ordine, cancella\_ordine, visualizza\_listino, inserimento\_prodotto, modifica\_prodotto, cancella\_prodotto.
- Sottosistema **AddettoRistorante**: Include tutte le interfacce grafiche l'Addetto Ristorante può accedere e cioè: annullamento\_ordinazione, visualizza\_ordinazione,

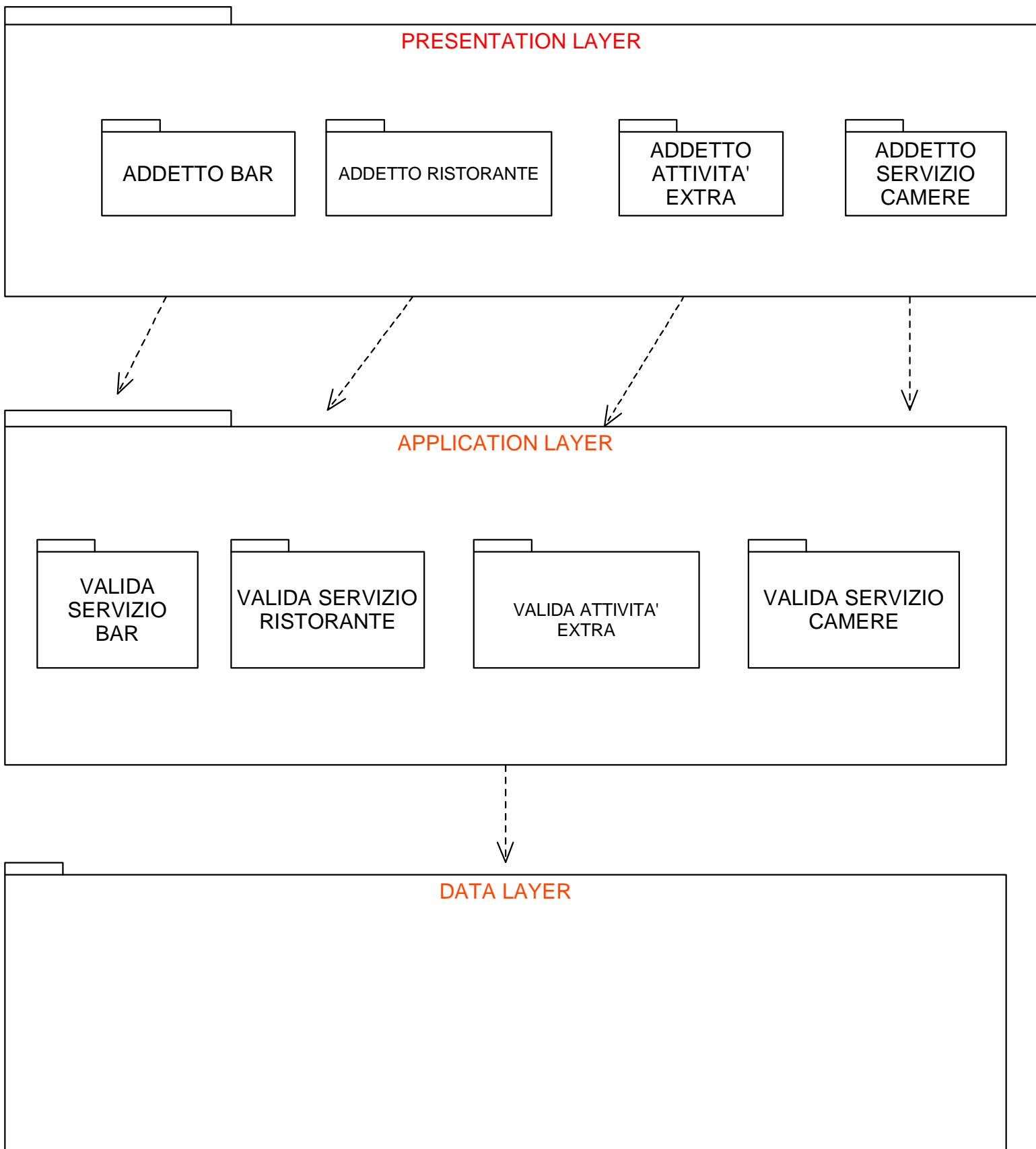
modifica\_ordinazione, prenotazione\_menu, visualizza\_saldo, visualizza\_menu, modifica\_menu, inserimento\_menu, consultazione\_prenotazione.

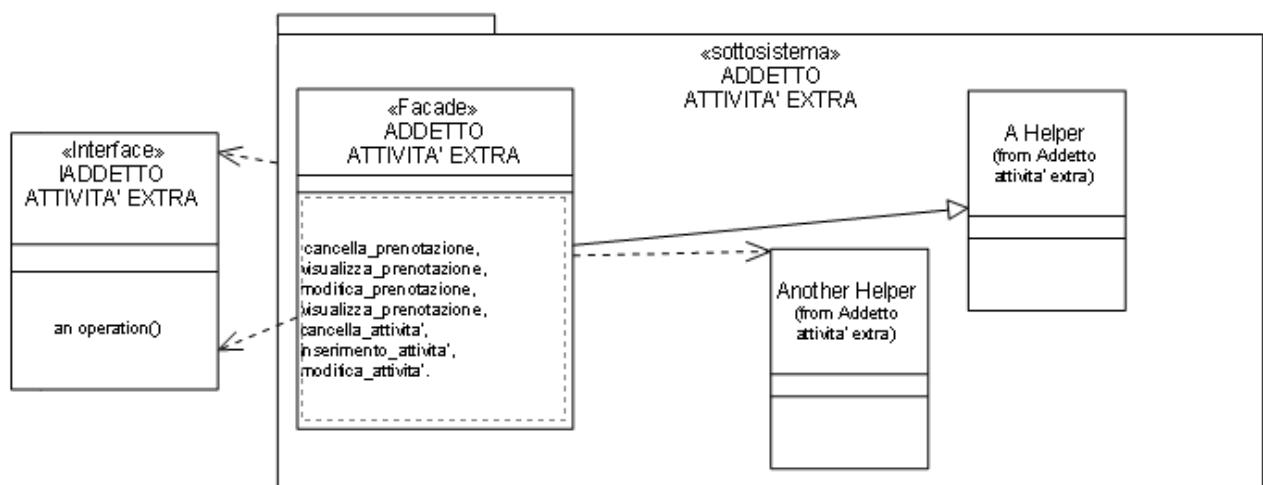
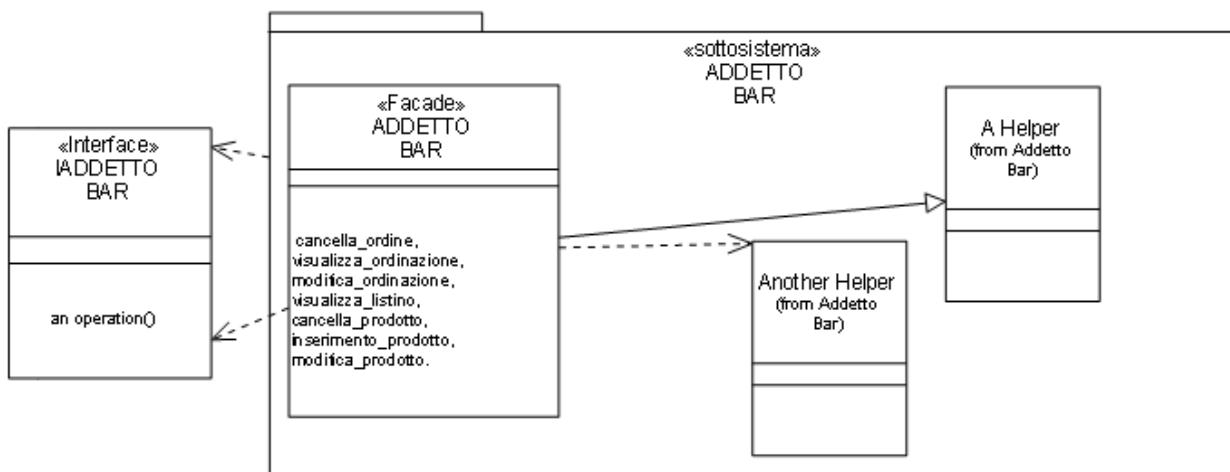
- Sottosistema **AddettoAttivitàExtra**: Include tutte le interfacce grafiche a cui l'Addetto Attività Extra può accedere e cioè: annullamento\_prenotazione, visualizza\_prenotazione, modifica\_prenotazione, nuova\_prenotazione, resoconto\_prenotazione, modifica\_attivita, cancella\_attivita, inserimento\_attivita.
- Sottosistema **AddettoServizioCamere**: Include tutte le interfacce grafiche a cui l'Addetto Servizio può accedere e cioè: libera\_camera, visualizza\_camera, modifica\_camera, assegna\_camera.

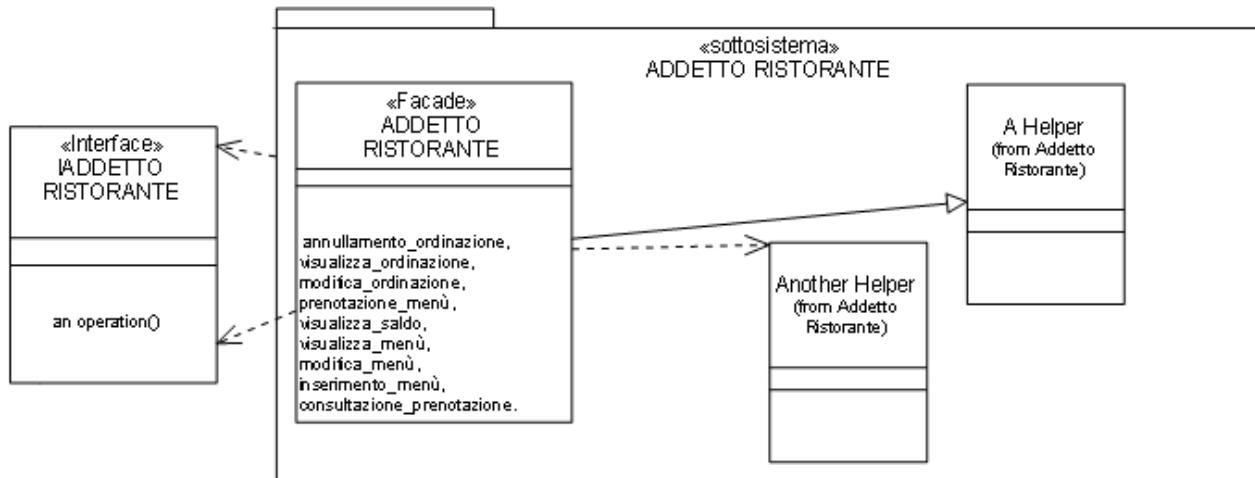
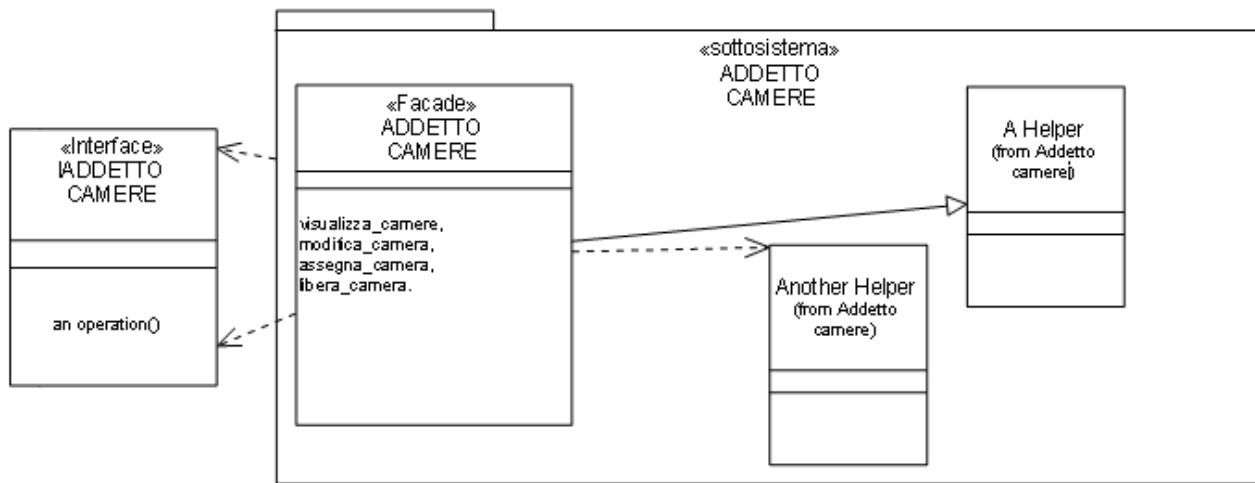
Il sottosistema **Application Layer** è stato diviso in cinque sottosistemi principali. Le partizioni fanno riferimento alle cinque principali entità a cui l'utente accede:

- Sottosistema **ValidaServizioBar**: In esso sono incluse tutte le unità di controllo che accedono all'entità Addetto Bar e cioè: login, logout, visualizza/modifica prodotti, aggiungi prodotti, cancella prodotto.
- Sottosistema **ValidaServizioRistorante**: In esso sono incluse tutte le unità di controllo che accedono all'entità Addetto Ristorante e cioè: login, logout, visualizza/modifica menù, aggiungi menù, cancella menù.
- Sottosistema **ValidaAttivitàExtra**: In esso sono incluse tutte le unità di controllo che accedono all'entità Addetto Attività Extra e cioè: login, logout, visualizza/modifica attività, aggiungi attività, cancella attività.
- Sottosistema **ValidaServizioCamere**: In esso sono incluse tutte le unità di controllo che accedono all'entità Addetto Camere e cioè: login, logout, visualizza/modifica prenotazione camera, cancella prenotazione camera.

La decomposizione in sottosistemi descritta è illustrata nella seguente figura







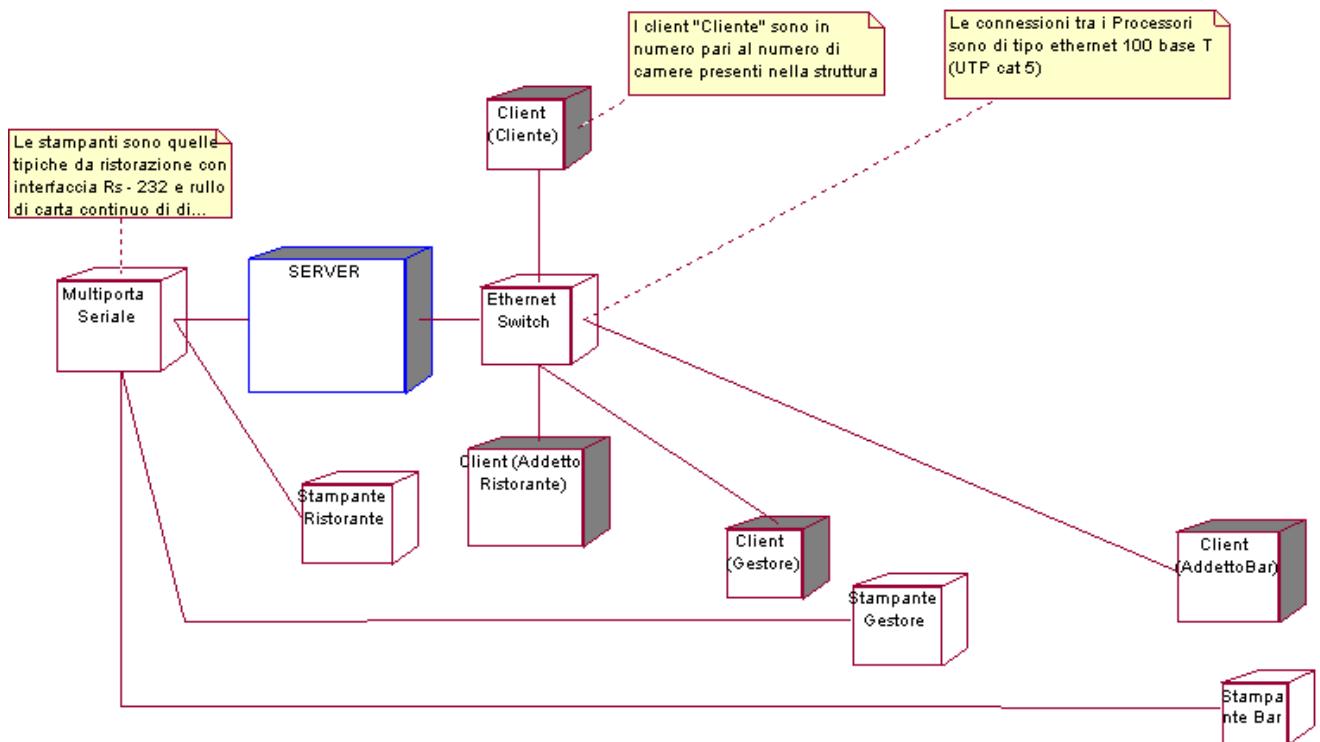
### 3.3 Hardware-Software mapping

L'intero sistema risulta essere composto da quattro componenti software principali: le cinque interfacce grafiche, le cinque interfacce di controllo, la gestione comunicazione e la gestione dati. Tali componenti non risiedono sulla stessa macchina ma su macchine differenti.

(Le interfacce Grafiche e le interfacce di controllo illustrate in ciascun package sono accoppiate ovviamente su una sola macchina).

I componenti sono inoltre indipendenti gli uni dagli altri tranne che per il server che è l'unico componente che comunica con gli altri. A parte questa comunicazione gli altri componenti sono fra loro isolati, ovvero non si scambiano messaggi direttamente, ma alcune delle componenti possono ricevere dal SERVER informazioni sullo stato di altre. Mostriamo di seguito come tale struttura venga modellata mediante l'utilizzo del diagramma di deployment.

#### Diagramma di Deployment del sistema.



L'intero programma è gestito da un server al quale sono collegati quattro client e le rispettive stampanti.

I client sono collegati al server tramite uno “Switch Ethernet” che collega i processori dell’Addetto Ristorante, dell’Addetto Bar, del Gestore e dei Clienti della struttura alberghiera (uno per ogni camera d’albergo). I compiti principali sono svolti dal server che ha il compito di immagazzinare dati, gestire le ordinazioni e le varie operazioni svolte dagli impiegati dell’albergo e soprattutto dei clienti.

Come detto in precedenza, al server è inoltre collegata una “Multiporta seriale” che permette di collegare le varie stampanti utili agli addetti e al gestore per avere resoconti e informazioni riguardanti ad esempio il resoconto delle prenotazioni per un’attività proposta dall’albergo o il saldo di un cliente.

Le stampanti sono quelle tipiche per strutture di questo tipo con interfaccia “RS 232” e rullo di carta continuo che rende più semplice e immediata la stampa delle informazioni.

## 3.4 Gestione Dati Persistenti

I dati persistenti del sistema rappresentano le informazioni relative al sottosistema Storage. Esso contiene i DatiCliente, DatiUtente, DatiCamere, DatiProdottiBar, DatiAttivitàExtra, DatiMenù, DatiPortateMenù, DatiTipoPortate, DatiCategorie, DatiPrenotazione, DatiDettagliPrenotazioneAttivitàExtra, DatiDettaglioPrenotazioneBar, DatiDettaglioPrenotazioneRistorante.

Per la memorizzazione di tali dati viene scelto un database relazionale. La scelta è giustificata dal fatto che sugli attributi vengono effettuate un numero cospicuo di query complesse.

## 3.5 Controllo degli Accessi e Sicurezza

L'utente deve essere autenticato, e l'operazione di login viene eseguita tramite la compilazione del form che compare all'accensione della macchina. Nei casi di login fallito, ogni sottosistema ripropone il form di login. Dato che i compiti svolti dai vari "attori" nel nostro sistema sono differenti ed accedono ad informazioni diverse dell'archivio al fine di evitare:

- la visualizzazione di dati specifici per un certo compito e non utilizzati per altre mansioni;
- la modifica di dati fondamentali per la gestione dell'albergo stesso;
- e che le attività svolte dagli impiegati dell'albergo non possano essere ricondotte ad una specifica persona;

Si è pensato di utilizzare più meccanismi di protezione dei dati e delle attività eseguite.

***Il primo livello di protezione*** è un filtro che si è realizzato nell'accesso e nella modifica dei dati e che viene posto a livello delle funzionalità offerte ad ogni impiegato dal sistema in esecuzione sul proprio terminale, secondo il seguente schema:

1. **Cliente**: può accedere solo in lettura ai dati relativi ai servizi dell'albergo.

Può però anche inserire nuovi dati nel database quando ad esempio decide di ordinare qualcosa al bar oppure desidera prenotarsi per un'escursione

2. **Addetto Bar/Ristorante:** può accedere ai dati del magazzino per visualizzare tutti i dati relativi ai prodotti, agli ordini emessi, all'insieme di tutte le transazioni ed al menu ristorante.

Si è scelto di dare all'addetto bar/ristorante delle funzioni di sola consultazione per isolare l'insieme delle sue attività da quelle di modifica dei dati.

3. **Gestore:** può accedere ad ogni informazione dell'archivio dati per consultazione e per modifiche.

***Un secondo livello di protezione*** viene garantito durante la fase di accesso al sistema dalla necessità di dover inserire una password ed il proprio ID.

Questa operazione preliminare deve essere svolta da tutti gli utilizzatori del sistema ad eccezione del cliente.

L'utilizzo delle password insieme all'ID garantisce inoltre la possibilità di poter ricondurre ogni operazione di modifica dei dati ad un impiegato.

La gestione delle password inoltre viene effettuata solamente dal Gestore.

## 3.6 Controllo Globale del Software

Si è ritenuto opportuno utilizzare una gestione del *controllo centralizzato* di tipo **event-driven**. Il sistema prevede una serie di bottoni, form e notifiche che realizzano l'interfaccia grafica. Prevede l'accesso solamente ai terminali predisposti su cui è installato il client di QuickService e collegati alla rete locale, ovvero i terminali installati nelle camere e le postazioni del bar, del ristorante e del gestore. L'accesso a funzionalità di inserimento, modifica e cancellazione di servizi offerti quali prodotti del bar, menu del ristorante, attività extra o gestione camere è consentita solamente dopo che l'utente ha effettuato l'autenticazione, questo assicura che tali operazioni sono svolte solamente dal personale autorizzato.

Per effettuare l'autenticazione il sistema mostra una maschera che richiede l'inserimento del login name e della password.



l'utente inserisce i dati e preme il bottone per la convalida, viene invocato il metodo remoto di convalida che prende i dati inseriti dall'utente controlla la corrispondenza nel database e consente l'accesso se i dati sono corretti altrimenti l'accesso sarà negato.

Per il cliente non è prevista alcuna autenticazione, infatti, può accedere alle funzionalità del sistema in qualsiasi momento, l'utente è riconosciuto in base all'identificativo del client presente nella camera.

## 3.7 Condizioni Boundary

### • Inizializzazione

Data la natura distribuita del sistema proposto l'avvio e l'esecuzione delle sue varie parti ossia dei sottosistemi che forniscono le varie funzionalità ai vari “attori” può avvenire indipendentemente.

I sottosistemi quindi non hanno bisogno di accedere a nessun dato, poiché i client conoscono già la loro identità nel sistema.

All'avvio, se si sta avviando un sottosistema che riguardante la gestione (Gestione Ordinazioni Ristorante, Gestione Prenotazioni Attività extra, ecc...), il sistema richiederà una login ed una password e poi alla visualizzazione principale; altrimenti, se l'utente è il cliente dell'albergo, si aprirà subito la visualizzazione principale.

L'interfaccia grafica è molto semplice ed intuitiva, da questa si può accedere a tutti i servizi offerti a quel determinato utente, infatti, essa varia a seconda dei servizi offerti, che sono diversi per ogni tipo di utente (Addetto bar, Addetto Camere, Cliente).

## • Terminazione

Per il corretto funzionamento di un sottosistema non è richiesto che gli altri siano attivi, quindi se uno di essi termina non intralcerrebbe il lavoro degli altri sottosistemi fatta eccezione che per il server. E' richiesto, infatti, che sia prima avviato il server affinché una qualunque postazione client possa usufruire dei servizi offerti da esso. Sebbene questa sia una condizione indispensabile per il corretto funzionamento del sistema (nella sua interezza o in parte) se si verifica l'avvio di un sottosistema senza che il server sia attivo si avrà comunque una gestione di tale situazione da parte del sottosistema stesso.

La terminazione si realizza soltanto quando tutti i sottosistemi sono stati disattivati, infatti, il sistema resta comunque attivo anche se solo due sottosistemi (di cui uno è il server) sono ancora in esecuzione.

La disattivazione di un singolo sottosistema avviene mediante la funzione di exit presente su tutte le varie postazioni. Questa funzionalità sulla postazione del cliente, ha come unico effetto la chiusura del suo terminale.

Essendo quindi ogni sottosistema indipendente, non c'è bisogno di notificare a gli altri che uno di essi è stato terminato, fatta eccezione per il server, infatti, questa situazione genererebbe errore in tutti gli altri.

## • Fallimento

Quando avviene un blocco generale del sistema, esso viene riavviato, quindi viene effettuata una nuova inizializzazione. Non c'è rischio di perdite di dati, poiché tutte le comande in corso vengono gestite non appena esse sono confermate dall'utente.

Quando invece avviene il collasso di un nodo o di una maglia di comunicazione, il sistema lancia un avviso di errore, il quale oltre ad avvisare l'utente della mancata esecuzione di un'operazione, da la possibilità di riprovare la stessa in un secondo momento.

### 3 Servizi dei Sottosistemi

#### 3.1. Application Layer

##### 3.1.1. ValidaServizioBar

Tabella 1: Descrizione e servizi del sottosistema **ValidaServizioBar**

<b>Sottosistema: ValidaServizioBar</b>	
<b>Descrizione</b>	Sottosistema appartenente alla stratificazione <i>Application Layer</i> , contiene un interfaccia che gestisce l'accesso e le operazioni comuni a tutti gli utenti.
<b>Servizi del sottosistema</b>	
<b>Operazioni</b>	<b>Comportamento</b>
<b>Login()</b>	Operazione che offre la possibilità a un Utente Registrato di accedere alla sezione ad egli riservata.
<b>LogOut ()</b>	Operazione che offre la possibilità a un Utente Registrato di uscire dal sistema
<b>VisualizzaProdotti ()</b>	Operazione che offre la possibilità a un Utente Registrato di visualizzare i prodotti del bar
<b>ModificaProdotti ()</b>	Operazione che offre la possibilità ad un Utente Registrato di modificare i prodotti.
<b>AggiungiProdotti ()</b>	Operazione che offre la possibilità ad un Utente Registrato di aggiungere i prodotti.
<b>CancellaProdotti ()</b>	Operazione che offre la possibilità ad un Utente Registrato di cancellare i prodotti.

### 3.1.2. ValidaServizioRistorante

Tabella 2: Descrizione e servizi del sottosistema **ValidaServizioRistorante**

<b>Sottosistema: <i>ValidaServizioRistorante</i></b>	
<b>Descrizione</b>	Sottosistema appartenente alla stratificazione <i>Application Layer</i> , contiene un interfaccia che gestisce l'accesso e le operazioni comuni a tutti gli utenti.
<b>Servizi del sottosistema</b>	
<b>Operazioni</b>	<b>Comportamento</b>
<b>Login()</b>	Operazione che offre la possibilità a un Utente Registrato di accedere alla sezione ad egli riservata.
<b>LogOut ()</b>	Operazione che offre la possibilità a un Utente Registrato di uscire dal sistema
<b>VisualizzaMenù()</b>	Operazione che offre la possibilità a un Utente Registrato di visualizzare i menù del ristorante.
<b>ModificaMenù ()</b>	Operazione che offre la possibilità ad un Utente Registrato di modificare i menù
<b>AggiungiMenù ()</b>	Operazione che offre la possibilità ad un Utente Registrato di aggiungere i menù
<b>CancellaMenù ()</b>	Operazione che offre la possibilità ad un Utente Registrato di cancellare i menù.

### 3.1.3. ValidaAttivitàExtra

**Tabella 3: Descrizione e servizi del sottosistema ValidaAttivitàExtra**

<b>Sottosistema: ValidaAttivitàExtra</b>	
<b>Descrizione</b>	<b>Servizi del sottosistema</b>
<b>Operazioni</b>	<b>Comportamento</b>
<b>Login()</b>	Operazione che offre la possibilità a un Utente Registrato di accedere alla sezione ad egli riservata.
<b>LogOut ()</b>	Operazione che offre la possibilità a un Utente Registrato di uscire dal sistema
<b>VisualizzaAttività()</b>	Operazione che offre la possibilità a un Utente Registrato di visualizzare le attività extra.
<b>ModificaAttività ()</b>	Operazione che offre la possibilità ad un Utente Registrato di modificare le attività
<b>AggiungiAttività ()</b>	Operazione che offre la possibilità ad un Utente Registrato di aggiungere le attività
<b>CancellaAttività ()</b>	Operazione che offre la possibilità ad un Utente Registrato di cancellare le attività

### 3.1.4. ValidaServizioCamere

**Tabella 4: Descrizione e servizi del sottosistema ValidaServizioCamere**

<b>Sottosistema: ValidaServizioCamere</b>	
<b>Descrizione</b>	<b>Servizi del sottosistema</b>
<b>Operazioni</b>	<b>Comportamento</b>
<b>Login()</b>	Operazione che offre la possibilità a un Utente Registrato di accedere alla sezione ad egli riservata.
<b>LogOut ()</b>	Operazione che offre la possibilità a un Utente Registrato di uscire dal sistema
<b>VisualizzaPrenotazioneCamera()</b>	Operazione che offre la possibilità a un Utente Registrato di visualizzare la prenotazione delle camere.
<b>ModificaPrenotazioneCamera ()</b>	Operazione che offre la possibilità ad un Utente Registrato di modificare la prenotazione delle camere.
<b>AggiungiPrenotazioneCamera ()</b>	Operazione che offre la possibilità ad un Utente Registrato di aggiungere la prenotazione delle camere.
<b>CancellaPrenotazioneCamera ()</b>	Operazione che offre la possibilità ad un Utente Registrato di cancellare la prenotazione delle camere.

### 3.2. Data Layer

**Tabella 5: Descrizione e servizi del sottosistema Data Layer**

<b>Sottosistema: Data Layer</b>	
<b>Descrizione</b>	Sottosistema contenente un interfaccia che accede ai dati persistenti dell'applicazione
<b>Servizi del sottosistema</b>	
<b>Operazioni</b>	<b>Comportamento</b>
<b>DatiCliente ()</b>	Operazione che contiene le informazioni sui clienti che accedono al sistema per effettuare la prenotazione e consente anche la modifica e la creazione di clienti.
<b>DatiUtente ()</b>	Operazione che serve per memorizzare le registrazioni di vari addetti che gestiscono il sistema di prenotazione
<b>DatiCamere ()</b>	Operazione che contiene le informazioni della camera con dati riguardanti i numeri di posti disponibili e sul suo stato (occupata o no)
<b>DatiProdottiBar ()</b>	Operazione che mostra i dettagli dei singoli prodotti del bar con la disponibilità e il prezzo
<b>DatiAttivitàExtra ()</b>	Operazione che mostra le varie attività extra dell'albergo come per esempio gite escursioni ecc.
<b>DatiMenù ()</b>	Operazione che serve per mostrare un menu formato dalla data, dal tipo (cena o pranzo).
<b>DatiPortateMenù ()</b>	Operazione collegata alla classe menù, serve per mostrare le singole portate che formano il menù.
<b>DatiTipoPortate ()</b>	Operazione collegata alla classe menù, serve per mostrare le tipologie della portata.
<b>DatiCategorie ()</b>	Operazione che serve per elencare le varie categorie che formano il bar (per esempio: bibite, liquori, ecc.)
<b>DatiPrenotazione ()</b>	Operazione che consente ad un cliente di effettuare le prenotazioni di: attività extra, bar, ristorante
<b>DatiDettagliPrenotazioneAttivitàExtra ()</b>	Operazione che serve per mostrare i dettagli di una singola attività includendo dati come la disponibilità dei posti

<b>DatiDettaglioPrenotazioneBar ()</b>	Operazione che mostra e gestisce le varie prenotazioni che si possono effettuare al bar
<b>DatiDettaglioPrenotazioneRistorante ()</b>	Operazione che gestisce i dettagli riguardanti le prenotazioni del ristorante

## 5.Glossario

**Client:** componente che accede ai servizi o alle risorse di un'altra componente, detta server.

**Deployment Diagram:** Schema che descrive la struttura dinamica del sistema

**DBMS:** programma informatico (o, più frequentemente, un insieme di programmi) progettato per gestire un database, ovvero un insieme di numerosi dati strutturati. Le operazioni, normalmente, sono richieste da un gran numero di utenti.

**Event-driven:** modello in cui la segnalazione o notifica di un evento ha lo scopo di informare le applicazioni di uno o più domini destinatari che, sulla base del significato del messaggio associato all'evento, producono un cambiamento permanente del valore dei propri oggetti applicativi.

**Form:** finestra di dialogo incorporata in una pagina Web che consente all'utente di inserire informazioni destinate ad un server. Generalmente richiede un programma sul server che si occupi di esaminare le informazioni inviate. È composto da spazi (campi) predefiniti, ad esempio menù a tendina, elenchi puntati o caselle di testo libero.

**Internet:** Un sistema internazionale che connette tra loro piccole e grandi reti telematiche. Le reti connesse tramite Internet usano un particolare standard di comunicazione, conosciuto come TCP/IP.

**Java:** Linguaggio di programmazione ad oggetti, sviluppato da Sun Microsystem, che permette di generare sia programmi completi che moduli interattivi più piccoli (Applets), leggibili da un browser

**JDBC:** API per il linguaggio di programmazione Java che serve ai client per connettersi a un database. Fornisce metodi per interrogare e modificare i dati. È orientata ai database relazionali.

**Login:** Procedura attraverso la quale ci si collega con un qualsiasi servizio in linea. All'utente viene assegnato un nome di login ed una password che vengono richiesti dal sistema ogni volta che ci si collega.

**Layer:** E' un insieme di classi con funzionalità simile (tipicamente raggruppati in un unico package).

**Logout:** Operazione attraverso la quale si termina un collegamento con un sistema al quale si ha accesso attraverso un nome utente e una password (vedi *login*).

**Mysql:** Database management system relazionale, composto da un client con interfaccia a caratteri e un server, disponibile su molte piattaforme.

**Password:** E' un metodo di sicurezza che, mediante una stringa di caratteri, permette di identificare un utente specifico. Generalmente le password sono formate da una sequenza di lettere e numeri; digitando correttamente questi caratteri, si può avere accesso al computer o alla rete.

**Package:** Un Package rappresenta una collezione di classi ed interfacce che possono essere raggruppate in base alla funzione comune da esse svolta

**Server:** Programma di gestione di un servizio che invia informazioni in un particolare formato ricevuto e interpretato da un programma Client dal lato ricevente.

**Shutdown:** fase di terminazione del sistema.

**Start-up:** processo di accensione e di avvio di un computer, di un dispositivo o di un sistema.



**Università degli Studi di Salerno**  
Corso di Ingegneria del Software

---

# **Object Design Document**

## **Quick Service Project**

Versione 1.0



**Data 19/05/2008**

**Prefazione:**

Questo documento contiene i requisiti del Quick Service. E' destinato ai designers e ai clienti del progetto.

**Target:**

Clienti, Sviluppatori

**MEMBRI DEL QUICK SERVICE:****Team Members:**

*Vicidomini Vincenzo*

*Mercurio Antonio*

*Iacoletti Alessandro*

*Pacifico Marta*

*Pizza Luca Ernesto*

## **INDICE DEI CONTENUTI**

<b>1. INTRODUZIONE.....</b>	.....
1.1	OBJECT DESIGN TRADE-OFF.....
1.2	LINEE GUIDA PER LA DOCUMENTAZIONE DELLE INTERFACCIE.....
1.3	DEFINIZIONI, ACRONIMI ED ABBREVIAZIONI.....
<b>2. PACKAGE</b>	
<b>3. INTERFACCIA DELLE CLASSI</b>	
<b>4. GLOSSARIO</b>	

# **1. Introduzione**

Il documento descrive gli obiettivi del design imposti dagli sviluppatori e le linee guida per lo sviluppo delle interfacce dei sottosistemi e la suddivisione dei sottosistemi in package e classi.

## **1.1. Object Design trade – off**

Questo documento si propone di evidenziare gli obiettivi principali dello sviluppo dal punto di vista degli sviluppatori:

- L'applicazione non deve richiedere molte risorse al sistema in termini di memoria e tempo CPU, in questo modo è possibile usare anche macchine non recenti
- Il tempo di risposta non è fondamentale, ma non deve esserci una risposta troppo lenta del sistema, in particolare la parte che interagisce con l'utente.
- Il sistema deve sfruttare al meglio la rete senza sovraccaricarla inutilmente
  - Lo spazio occupato dai file dell'applicazione sul disco non è fondamentale visto il rapporto costo/spazio dei supporti

## **1.2. Linee guida per la documentazione delle Interfacce**

Nel redigere il codice del programma verranno usate le seguenti convenzioni oltre alle convenzioni che sono normalmente adottate nella scrittura del codice Java:

- I nomi delle Classi devono essere al singolare ed avere l'iniziale maiuscola, in caso sia il nome di un frame preceduti dal prefisso “frm\_”.
- I nomi dei pannelli sono preceduti dal prefisso “pnl”
- I nomi dei pulsanti sono preceduti dal prefisso “btn”
- I nomi delle label sono preceduti dal prefisso “lbl”
- I nomi delle tabelle sono preceduti dal prefisso “tbl”
- Le classi che implementano le interfacce hanno il suffisso “Imp”
- I metodi hanno per nome un verbo che indica l'azione che compie il metodo

## **1.3. Definizioni, acronimi e abbreviazioni**

Di seguito sono riportati alcune abbreviazioni usate dagli sviluppatori durante la scrittura del codice:

- obj = object
- uid = User ID

- pwd = password
- hh = ora
- mm = minuti
- ss = secondi

## 2. Package

In questa sezione sono riportati tutti i package che costituiscono l'applicazione.

### 2.1. Package QsServer

In questo Package ci sono le classi e le interfacce relative all'interfaccia e alla classe della sessione semplice e altre classi necessarie al funzionamento del server, le classi delle altre sessione si trovano dei sottopacchetti QsServer.DataLayer, QsServer.AddettoBar, QsServer.Cliente.

### Interface Summary

<u>Sessione</u>	Titolo: Sessione Descrizione: Interfaccia per la sessione di default
-----------------	--

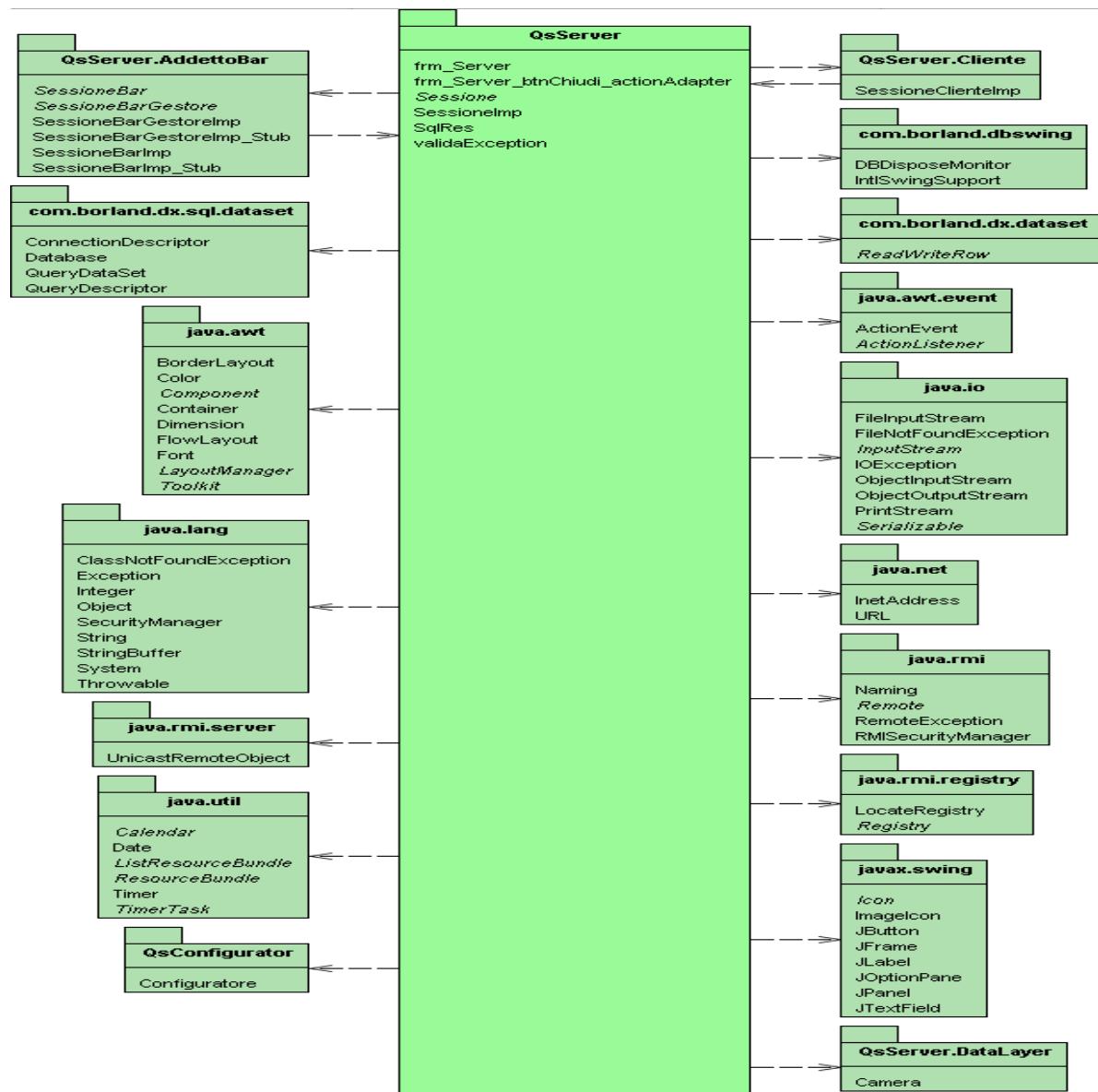
### Class Summary

<u>frm_Log</u>	Titolo: frm_Log Descrizione: La classe permette di gestire i log registrati dal sistema
<u>frm_Log_btnChiudi_actionAdapter</u>	Titolo: frm_Log_btnChiudi_actionAdapter Descrizione: L'Adapter associa il metodo actionPerformed al pulsante btnChiudi
<u>frm_Log_btnClear_actionAdapter</u>	Titolo: frm_Log_btnClear_actionAdapter Descrizione: Permette di associare l'actionPerformed al pulsante btnClear

<b><u>frm_Log_btnRefresh_actionAdapter</u></b>	Titolo: frm_Log_btnRefresh_actionAdapter Descrizione: L'Adapter associa il metodo actionPerformed al pulsante btnRefresh
<b><u>frm_Server</u></b>	Titolo: frm_Server Descrizione: Mostra il frame principale del server
<b><u>frm_Server_btnChiudi_actionAdapter</u></b>	Titolo: frm_Server_btnChiudi_actionAdapter Descrizione: Permette di associare l'actionPerformed al pulsante btnChiudi
<b><u>frm_Server_btnLog_actionAdapter</u></b>	Titolo: frm_Server_btnLog_actionAdapter Descrizione: Permette di associare l'actionPerformed al pulsante btnLog
<b><u>SessioneImp</u></b>	Titolo: SessioneImp Descrizione: Progetto QuickService
<b><u>SqlRes</u></b>	Titolo: SqlRes Descrizione: Raggruppa tutte le query in una matrice

## Exception Summary

<b><u>validaException</u></b>	Titolo: validaException Descrizione: Eccezione per la funziona valida
-------------------------------	--



## 2.2. Package QsServer.AddettoBar

Contiene le classi e le interfaccie relative alle sessioni dell'addettoBar.

### Interface Summary

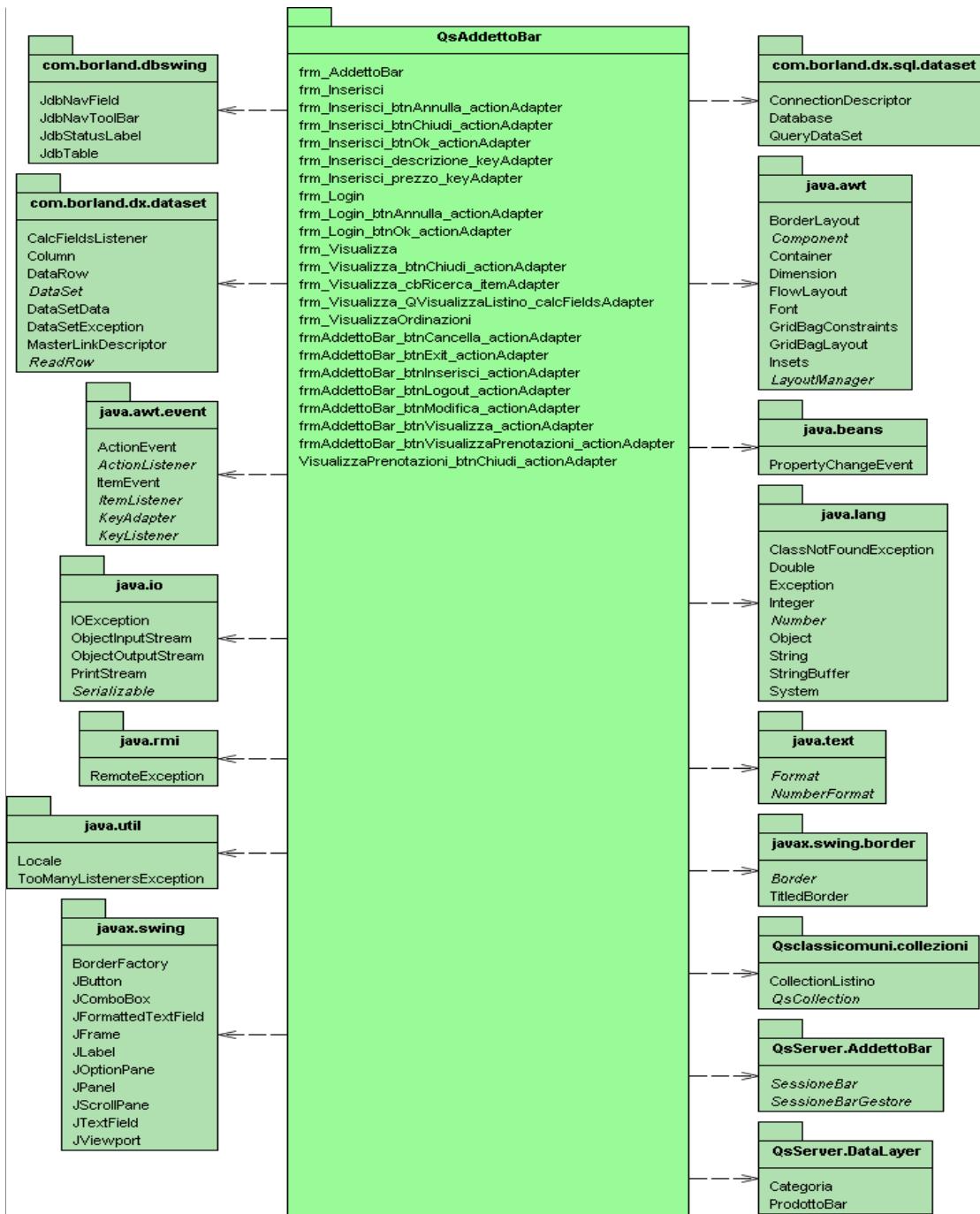
<u>SessioneBar</u>	<p><b>Titolo:</b> SessioneBar</p> <p><b>Descrizione:</b> Interfaccia per la sessione Bar, fornisce i metodi basilari per il bar</p>
<u>SessioneBarGestore</u>	<p><b>Titolo:</b> SessioneBarGestore</p> <p><b>Descrizione:</b> Interfaccia per la sessione Bar, fornisce i metodi basilari per la gestione del bar</p>

## Class Summary

<u>SessioneBarGestoreImp</u>	<p>Titolo: SessioneBarGestoreImp Descrizione: Implementa i metodi dell'interfaccia SessioneBarGestore</p>
<u>SessioneBarImp</u>	<p>Titolo: SessioneBarImp Descrizione: Implementa i metodi dell'interfaccia SessioneBar</p>

## Exception Summary

<u>ProdottoBarException</u>	<p>Titolo: ProdottoBarException Descrizione: Eccezione che viene lanciata in caso di errori nella gestione dei prodotti bar</p>
-----------------------------	---



### 2.3. Package QsServer.Cliente

Contiene le classi e le interfacce relative alla sessione del Cliente

## Interface Summary

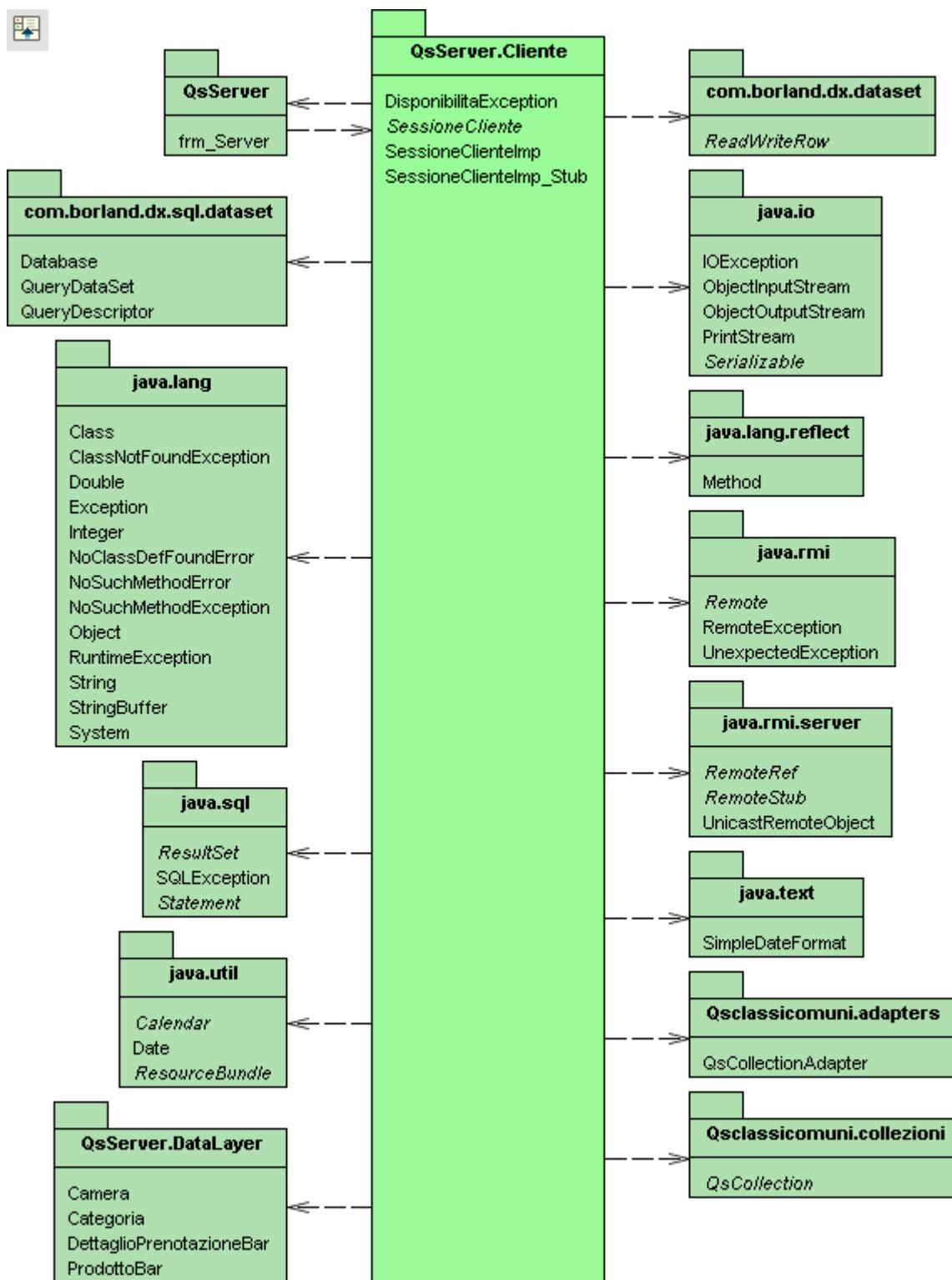
<b><u>SessioneCliente</u></b>	<p>Titolo: SessioneCliente Descrizione: Interfaccia che fornisce i metodi necessari al client per effettuare/modificare/cancellare le ordinazioni</p>
-------------------------------	---

## Class Summary

<b><u>SessioneClienteImp</u></b>	<p>Titolo: SessioneClienteImp Descrizione: Progetto QuickService</p>
----------------------------------	--

## Exception Summary

<b><u>DisponibilitaException</u></b>	<p>Titolo: DisponibilitaException Descrizione: Eccezione che viene lanciata in caso la disponibilità non è sufficiente per soddisfare la richiesta di un determinato prodotto</p>
--------------------------------------	---

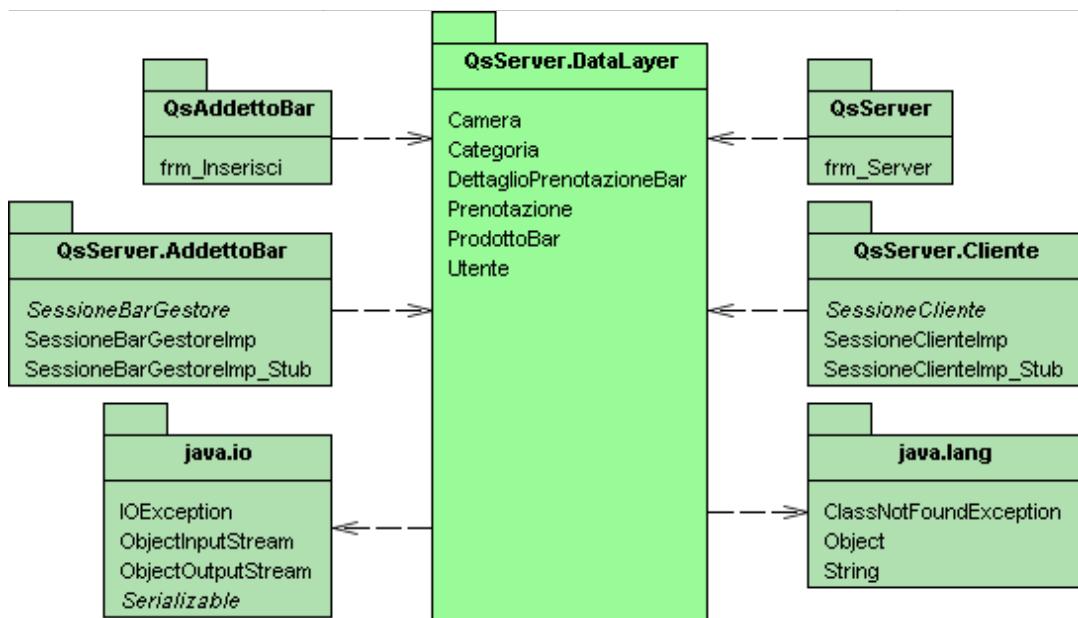


## 2.4. Package QsServer.DataLayer

Contiene le classi del data layer che vengono usate per rappresentare le entità di cui tener traccia all'interno del sistema.

## Class Summary

<u>Camera</u>	Titolo: Camera Descrizione: Describe una camera della struttura
<u>Categoria</u>	Titolo: Categoria Descrizione: Describe una categoria di prodotti
<u>DettaglioPrenotazioneBar</u>	Titolo: DettaglioPrenotazioneBar Descrizione: Describe elemento di una prenotazione
<u>Prenotazione</u>	Titolo: Prenotazione Descrizione: Describe una prenotazione
<u>ProdottoBar</u>	Titolo: ProdottoBar Descrizione: Describe un prodotto del bar
<u>Utente</u>	Titolo: Utente Descrizione: Describe l'utente che si logga al sistema



### 2.5. Package `QsServer.util`

Contiene le classi per la gestione dei log di sistema

## Class Summary

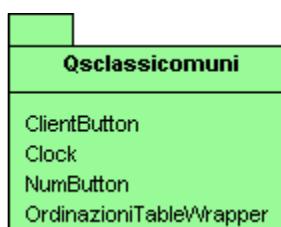
<u>Log</u>	Titolo: Log Descrizione: Descrive un evento del sistema
<u>LogModel</u>	Titolo: LogModel Descrizione: Modello di tabella, questo modello incorpora una array list come struttura dati interna, la classe client continua ad usare la tabella a prescindere dalla struttura dati utilizzata per la collezione

## 2.6. Package Qsclassicomuni

Contiene Classi che vengono usate sia dai client che dal server, le classi adapter e le collezioni sono nei sottopacchetti Qsclassicomuni.adapters, Qsclassicomuni.collezione.

## Class Summary

<u>ClientButton</u>	Titolo: ClientButton Descrizione: Descrive i pulsanti per i prodotti del listino cliente
<u>Clock</u>	Titolo: Clock Descrizione: Descrive un orologio
<u>NumButton</u>	Titolo: NumButton Descrizione: Descrive i pulsanti del tastierino numerico
<u>OrdinazioniTableWrapper</u>	Titolo: OrdinazioniTableWrapper Descrizione: Modello di tabella, questo modello incorpora una array list come struttura dati interna, la classe client continua ad usare la tabella a prescindere dalla struttura dati utilizzata per la collezione



## 2.7. Package Qsclassicomuni.adapters

Contiene gli adapter per le classi collection

## Class Summary

<b><u>QsCollectionAdapter</u></b>	<p>Titolo: QsCollectionAdapter</p> <p>Descrizione: Questa Classe invece ha il ruolo di adapter cioè fa da tramite tra la classe target e la classe Adaptee, ovvero la classe che fornisce l'implementazione al target, che nel nostro caso è la classe ArrayList.</p>
-----------------------------------	---

### 2.8. Package Qsclassicomuni.collezioni

Contiene le classi collection per recuperare più oggetti dello stesso tipo dal database.

## Interface Summary

<b><u>QsCollection</u></b>	<p>Titolo: QsCollection</p> <p>Descrizione: Questa classe rappresenta la classe Target a cui il client accede.</p>
----------------------------	--

## Class Summary

<b><u>CollectionListino</u></b>	<p>Titolo: CollectionListino</p> <p>Descrizione: Collezione di oggetti di tipo prodottoBar che compongono il listino</p>
---------------------------------	--

### 2.9. Package QsAddettoBar

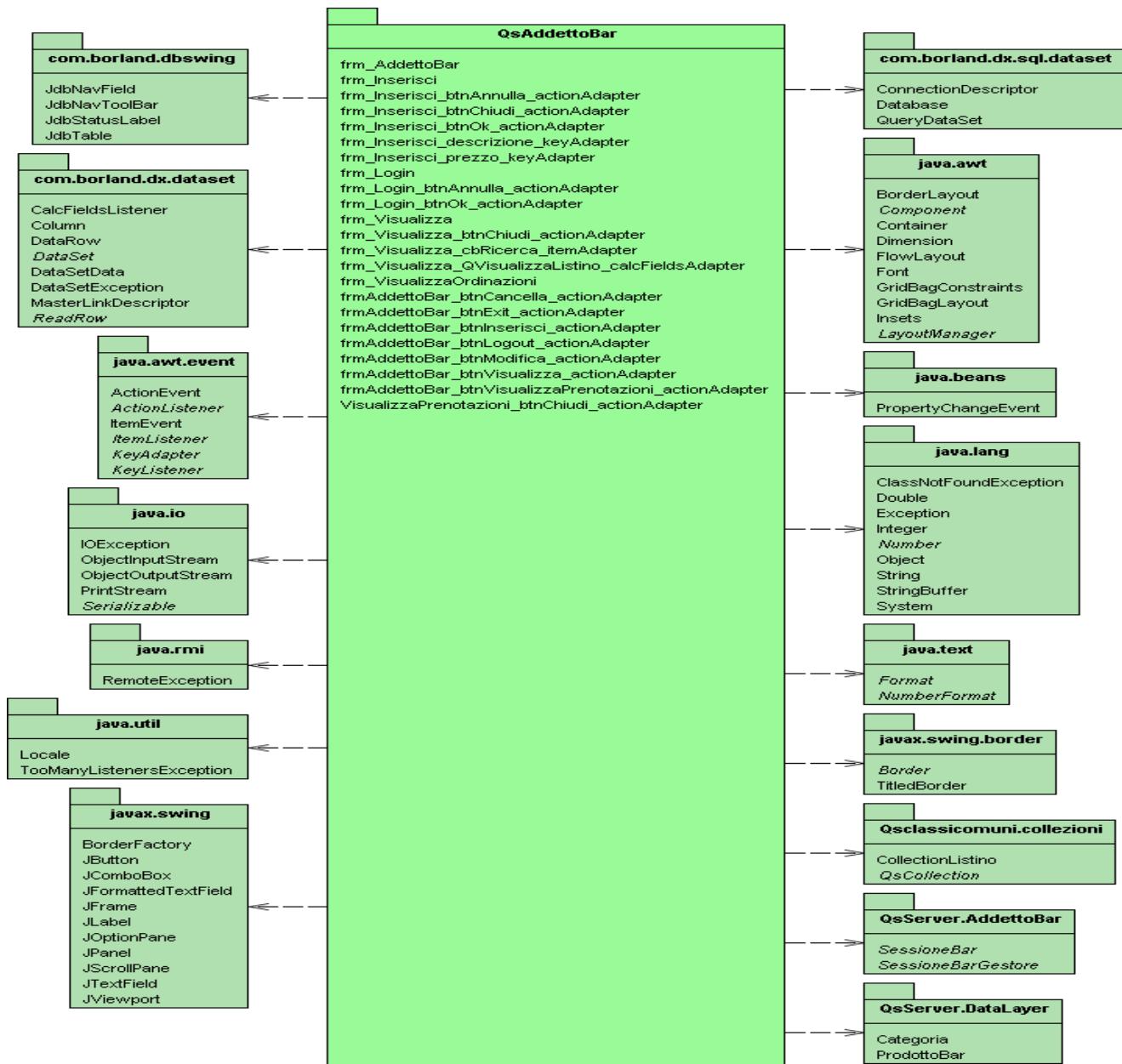
Contiene le classi per l'interfaccia grafica del client dell'addetto bar.

## Class Summary

<b><u>frm AddettoBar</u></b>	<p>Titolo: frm_AddettoBar</p> <p>Descrizione: La classe rappresenta il frame principale dove vengono istanziati i pulsanti che permettono l'interazione con l'utente.</p>
<b><u>frm Inserisci</u></b>	<p>Titolo: frm_Inserisci</p> <p>Descrizione: Rappresenta il frame che permette l'inserimento di un nuovo prodotto nell'archivio</p>

<b><u>frm Inserisci btnAnnulla actionAdapter</u></b>	Titolo: frm_Inserisci_btnAnnulla_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerformed del pulsante btnAnnulla
<b><u>frm Inserisci btnChiudi actionAdapter</u></b>	Titolo: frm_Inserisci_btnChiudi_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerformed del pulsante btnChiudi
<b><u>frm Inserisci btnOk actionAdapter</u></b>	Titolo: frm_Inserisci_btnOk_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerformed del pulsante btnOk
<b><u>frm Inserisci cbxillimitato itemAdapter</u></b>	Titolo: frm_Inserisci_cbxillimitato_itemAdapter Descrizione: Associa gli eventi legati al cambiamento di stato del check box cbxillimitato
<b><u>frm Inserisci descrizione keyAdapter</u></b>	Titolo: frm_Inserisci_desrizione_keyAdapter Descrizione: Associa gli eventi legati all'input da tastiera per l'inserimento del campo descrizione
<b><u>frm Inserisci prezzo keyAdapter</u></b>	Titolo: frm_Inserisci_prezzo_keyAdapter Descrizione: Associa gli eventi legati all'input da tastiera per l'inserimento del campo prezzo
<b><u>frm Login</u></b>	Titolo: frm_Login Descrizione: Il frame permette all'utente di inserire nome e password e loggarsi all'interno del sistema)
<b><u>frm Login btnAnnulla actionAdapter</u></b>	Titolo: frm_Login_btnAnnulla_actionAdapter Descrizione: Progetto La classe funge da Adapter per poter gestire l'actionPerformed del pulsante btnAnnulla
<b><u>frm Login btnOk actionAdapter</u></b>	Titolo: frm_Login_btnOk_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerformed del pulsante btnOk
<b><u>frm Visualizza</u></b>	Titolo: frm_Visualizza Descrizione: Visualizza in una tabella il listino
<b><u>frm Visualizza btnChiudi actionAdapter</u></b>	Titolo: frm_Visualizza_btnChiudi_actionAdapter Descrizione: permette di associare il metodo btnChiudiActionPerformed(e) al pulsante btnChiudi
<b><u>frm Visualizza cbRicerca itemAdapter</u></b>	Titolo: frm_Visualizza_cbRicerca_itemAdapter Descrizione: Permette di associare l'evento cbRicerca_itemStateChanged(e) all'oggetto

<b><u>frm_Visualizza_QVisualizzaListino_calcFieldsAdapter</u></b>	Titolo: frm_Visualizza_QVisualizzaListino_calcFieldsAdapter Descrizione: classe Adapter per i campi calcolati
<b><u>frm_VisualizzaOrdinazioni</u></b>	Titolo: frm_VisualizzaOrdinazioni Descrizione: visualizza le ordinazioni effettuate dai clienti che sono state
<b><u>frmAddettoBar_btnCancella_actionAdapter</u></b>	Titolo: frmAddettoBar_btnCancella_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btncancella
<b><u>frmAddettoBar_btnExit_actionAdapter</u></b>	Titolo: frmAddettoBar_btnExit_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btnExit
<b><u>frmAddettoBar_btnInserisci_actionAdapter</u></b>	Titolo: frmAddettoBar_btnInserisci_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btnInserisci
<b><u>frmAddettoBar_btnLogout_actionAdapter</u></b>	Titolo: frmAddettoBar_btnLogout_actionAdapter Descrizione: a classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btnLogout
<b><u>frmAddettoBar_btnModifica_actionAdapter</u></b>	Titolo: frmAddettoBar_btnModifica_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btnModifica
<b><u>frmAddettoBar_btnVisualizza_actionAdapter</u></b>	Titolo: frmAddettoBar_btnVisualizza_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btnVisualizza
<b><u>frmAddettoBar_btnVisualizzaPrenotazioni_actionAdapter</u></b>	Titolo: frmAddettoBar_btnVisualizzaPrenotazioni_actionAdapter Descrizione: La classe funge da Adapter per poter gestire l'actionPerfomed del pulsante btnVisualizzaPrenotazioni
<b><u>VisualizzaPrenotazioni_bt nChiudi_actionAdapter</u></b>	Titolo: VisualizzaPrenotazioni_btnChiudi_actionAdapter Descrizione: Funge da adapter per associare il metodo btnChiudi_actionPerformed(e) al pulsante btnChiudi



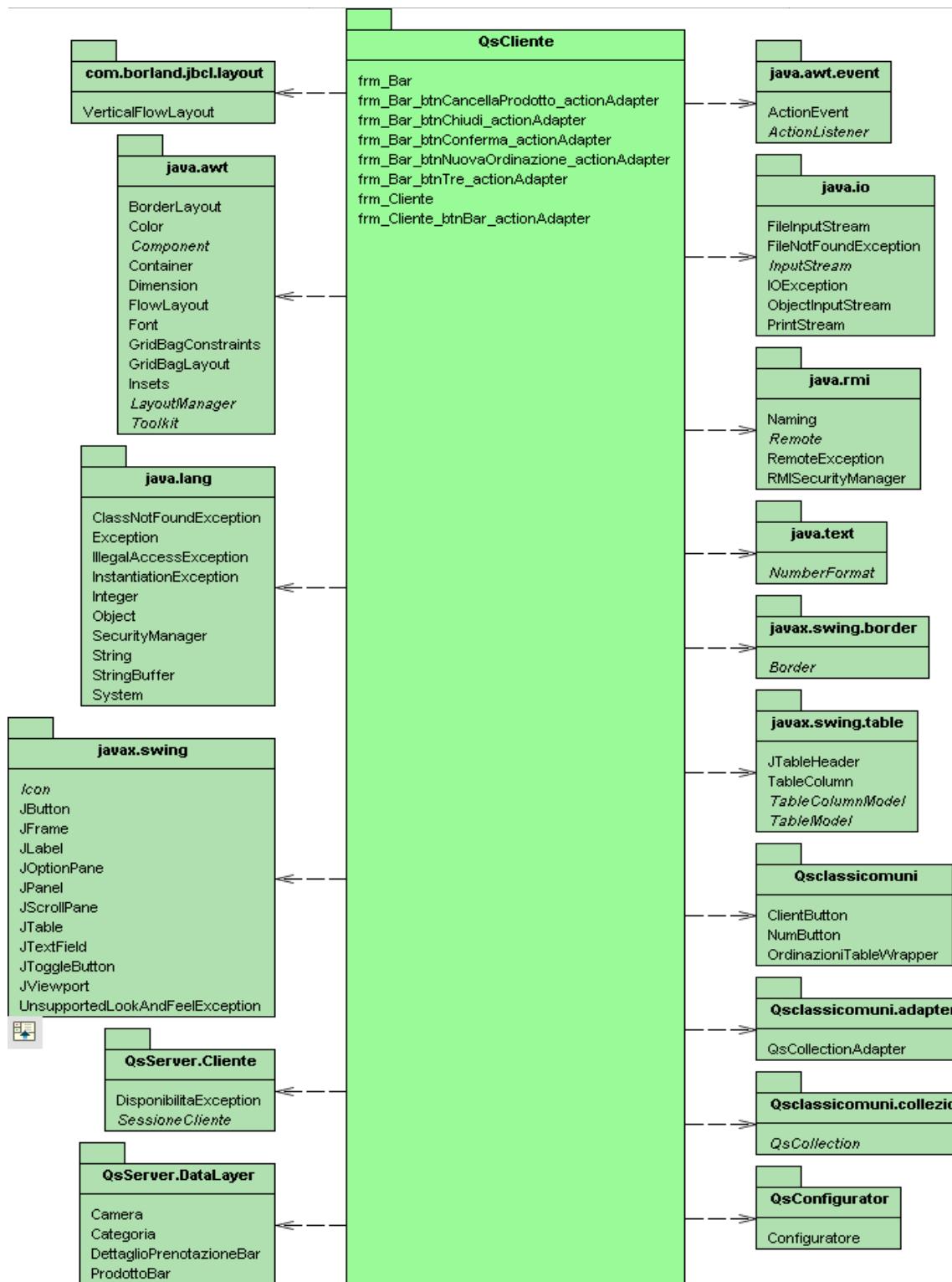
## 2.10. Package QsCliente

Contiene le classi usate dai client per l’interfaccia grafica del client per i clienti

### Class Summary

<b><u>frm Bar</u></b>	<p>Titolo: frm_bar</p> <p>Descrizione: La classe permette di navigare il listino ed effettuare delle prenotazioni</p>
-----------------------	---

<b><u>frm Bar btnCancellaProdotto actionAdapter</u></b>	Titolo: frm_Bar_btnCancellaProdotto_actionAdapter Descrizione: L'Adapter permette di gestire l'actionPerformed del pulsante btnCancellaProdotto
<b><u>frm Bar btnChiudi actionAdapter</u></b>	Titolo: frm_Bar_btnChiudi_actionAdapter Descrizione: L'Adapter permette di gestire l'actionPerformed del pulsante btnChiudi
<b><u>frm Bar btnConferma actionAdapter</u></b>	Titolo: frm_Bar_btnConferma_actionAdapter Descrizione: L'Adapter permette di gestire l'actionPerformed del pulsante btnConferma
<b><u>frm Bar btnNuovaOrdinazione actionAdapter</u></b>	Titolo: frm_Bar_btnNuovaOrdinazione_actionAdapter Descrizione: L'Adapter permette di gestire l'actionPerformed del pulsante btnNuovaOrdinazione
<b><u>frm Bar btnOpzioni actionAdapter</u></b>	Titolo: frm_Bar_btnOpzioni_actionAdapter Descrizione: L'adapter permette di gestire l'actionPerformed del pulsante btnOpzioni
<b><u>frm Cliente</u></b>	Titolo: frm_Cliente Descrizione: Mostra il frame principale del cliente
<b><u>frm Cliente btnBar actionAdapter</u></b>	Titolo: frm_Cliente_btnBar_actionAdapter Descrizione: Permette di associare l'actionPerformed al pulsante
<b><u>frm Note</u></b>	Titolo: frm_Note Descrizione: Rappresenta il frame per l'inserimento delle opzioni sul prodotto selezionato
<b><u>frm Note btnAnnulla actionAdapter</u></b>	Titolo: frm_Note_btnAnnulla_actionAdapter Descrizione: l'adapter permette di associare l'actionPerformed al pulsante btnAnnulla
<b><u>frm Note btnInserisci actionAdapter</u></b>	Titolo: frm_Note_btnInserisci_actionAdapter Descrizione: l'adapter permette di associare l'actionPerformed al pulsante btnInserisci



### 3. Class Interface

Verranno presentate solo le classi principali del progetto per le altre riferirsi alla documentazione javadoc.

**QsServer**

#### Class SessioneImp

```
java.lang.Object
  java.rmi.server.RemoteObject
    java.rmi.server.RemoteServer
      java.rmi.server.UnicastRemoteObject
        QsServer.SessioneImp
```

##### All Implemented Interfaces:

java.rmi.Remote, java.io.Serializable, [Sessione](#)

##### Direct Known Subclasses:

[SessioneBarImp](#)

---

```
public class SessioneImp
extends java.rmi.server.UnicastRemoteObject
implements Sessione
```

Title: SessioneImp

Description: Implementazione della sessione di default

##### Version:

1.0

##### Author:

Mercurio Antonio, Vicedomini Vincenzo

---

## Field Summary

<b>Fields inherited from class java.rmi.server.RemoteObject</b>
---

Ref
-----

---

## Constructor Summary

<b>SessioneImp( )</b>	
-----------------------	--

Costruttore di default	
------------------------	--

## Method Summary

boolean	<b>controllaOra</b> (int[] d, int ahh, int amm, int ass, int mills) controlla l'ora del server con l'ora relativa alla prenotazione
java.util.Date	<b>getData()</b> Restituisce la data del server

### Methods inherited from class java.rmi.server.UnicastRemoteObject

clone, exportObject, exportObject, exportObject, unexportObject

### Methods inherited from class java.rmi.server.RemoteServer

getClientHost, getLog, setLog

### Methods inherited from class java.rmi.server.RemoteObject

equals, getRef, hashCode, toString, toStub

### Methods inherited from class java.lang.Object

finalize, getClass, notify, notifyAll, wait, wait, wait

## Constructor Detail

### SessioneImp

```
public SessioneImp()  
    throws java.rmi.RemoteException
```

Costruttore di default

#### Throws:

java.rmi.RemoteException - lancia l'eccezione per problemi con RMI

## Method Detail

### getData

```
public java.util.Date getData()
```

Restituisce la data del server

**Specified by:**

getData in interface Sessione

**Returns:**

Date data del server

---

## controllaOra

```
public boolean controllaOra(int[] d,  
                           int ahh,  
                           int amm,  
                           int ass,  
                           int mills)
```

controlla l'ora del server con l'ora relativa alla prenotazione

**Specified by:**

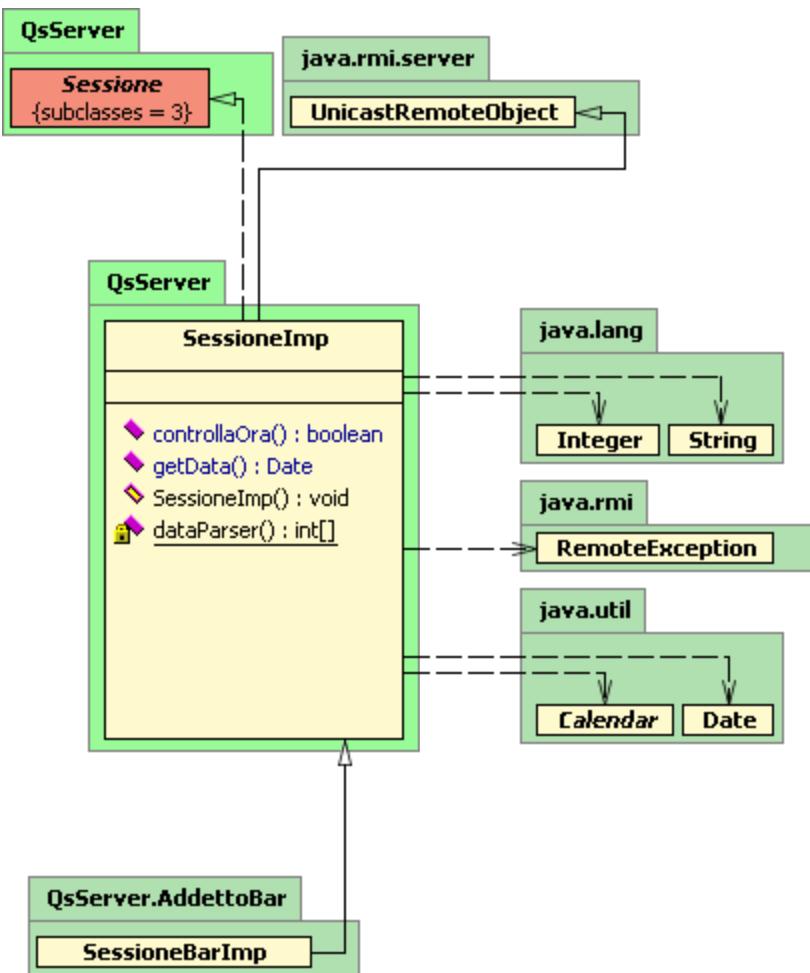
controllaOra in interface Sessione

**Parameters:**

d - int[] array che contiene la data della prenotazione  
ahh - int ora della prenotazione  
amm - int minuti della prenotazione  
ass - int secondi della prenotazione  
mills - int millisecondi tempo che deve trascorrere per confermare la prenotazione

**Returns:**

boolean ritorna true se il tempo trascorso dal momento della prenotazione è maggiore di mills false altrimenti



## QsServer.AddettoBar Class SessioneBarImp

```

java.lang.Object
  java.rmi.server.RemoteObject
    java.rmi.server.RemoteServer
      java.rmi.server.UnicastRemoteObject
        QsServer.SessioneImp
          QsServer.AddettoBar.SessioneBarImp

```

### All Implemented Interfaces:

`java.rmi.Remote, java.io.Serializable, Sessione, SessioneBar`

```

public class SessioneBarImp
extends SessioneImp
implements SessioneBar

```

Title: SessioneBarImp

Description: Implementa i metodi dell'interfaccia SessioneBar

### Version:

1.0

### Author:

Alessandro Iacoletti, Pacifico Marta

## Field Summary

(package private) com.borland.dx.sql.dataset.Database	<b><u>database</u></b>
(package private) com.borland.dx.sql.dataset.QueryDataSet	<b><u>qds</u></b>
(package private) java.util.ResourceBundle	<b><u>sqlRes</u></b>

## Fields inherited from class java.rmi.server.RemoteObject

Ref

## Constructor Summary

**SessioneBarImp**(com.borland.dx.sql.dataset.Database database,  
java.util.ResourceBundle sqlRes)  
Costruttore pubblico Costruttore pubblico, imposta l'oggetto remoto

## Method Summary

com.borland.dx.dataset.DataSetData	<b><u>visualizzaListino</u></b> ( ) Restituisce tutti i prodotti del bar
com.borland.dx.dataset.DataSetData []	<b><u>visualizzaOrdinazione</u></b> ( ) visualizza le ordinazioni che sono state confermate

## Methods inherited from class QsServer.SessioneImp

controllaOra, getData

## Methods inherited from class java.rmi.server.UnicastRemoteObject

clone, exportObject, exportObject, exportObject, unexportObject

## Methods inherited from class java.rmi.server.RemoteServer

getClientHost, getLog, setLog

## Methods inherited from class java.rmi.server.RemoteObject

```
equals, getRef, hashCode, toString, toStub
```

#### Methods inherited from class `java.lang.Object`

```
finalize, getClass, notify, notifyAll, wait, wait, wait
```

#### Methods inherited from interface `QsServer.Sessione`

```
controllaOra, getData
```

## Field Detail

### database

```
com.borland.dx.sql.dataset.Database database
```

---

### sqlRes

```
java.util.ResourceBundle sqlRes
```

---

### qds

```
com.borland.dx.sql.dataset.QueryDataSet qds
```

## Constructor Detail

### SessioneBarImp

```
public SessioneBarImp(com.borland.dx.sql.dataset.Database database,
                      java.util.ResourceBundle sqlRes)
                    throws java.rmi.RemoteException
```

Costruttore pubblico Costruttore pubblico, imposta l'oggetto remoto

#### Parameters:

database - Database connessione al database

sqlRes - ResourceBundle matrice con le query SQL

#### Throws:

`java.rmi.RemoteException` - lancia l'eccezione in caso di errore RMI

## Method Detail

### visualizzaListino

```
public com.borland.dx.dataset.DataSetData visualizzaListino()
```

```
throws  
java.rmi.RemoteException,  
com.borland.dx.dataset.DataSetException
```

Restituisce tutti i prodotti del bar

**Specified by:**

[visualizzaListino](#) in interface [SessioneBar](#)

**Returns:**

DataSetData contiente tutti i prodotti del listino bar

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI  
com.borland.dx.dataset.DataSetException - l'eccezione se la query non ha successo

---

## **visualizzaOrdinazione**

```
public com.borland.dx.dataset.DataSetData[] visualizzaOrdinazione()  
throws  
java.rmi.RemoteException,  
com.borland.dx.dataset.DataSetException
```

visualizza le ordinazioni che sono state confermate

**Specified by:**

[visualizzaOrdinazione](#) in interface [SessioneBar](#)

**Returns:**

DataSetData[] contiente le informazioni di tutte le ordinazioni confermate

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI  
com.borland.dx.dataset.DataSetException - lancia l'eccezione se la query non ha successo

**QsServer.AddettoBar**

## **Interface SessioneBarGestore**

**All Superinterfaces:**

java.rmi.Remote

**All Known Implementing Classes:**

[SessioneBarGestoreImp](#)

---

```
public interface SessioneBarGestore  
extends java.rmi.Remote
```

Title: SessioneBarGestore

Description: Interfaccia per la sessione Bar, fornisce i metodi basilari per la gestione del bar

**Version:**

1.0

**Author:**

Pizza Luca Ernesto, Vicedomini Vincenzo

## Method Summary

Qsclassicomuni.collezioni.QsCollection	<b><u>caricaCategorie( )</u></b> Restituisce una collezione con le categorie presenti nel database;
void	<b><u>controllaProdotto( ProdottoBar p )</u></b> Controlla se i dati riguardanti il prodotto sono corretti
Utente	<b><u>getDatiUtente( )</u></b> Restituisce i dati dell'utente loggato
void	<b><u>inserisciProdotto( ProdottoBar p )</u></b> Inserisce il prodotto contenuto in p
boolean	<b><u>valida( java.lang.String UID, java.lang.String PWD )</u></b> Effettua il login dell'utente

## Method Detail

**valida**

```
public boolean valida(java.lang.String UID,
                     java.lang.String PWD)
    throws java.rmi.RemoteException,
           validaException
```

Effettua il login dell'utente

**Parameters:**

UID - String (User id Utente)  
 PWD - String (Pasword Utente)

**Returns:**

boolean Ritorna True se la User Id e la Password Sono corrette

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI  
validaException

**caricaCategorie**

```
public Qsclassicocomuni.collezioni.QsCollection caricaCategorie()
throws
java.rmi.RemoteException
```

Restituisce una collezione con le categorie presenti nel database;

**Returns:**

DataSetData contiente tutte le categorie dei prodotti del bar

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI

---

## **controllaProdotto**

```
public void controllaProdotto(ProdottoBar p)
throws java.rmi.RemoteException,
ProdottoBarException
```

Controlla se i dati riguardanti il prodotto sono corretti

**Parameters:**

p - ProdottoBar è il prodotto da controllare

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI

ProdottoBarException - lancia l'eccezione se uno o più dati all'interno della classe sono errati

---

## **inserisciProdotto**

```
public void inserisciProdotto(ProdottoBar p)
throws java.rmi.RemoteException
```

Inserisce il prodotto contenuto in p

**Parameters:**

p - Prodotto riferimento ad un oggetto che contiene i dati relativi ad un prodotto

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI

---

## **getDatiUtente**

```
public Utente getDatiUtente()
throws java.rmi.RemoteException
```

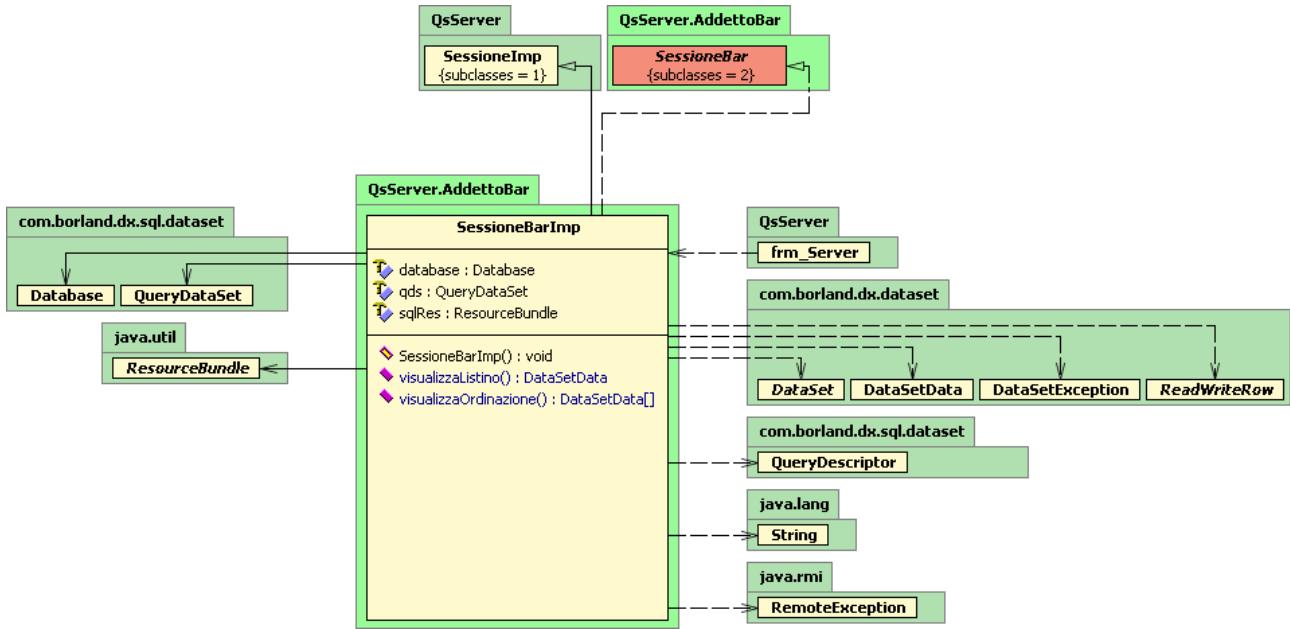
Restituisce i dati dell'utente loggato

**Returns:**

Utente istanza con i dati dell'utente loggato

**Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI



### **QsServer.Cliente**

## **Class SessioneClienteImp**

```

java.lang.Object
java.rmi.server.RemoteObject
java.rmi.server.RemoteServer

```

```

java.rmi.server.UnicastRemoteObject
QsServer.Cliente.SessioneClienteImp

```

### **All Implemented Interfaces:**

`java.rmi.Remote, java.io.Serializable, SessioneCliente`

```

public class SessioneClienteImp
extends java.rmi.server.UnicastRemoteObject
implements SessioneCliente

```

Title: SessioneClienteImp

Description: Progetto QuickService (Gruppo De Sio)

**Version:**

1.0

**Author:**

Alessandro Iacoletti, Pacifico Marta

## **Field Summary**

### **Fields inherited from class `java.rmi.server.RemoteObject`**

ref

## Constructor Summary

**SessioneClienteImp**(com.borland.dx.sql.dataset.Database database,  
java.util.ResourceBundle sqlRes, Camera camera)  
Costruttore pubblico

## Method Summary

<u>Camera</u>	<b>getDatiCamera()</b> restituisce i dati della camera
void	<b>ordinazioneBar</b> (int room, Qsclassicomuni.collezioni.QsCollection collect, double tot)  invia l'ordinazione al server il metodo è sincronizzato in modo da evitare problemi di concorrenza nella chiamata contemporanea di questo metodo da parte dei client
Qsclassicomuni.collezioni.QsCollection	<b>visualizzaListino</b> (int codCat) Mostra i prodotti disponibili in base alla categoria scelta

### Methods inherited from class java.rmi.server.UnicastRemoteObject

clone, exportObject, exportObject, exportObject, unexportObject

### Methods inherited from class java.rmi.server.RemoteServer

getClientHost, getLog, setLog

### Methods inherited from class java.rmi.server.RemoteObject

equals, getRef, hashCode, toString, toStub

### Methods inherited from class java.lang.Object

finalize, getClass, notify, notifyAll, wait, wait, wait

## Constructor Detail

## **SessioneClienteImp**

```
public SessioneClienteImp(com.borland.dx.sql.dataset.Database database,
                          java.util.ResourceBundle sqlRes,
                          Camera camera)
                        throws java.rmi.RemoteException
```

Costruttore pubblico

### **Parameters:**

camera - Camera camera del cliente che apre la sessione

### **Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI

## **Method Detail**

### **ordinazioneBar**

```
public void ordinazioneBar(int room,
                           Qsclassicomuni.collezioni.QsCollection collect,
                           double tot)
                           throws DisponibilitaException
```

invia l'ordinazione al server il metodo è sincronizzato in modo da evitare problemi di concorrenza nella chiamata contemporanea di questo metodo da parte dei client

### **Specified by:**

[ordinazioneBar](#) in interface [SessioneCliente](#)

### **Parameters:**

room - int numero della camera

collect - QsCollection prodotti dell'ordinazione

tot - double totale dell'ordinazione

### **Throws:**

[DisponibilitaException](#) - lancia l'eccezione in caso sia impossibile inviare l'ordinazione

---

### **visualizzaListino**

```
public Qsclassicomuni.collezioni.QsCollection visualizzaListino(int codCat)
                     throws
java.rmi.RemoteException
```

Mostra i prodotti disponibili in base alla categoria scelta

### **Specified by:**

[visualizzaListino](#) in interface [SessioneCliente](#)

### **Parameters:**

codCat - int codice categoria

### **Returns:**

QsCollection ritorna un insieme di prodotti

### **Throws:**

java.rmi.RemoteException - lancia l'eccezione in caso di errore RMI

## **getDatiCamera**

```
public Camera getDatiCamera()
```

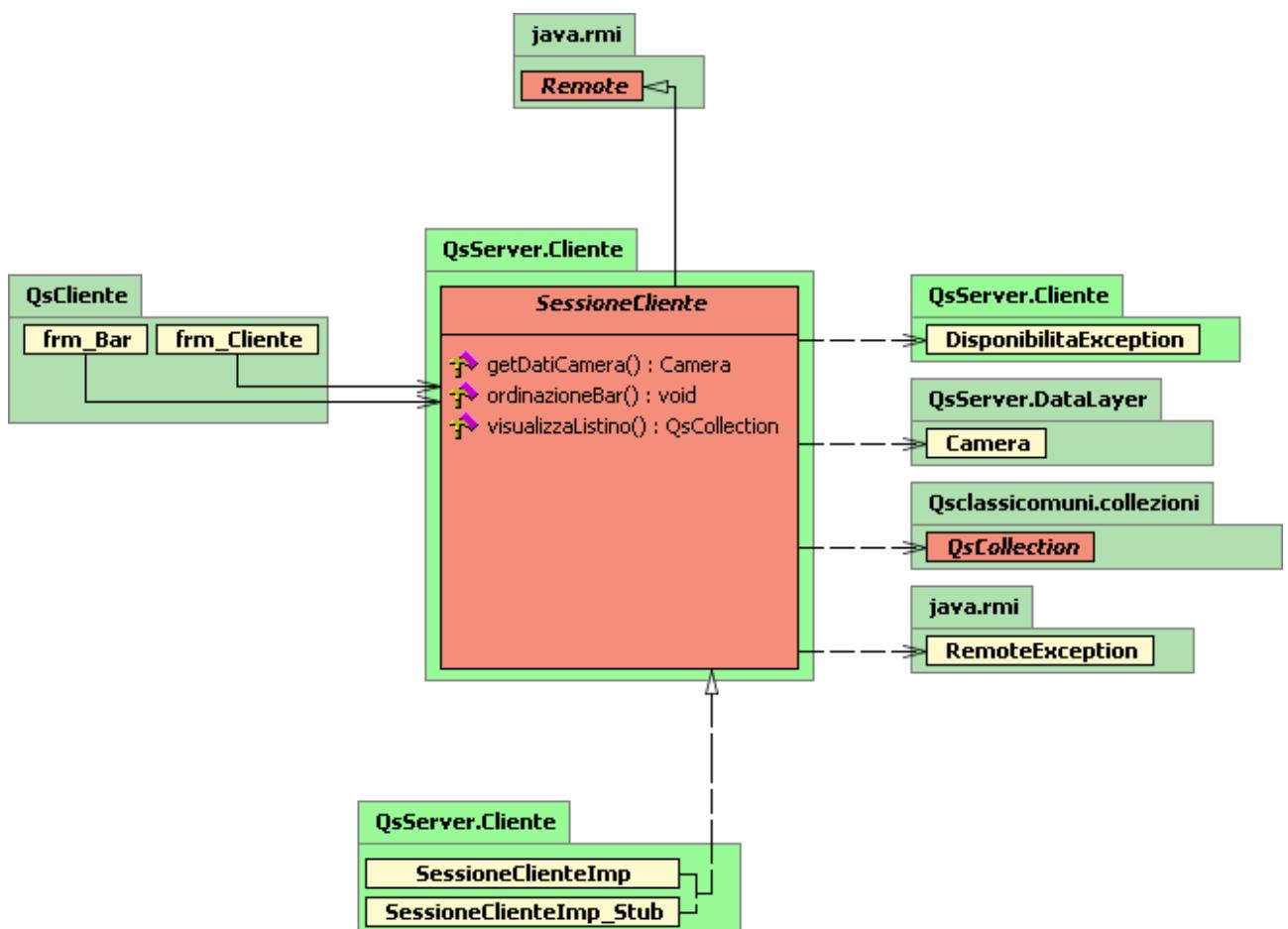
restituisce i dati della camera

**Specified by:**

getDatiCamera in interface SessioneCliente

**Returns:**

Camera dati della camera



**QsServer.DataLayer**

## **Class Camera**

`java.lang.Object`

└ `QsServer.DataLayer.Camera`

**All Implemented Interfaces:**

`java.io.Serializable`

---

```
public class Camera
```

extends `java.lang.Object`

implements `java.io.Serializable`

Title: Camera

Description: Descrive una camera della struttura

**Version:**

1.0

**Author:**

Mercurio Antonio

## Constructor Summary

**Camera()**

Costruttore di default

**Camera(int numero, double saldo, boolean occupata, boolean pensioneCompleta, int nadulti, int nbambini)**

Costruttore pubblico

## Method Summary

int	<b><u>getNadulti()</u></b>	Ritorna il numero di posti per adulti disponibili nella camera
int	<b><u>getNbambini()</u></b>	Ritorna il numero di posti per bambini nella camera
int	<b><u>getNumero()</u></b>	recupera il numero della camera
double	<b><u>getSaldo()</u></b>	ritorna il saldo
boolean	<b><u>isOccupata()</u></b>	Verifica se la camera è occupata
boolean	<b><u>isPensioneCompleta()</u></b>	verifica il tipo di pensione
void	<b><u>load(int numero, double saldo, boolean occupata, boolean pensioneCompleta, int nadulti, int nbambini)</u></b>	carica i dati relativi alla camera
void	<b><u>setNadulti(int nadulti)</u></b>	Imposta il numero di posti adulti della camera
void	<b><u>setNbambini(int nbambini)</u></b>	Imposta il numero di posti per bambini della camera
void	<b><u>setNumero(int numero)</u></b>	Imposta il numero della camera
void	<b><u>setOccupata(boolean occupata)</u></b>	Imposta lo stato della camera
void	<b><u>setPensioneCompleta(boolean pensioneCompleta)</u></b>	imposta il tipo di pensione
void	<b><u>setSaldo(double saldo)</u></b>	imposta il saldo

## Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait,  
wait, wait
```

## Constructor Detail

### Camera

```
public Camera()
```

Costruttore di default

---

### Camera

```
public Camera(int numero,  
             double saldo,  
             boolean occupata,  
             boolean pensioneCompleta,  
             int nadulti,  
             int nbambini)
```

Costruttore pubblico

#### Parameters:

numero - int numero camera

saldo - double saldo camera

occupata - boolean stato della camera true se occupata false altrimenti

pensioneCompleta - boolean tipo di pensione true se completa falsa altrimenti

nadulti - int numero di posti adulti della camera

nbambini - int numero di posti bambini della camera

## Method Detail

### load

```
public void load(int numero,  
                double saldo,  
                boolean occupata,  
                boolean pensioneCompleta,  
                int nadulti,  
                int nbambini)
```

carica i dati relativi alla camera

#### Parameters:

numero - int numero camera

saldo - double saldo camera

occupata - boolean stato della camera true se occupata false altrimenti

pensioneCompleta - boolean tipo di pensione true se completa falsa altrimenti

nadulti - int numero di posti adulti della camera

nbambini - int numero di posti bambini della camera

---

### getNadulti

```
public int getNadulti()
```

Ritorna il numero di posti per adulti disponibili nella camera

**Returns:**

int posti adulti

---

### **setNadulti**

```
public void setNadulti(int nadulti)
```

Imposta il numero di posti adulti della camera

**Parameters:**

nadulti - int posti adulti

---

### **getNbambini**

```
public int getNbambini()
```

Ritorna il numero di posti per bambini nella camera

**Returns:**

int numero bambini

---

### **setNbambini**

```
public void setNbambini(int nbambini)
```

Imposta il numero di posti per bambini della camera

**Parameters:**

nbambini - int numero bambini

---

### **getNumero**

```
public int getNumero()
```

recupera il numero della camera

**Returns:**

int numero camera

---

### **setNumero**

```
public void setNumero(int numero)
```

Imposta il numero della camera

**Parameters:**

numero - int numero camera

---

### **isOccupata**

```
public boolean isOccupata()
```

Verifica se la camera è occupata

**Returns:**

boolean true se la camera è occupata false altrimenti

---

### **setOccupata**

```
public void setOccupata(boolean occupata)
```

Imposta lo stato della camera

**Parameters:**

occupata - boolean true per se occupata false altrimenti

---

**isPensioneCompleta**

public boolean **isPensioneCompleta()**

verifica il tipo di pensione

**Returns:**

boolean se completa true, false altrimenti

---

**setPensioneCompleta**

public void **setPensioneCompleta(boolean pensioneCompleta)**

imposta il tipo di pensione

**Parameters:**

pensioneCompleta - boolean true se completa false altrimenti

---

**getSaldo**

public double **getSaldo()**

ritorna il saldo

**Returns:**

double totale saldo

---

**setSaldo**

public void **setSaldo(double saldo)**

imposta il saldo

**Parameters:**

saldo - double totale saldo

---

QsServer.DataLayer

**Class Categoria**

java.lang.Object

└ QsServer.DataLayer.Categoria

**All Implemented Interfaces:**

java.io.Serializable

---

public class **Categoria**

extends java.lang.Object

implements java.io.Serializable

Title: Categoria

Description: Descrive una categoria di prodotti

**Version:**

1.0

**Author:**

Pizza Luca Ernesto, Vicedomini Vincenzo

**Field Summary**

(package private) java.lang.String	<u>descrizione</u>
(package private) int	<u>idCategoria</u>

**Constructor Summary**

<u>Categoria()</u>	Costruttore di default
<u>Categoria(java.lang.String descrizione, int id)</u>	Costruttore pubblico

**Method Summary**

java.lang.String	<u>getDescrizione()</u> Restituisce la descrizione della categoria
int	<u>getIdCategoria()</u> Restituisce l'id della categoria
void	<u>load(int idCategoria, java.lang.String descrizione)</u> Imposta le variabili istanza
void	<u>setDescrizione(java.lang.String descrizione)</u> Setta la descrizione della Categoria
void	<u>setIdCategoria(int idCategoria)</u> Setta l'id della categoria

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Field Detail****descrizione**

```
java.lang.String descrizione
```

---

### **idCategoria**

```
int idCategoria
```

## Constructor Detail

### **Categoria**

```
public Categoria()
```

Costruttore di default

---

### **Categoria**

```
public Categoria(java.lang.String descrizione,  
                int id)
```

Costruttore pubblico

#### **Parameters:**

descrizione - String descrizione della categoria

id - int identificatore della categoria

## Method Detail

### **load**

```
public void load(int idCategoria,  
                 java.lang.String descrizione)
```

Imposta le variabili istanza

#### **Parameters:**

idCategoria - int identificatore della categoria

descrizione - String descrizione della categoria

---

### **getDescrizione**

```
public java.lang.String getDescrizione()
```

Restituisce la descrizione della categoria

#### **Returns:**

String descrizione categoria

---

### **getIdCategoria**

```
public int getIdCategoria()
```

Restituisce l'id della categoria

#### **Returns:**

int identificatore categoria

---

### **setDescrizione**

```
public void setDescrizione(java.lang.String descrizione)
```

Setta la descrizione della Categoria

#### **Parameters:**

`descrizione` - String descrizione categoria

---

### **setIdCategoria**

`public void setIdCategoria(int idCategoria)`

Setta l'id della categoria

#### **Parameters:**

`idCategoria` - int identificatore categoria

`QsServer.DataLayer`

## **Class DettaglioPrenotazioneBar**

`java.lang.Object`

└ `QsServer.DataLayer.DettaglioPrenotazioneBar`

#### **All Implemented Interfaces:**

`java.io.Serializable`

---

`public class DettaglioPrenotazioneBar`

`extends java.lang.Object`

`implements java.io.Serializable`

Title: DettaglioPrenotazioneBar

Description: Descrive elemento di una prenotazione

#### **Version:**

1.0

#### **Author:**

Vicidomini Vincenzo

---

## **Field Summary**

(package private) <code>java.lang.String</code>	<u>descrizione</u>
(package private) <code>int</code>	<u>idProdotto</u>
(package private) <code>java.lang.String</code>	<u>note</u>
(package private) <code>double</code>	<u>prezzo</u>
(package private) <code>int</code>	<u>quantita</u>

---

## **Constructor Summary**

`DettaglioPrenotazioneBar()`

Costruttore di default

```
DettaglioPrenotazioneBar(int idProdotto, java.lang.String descrizione,
int quantita, double prezzo, java.lang.String note)
Costruttore pubblico
```

## Method Summary

java.lang.String	<b><u>getDescrizione()</u></b> Restituisce la descrizione del prodotto ordinato
int	<b><u>getIdProdotto()</u></b> Restituisce l'id del prodotto ordinato
java.lang.String	<b><u>getNote()</u></b> Restituisce le note del prodotto ordinato
double	<b><u>getPrezzo()</u></b> Restituisce il prezzo del prodotto ordinato
int	<b><u>getQuantita()</u></b> Restituisce la quantita del prodotto ordinato
void	<b><u>load</u></b> (int idProdotto, java.lang.String descrizione, int quantita, double prezzo, java.lang.String note) Carica i dati
void	<b><u>setDescrizione</u></b> (java.lang.String descrizione) Setta la descrizione del prodotto ordinato
void	<b><u>setIdProdotto</u></b> (int idProdotto) Setta l'id del prodotto ordinato
void	<b><u>setNote</u></b> (java.lang.String note) Setta le note del prodotto ordinato
void	<b><u>setPrezzo</u></b> (double prezzo) Setta il prezzo del prodotto ordinato
void	<b><u>setQuantita</u></b> (int quantita) Setta la quantita del prodotto ordinato

## Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

## Field Detail

### **idProdotto**

int **idProdotto**

---

### **quantita**

```
int quantita
```

---

### **descrizione**

```
java.lang.String descrizione
```

---

### **note**

```
java.lang.String note
```

---

### **prezzo**

```
double prezzo
```

## **Constructor Detail**

### **DettaglioPrenotazioneBar**

```
public DettaglioPrenotazioneBar()
```

Costruttore di default

---

### **DettaglioPrenotazioneBar**

```
public DettaglioPrenotazioneBar(int idProdotto,  
                               java.lang.String descrizione,  
                               int quantita,  
                               double prezzo,  
                               java.lang.String note)
```

Costruttore pubblico

**Parameters:**

idProdotto - int idProdotto

descrizione - String descrizione prodotto

quantita - int quantità prodotto

prezzo - double prezzo totale (prezzo unitario prodotto \* quantità)

note - String note opzionali

## **Method Detail**

### **load**

```
public void load(int idProdotto,  
                 java.lang.String descrizione,  
                 int quantita,  
                 double prezzo,  
                 java.lang.String note)
```

Carica i dati

**Parameters:**

idProdotto - int identificatore prodotto

descrizione - String descrizione

quantita - int quantità

`prezzo` - double prezzo  
`note` - String note opzionali

---

### **getDescrizione**

`public java.lang.String getDescrizione()`  
Restituisce la descrizione del prodotto ordinato  
**Returns:**  
String descrizione prodotto

---

### **setDescrizione**

`public void setDescrizione(java.lang.String descrizione)`  
Setta la descrizione del prodotto ordinato  
**Parameters:**  
`descrizione` - String descrizione prodotto

---

### **getIdProdotto**

`public int getIdProdotto()`  
Restituisce l'id del prodotto ordinato  
**Returns:**  
int descrizione prodotto

---

### **setIdProdotto**

`public void setIdProdotto(int idProdotto)`  
Setta l'id del prodotto ordinato  
**Parameters:**  
`idProdotto` - int identificatore prodotto

---

### **getNote**

`public java.lang.String getNote()`  
Restituisce le note del prodotto ordinato  
**Returns:**  
String note opzionali

---

### **setNote**

`public void setNote(java.lang.String note)`  
Setta le note del prodotto ordinato  
**Parameters:**  
`note` - String note opzionali

---

### **getPrezzo**

`public double getPrezzo()`  
Restituisce il prezzo del prodotto ordinato  
**Returns:**

---

double prezzo totale

---

### **setPrezzo**

public void **setPrezzo**(double prezzo)

Setta il prezzo del prodotto ordinato

**Parameters:**

prezzo - double prezzo totale

---

### **getQuantita**

public int **getQuantita()**

Restituisce la quantita del prodotto ordinato

**Returns:**

int quantità

---

### **setQuantita**

public void **setQuantita**(int quantita)

Setta la quantita del prodotto ordinato

**Parameters:**

quantita - int quantità

---

QsServer.DataLayer

## **Class Prenotazione**

java.lang.Object

└ QsServer.DataLayer.Prenotazione

**All Implemented Interfaces:**

java.io.Serializable

---

public class **Prenotazione**

extends java.lang.Object

implements java.io.Serializable

Title: Prenotazione

Description: Descrive una prenotazione

**Version:**

1.0

**Author:**

Pizza Ernesto

## Constructor Summary

```
Prenotazione(int idP, int numero, java.lang.String data, char tipo,
float costo)
Costruttore pubblico
```

## Method Summary

double	<b><u>getCosto()</u></b> Ritorna il costo totale della prenotazione
java.lang.String	<b><u>getData()</u></b> Ritorna la data della prenotazione
int	<b><u>getIdPrenotazione()</u></b> Restituisce l'identificatore univoco della prenotazione
int	<b><u>getNumeroCamera()</u></b> ritorna il numero della camera
char	<b><u>getTipo()</u></b> Ritorna il tipo della prenotazione
void	<b><u>setCosto(double costo)</u></b> Imposta il costo della prenotazione
void	<b><u>setData(java.lang.String data)</u></b> Imposta la data della prenotazione
void	<b><u>setIdPrenotazione(int idPrenotazione)</u></b> Imposta l'identificatore univoco della prenotazione
void	<b><u>setNumeroCamera(int numeroCamera)</u></b> Imposta il numero della camera
void	<b><u>setTipo(char tipo)</u></b> Imposta il tipo della prenotazione

### Methods inherited from class `java.lang.Object`

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait,
wait, wait
```

## Constructor Detail

### Prenotazione

```
public Prenotazione(int idP,
                     int numero,
```

```
        java.lang.String data,
        char tipo,
        float costo)
```

Costruttore pubblico

**Parameters:**

idP - int identificatore prenotazione  
numero - int numero camera  
data - String data della prenotazione  
tipo - char tipo della prenotazione  
costo - float costo totale prenotazione

## Method Detail

### **getCosto**

```
public double getCosto()
```

Ritorna il costo totale della prenotazione

**Returns:**

double costo totale

---

### **setCosto**

```
public void setCosto(double costo)
```

Imposta il costo della prenotazione

**Parameters:**

costo - double costo totale

---

### **getData**

```
public java.lang.String getData()
```

Ritorna la data della prenotazione

**Returns:**

String data prenotazione

---

### **setData**

```
public void setData(java.lang.String data)
```

Imposta la data della prenotazione

**Parameters:**

data - String data

---

### **getIdPrenotazione**

```
public int getIdPrenotazione()
```

Restituisce l'identificatore univoco della prenotazione

**Returns:**

int identificatore prenotazione

---

### **setIdPrenotazione**

```
public void setIdPrenotazione(int idPrenotazione)
```

Imposta l'identificatore univoco della prenotazione

**Parameters:**

idPrenotazione - int

---

### **setNumeroCamera**

public void **setNumeroCamera**(int numeroCamera)

Imposta il numero della camera

**Parameters:**

numeroCamera - int numero camera

---

### **setTipo**

public void **setTipo**(char tipo)

Imposta il tipo della prenotazione

**Parameters:**

tipo - char tipo prenotazione

---

### **getTipo**

public char **getTipo**()

Ritorna il tipo della prenotazione

**Returns:**

char tipo prenotazione

---

### **getNumeroCamera**

public int **getNumeroCamera**()

ritorna il numero della camera

**Returns:**

int numero camera

QsServer.DataLayer

## **Class ProdottoBar**

java.lang.Object

└ QsServer.DataLayer.ProdottoBar

### **All Implemented Interfaces:**

java.io.Serializable

---

public class **ProdottoBar**

extends java.lang.Object

implements java.io.Serializable

Title: ProdottoBar

Description: Descrive un prodotto del bar

**Version:**

1.0

**Author:**

Mercurio Antonio

## Constructor Summary

**ProdottoBar()**

Costruttore di default

**ProdottoBar(int codCategoria, java.lang.String descrizione, double prezzo, int disponibilita)**

Costruttore avanzato

## Method Summary

int	<b><u>getCodCategoria()</u></b> ritorna il codice della categoria
java.lang.String	<b><u>getDescrizione()</u></b> ritorna la descrizione del prodotto
int	<b><u>getDisponibilita()</u></b> ritorna la disponibilità di un prodotto
int	<b><u>getIdProdotto()</u></b> ritorna l'identificatore univoco del prodotto
double	<b><u>getPrezzo()</u></b> ritorna il prezzo unitario del prodotto
void	<b><u>load(int idProdotto, int codCategoria, java.lang.String descrizione, double prezzo, int disponibilita)</u></b> Carica i dati dall'esterno
void	<b><u>setCodCategoria(int codCategoria)</u></b> imposta codice categoria
void	<b><u>setDescrizione(java.lang.String descrizione)</u></b> imposta descrizione prodotto
void	<b><u>setDisponibilita(int disponibilita)</u></b> imposta la disponibilità di un prodotto
void	<b><u>setIdProdotto(int idProdotto)</u></b> imposta l'identificatore del prodotto
void	<b><u>setPrezzo(double prezzo)</u></b> imposta il prezzo unitario del prodotto

### Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

## Constructor Detail

## **ProdottoBar**

```
public ProdottoBar()  
    Costruttore di default
```

---

## **ProdottoBar**

```
public ProdottoBar(int codCategoria,  
                    java.lang.String descrizione,  
                    double prezzo,  
                    int disponibilita)  
    Costruttore avanzato  
Parameters:  
codCategoria - int codice categoria  
descrizione - String descrizione prodotto  
prezzo - double prezzo unitario prodotto  
disponibilita - int disponibilità (-1 per disponibilità ILLIMITATA)
```

## **Method Detail**

### **load**

```
public void load(int idProdotto,  
                  int codCategoria,  
                  java.lang.String descrizione,  
                  double prezzo,  
                  int disponibilita)
```

Carica i dati dall'esterno

#### **Parameters:**

```
idProdotto - int nome del prodotto  
codCategoria - int codice categoria  
descrizione - String descrizione prodotto  
prezzo - double prezzo unitario prodotto  
disponibilita - int disponibilità (-1 per disponibilità ILLIMITATA)
```

---

### **getCodCategoria**

```
public int getCodCategoria()  
    ritorna il codice della categoria  
Returns:  
int codice categoria
```

---

### **setCodCategoria**

```
public void setCodCategoria(int codCategoria)  
    imposta codice categoria  
Parameters:  
codCategoria - int codice categoria
```

---

### **getDisponibilita**

```
public int getDisponibilita()  
    ritorna la disponibilità di un prodotto
```

**Returns:**  
int disponibilità

---

### **setDisponibilità**

```
public void setDisponibilità(int disponibilità)
    imposta la disponibilità di un prodotto
```

**Parameters:**  
disponibilità - int disponibilità prodotto

---

### **getDescrizione**

```
public java.lang.String getDescrizione()
    ritorna la descrizione del prodotto
```

**Returns:**  
String descrizione prodotto

---

### **setDescrizione**

```
public void setDescrizione(java.lang.String descrizione)
    imposta descrizione prodotto
```

**Parameters:**  
descrizione - String descrizione prodotto

---

### **getPrezzo**

```
public double getPrezzo()
    ritorna il prezzo unitario del prodotto
```

**Returns:**  
double prezzo prodotto

---

### **setPrezzo**

```
public void setPrezzo(double prezzo)
    imposta il prezzo unitario del prodotto
```

---

### **getIdProdotto**

```
public int getIdProdotto()
    ritorna l'identificatore univoco del prodotto
```

**Returns:**  
int identificatore prodotto

---

### **setIdProdotto**

```
public void setIdProdotto(int idProdotto)
    imposta l'identificatore del prodotto
```

**Parameters:**  
idProdotto - int identificatore prodotto

---

**QsServer.DataLayer**

## Class Utente

java.lang.Object

└ QsServer.DataLayer.Utente

### All Implemented Interfaces:

java.io.Serializable

---

```
public class Utente
extends java.lang.Object
implements java.io.Serializable
```

Title: Utente

Description: Descrive l'utente che si logga al sistema

### Version:

1.0

### Author:

Iacoletti Alessandro

---

## Field Summary

(package private) java.lang.String	<u>cognome</u>
(package private) java.lang.String	<u>login</u>
(package private) java.lang.String	<u>nome</u>
(package private) java.lang.String	<u>pwd</u>

---

## Constructor Summary

Utente()

Costruttore pubblico di default

---

## Method Summary

java.lang.String	<u>getCognome()</u> ritorna il cognome dell'utente
java.lang.String	<u>getLogin()</u> Ritorna il login name dell'utente
java.lang.String	<u>getNome()</u>

	ritorna il nome dell'utente
java.lang.String	<b><u>getPwd()</u></b> ritorna la password dell'utente
void	<b><u>setCognome</u></b> ( java.lang.String cognome ) imposta il cognome dell'utente
void	<b><u>setLogin</u></b> ( java.lang.String login ) imposta il login name
void	<b><u>setNome</u></b> ( java.lang.String nome ) imposta il nome utente
void	<b><u>setPwd</u></b> ( java.lang.String pwd ) imposta la password dell'utente

### Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

## Field Detail

### nome

java.lang.String **nome**

---

### cognome

java.lang.String **cognome**

---

### login

java.lang.String **login**

---

### pwd

java.lang.String **pwd**

## Constructor Detail

### Utente

public **Utente()**

Costruttore pubblico di default

## Method Detail

### getCognome

public java.lang.String **getCognome()**

ritorna il cognome dell'utente

**Returns:**

String cognome utente

---

### **getLogin**

public java.lang.String **getLogin()**

Ritorna il login name dell'utente

**Returns:**

String login name

---

### **getNome**

public java.lang.String **getNome()**

ritorna il nome dell'utente

**Returns:**

String nome utente

---

### **getPwd**

public java.lang.String **getPwd()**

ritorna la password dell'utente

**Returns:**

String password

---

### **setCognome**

public void **setCognome**(java.lang.String cognome)

imposta il cognome dell'utente

**Parameters:**

cognome - String cognome

---

### **setLogin**

public void **setLogin**(java.lang.String login)

imposta il login name

**Parameters:**

login - String login name

---

### **setNome**

public void **setNome**(java.lang.String nome)

imposta il nome utente

**Parameters:**

nome - String nome dell'utente

---

### **setPwd**

public void **setPwd**(java.lang.String pwd)

imposta la password dell'utente

**Parameters:**

pwd - String password utente

**Qsclassicomuni.adapters**

## **Class QsCollectionAdapter**

```
java.lang.Object
└ java.util.AbstractCollection
  └ java.util.AbstractList
    └ java.util.ArrayList
      └ Qsclassicomuni.adapters.QsCollectionAdapter
```

### **All Implemented Interfaces:**

java.lang.Cloneable, java.util.Collection, java.util.List, [QsCollection](#),  
java.util.RandomAccess, java.io.Serializable

---

```
public class QsCollectionAdapter
extends java.util.ArrayList
implements QsCollection, java.io.Serializable
```

Title: QsCollectionAdapter

Description: Questa Classe invece ha il ruolo di adapter cioè fa da tramite tra la classe target e la classe Adaptee, ovvero la classe che fornisce l'implementazione al target, che nel nostro caso è la classe ArrayList. L'utilizzo di questa classe rende l'applicazione molto versatile e malleabile è possibile cambiare in ogni momento la struttura dati della collezione senza dover apportare modifiche in tutte le classi client della collezione stessa. Ogni invocazione di metodi della classe Adapter vengono indirizzate tutte ai metodi della classe adaptee

**Version:**

1.0

**Author:**

Pacifico Marta, Antonio Mercurio

---

## **Nested Class Summary**

### **Nested classes inherited from class java.util.AbstractList**

## Field Summary

### Fields inherited from class java.util.AbstractList

modCount

## Constructor Summary

[QsCollectionAdapter\(\)](#)

Costruttore pubblico

## Method Summary

void	<u><a href="#">aggiungi</a></u> (java.lang.Object o)	Aggiunge un elemento alla collection
int	<u><a href="#">dimensione</a></u> ()	Dimensione cioè numero di elementi all'interno della collection
void	<u><a href="#">modifica</a></u> (int i, java.lang.Object o)	Modifica un elemento all'interno della collection
java.lang.Object	<u><a href="#">preleva</a></u> (int i)	ritorna un elemento della collection
void	<u><a href="#">rimuovi</a></u> (int i)	Elimina un elemento dalla colecion
void	<u><a href="#">svuota</a></u> (java.util.Collection c)	Elimina tutti gli elementi dalla collection

### Methods inherited from class java.util.ArrayList

add, add, addAll, addAll, clear, clone, contains, ensureCapacity, get, indexOf, isEmpty, lastIndexOf, remove, removeRange, set, size, toArray, toArray, trimToSize

### Methods inherited from class java.util.AbstractList

equals, hashCode, iterator, listIterator, listIterator, subList

### Methods inherited from class java.util.AbstractCollection

```
containsAll, remove, removeAll, retainAll, toString
```

### Methods inherited from class java.lang.Object

```
finalize, getClass, notify, notifyAll, wait, wait, wait
```

### Methods inherited from interface java.util.List

```
containsAll, equals, hashCode, iterator, listIterator, listIterator, remove,  
removeAll, retainAll, subList
```

## Constructor Detail

### **QsCollectionAdapter**

```
public QsCollectionAdapter()  
Costruttore pubblico
```

## Method Detail

### **aggiungi**

```
public void aggiungi(java.lang.Object o)
```

Aggiunge un elemento alla collection

**Specified by:**

[aggiungi](#) in interface [QsCollection](#)

**Parameters:**

o - Object elemento da aggiungere

---

### **dimensione**

```
public int dimensione()
```

Dimensione cioè numero di elementi all'interno della collection

**Specified by:**

[dimensione](#) in interface [QsCollection](#)

**Returns:**

int dimensione

---

### **preleva**

```
public java.lang.Object preleva(int i)
```

ritorna un elemento della collection

**Specified by:**

[preleva](#) in interface [QsCollection](#)

**Parameters:**

i - int indice dell'elemento da prelevare

**Returns:**

Object elemento della collection

---

## modifica

```
public void modifica(int i,  
                      java.lang.Object o)
```

Modifica un elemento all'interno della collection

**Specified by:**

[modifica](#) in interface [QsCollection](#)

**Parameters:**

i - int indice dell'elemento da modificare

o - Object elemento da modificare

---

## rimuovi

```
public void rimuovi(int i)
```

Elimina un elemento dalla colecion

**Specified by:**

[rimuovi](#) in interface [QsCollection](#)

**Parameters:**

i - int indice dell'elemento da rimuovere

---

## svuota

```
public void svuota(java.util.Collection c)
```

Elimina tutti gli elementi dalla collection

**Specified by:**

[svuota](#) in interface [QsCollection](#)

**Parameters:**

c - Collection collection vuota

---

## Qsclassicomuni.collezioni

### Class CollectionListino

java.lang.Object

└ com.borland.dx.dataset.ReadRow

  └ com.borland.dx.dataset.ReadWriteRow

    └ com.borland.dx.dataset.DataSet

      └ com.borland.dx.dataset.StorageDataSet

      └ com.borland.dx.sql.dataset.QueryDataSet

      └ **Qsclassicomuni.collezioni.CollectionListino**

**All Implemented Interfaces:**

com.borland.dx.dataset.AccessListener, com.borland.dx.dataset.ColumnDesigner,  
com.borland.dx.dataset.Designable, java.util.EventListener,  
com.borland.dx.dataset.MasterNavigateListener, java.io.Serializable,  
com.borland.dx.dataset.StatusListener

---

```
public class CollectionListino  
extends com.borland.dx.sql.dataset.QueryDataSet  
implements java.io.Serializable
```

Title: CollectionListino

Description: Collezione di oggetti di tipo prodottoBar che compongono il listino

**Version:**

1.0

**Author:**

Vicidomini Vincenzo

## Field Summary

### Fields inherited from class com.borland.dx.dataset.StorageDataSet

## Constructor Summary

[CollectionListino\(\)](#)

## Method Summary

java.lang.String	<u><a href="#">getCategoria()</a></u> ritorna la categoria dell'elemento
java.lang.String	<u><a href="#">getDescrizione()</a></u> Ritorna la descrizione dall'elemento nella collection
int	<u><a href="#">getDisponibilita()</a></u> ritorna la disponibilità del prodotto, -1 per ILLIMITATO
double	<u><a href="#">getPrezzo()</a></u> ritorna il prezzo
void	<u><a href="#">setDescrizione(java.lang.String descrizione)</a></u> Imposta la descrizione

### Methods inherited from class com.borland.dx.sql.dataset.QueryDataSet

`closeStatement, executeQuery, getDatabase, getOriginalQueryString,`

```
getParameterRow, getQuery, getQueryString, isAccumulateResults, refetchRow,  
refresh, refreshSupported, saveChanges, saveChangesSupported,  
setAccumulateResults, setProvider, setQuery
```

## Methods inherited from class com.borland.dx.dataset.StorageDataSet

```
addCalcAggFieldsListener, addCalcFieldsListener,addColumn,addColumn,  
addColumn,addColumnChangeListener, addEditListener, addForeignKey,  
addLoadListener, addLoadRowListener, addUniqueColumn, cancelLoading,  
cancelOperation, changeColumn, changesPending, cloneColumns,  
cloneDataSetStructure, closeProvider, deleteDuplicates, dropAllIndexes,  
dropColumn, dropColumn, dropIndex, empty, endLoading, freeAllIndexes,  
getCalcAggFieldsListener, getCalcFieldsListener, getDataFile,  
getDeletedRowCount, getDeletedRows, getDuplicates, getForeignKeys,  
getInsertedRowCount, getInsertedRows, getLocale, getMaxDesignRows,  
getMaxResolveErrors, getMaxRows, getMetaDataUpdate, getNeedsRestructure,  
getOriginalRow, getProvider, getReferenceForeignKeys, getResolveOrder,  
getResolver, getSchemaName,getStore, getStoreClassFactory, getStoreName,  
getTableName, getUpdatedRowCount, getUpdatedRows, hasRowIds, indexExists,  
isReadOnly, isResolvable, loadRow, loadRow, moveColumn, postAllDataSets,  
provideMoreData, recalc, removeCalcAggFieldsListener, removeCalcFieldsListener,  
removeColumnChangeListener, removeEditListener, removeForeignKey,  
removeLoadListener, removeLoadRowListener, reset, resetPendingStatus,  
resetPendingStatus, restructure, setAllRowIds, setColumns, setDataFile,  
setForeignKeys, setLocale, setMaxDesignRows, setMaxResolveErrors, setMaxRows,  
setMetaDataUpdate, setReadOnly, setResolvable, setResolveOrder, setResolver,  
setRowIndex, setSchemaName, setStore, setStoreClassFactory, setStoreName,  
setTableName, startLoading, startLoading, startLoading
```

## Methods inherited from class com.borland.dx.dataset.DataSet

```
accessChange, addAccessListener, addDataChangeListener,  
addMasterNavigateListener, addNavigationListener, addOpenListener, addRow,  
addRowFilterListener, addRowReturnInternalRow, addStatusListener,  
allocateValues, atFirst, atLast, cancel, canNavigate, canSet, clearStatus,  
cloneDataSetView, close, columnIsVisible, deleteAllRows, deleteRow, dittoRow,  
dittoRow, dropIndex, editRow, emptyAllRows, emptyRow, enableDataSetEvents,  
first, format, format, getDataRow, getDataRow, getDetail, getDetails,  
getDisplayVariant, getInternalRow, getLastColumnVisited, getMasterLink, getRow,  
getRowCount, getRowFilterListener, getSort, getStatus, getStorageDataSet,  
getVariant, getVariant, goToClosestRow, goToInternalRow, goToRow, goToRow,  
hasDetail, hasValidations, inBounds, insertRow, interactiveLocate,  
isDetailDataSetWithFetchAsNeeded, isDisplayErrors, isEditable, isEditing,  
isEditingNewRow, isEmpty, isEnableDelete, isEnableInsert, isEnableUpdate,  
isModified, isModified, isNew, isOpen, isSortable, last, locate, lookup,  
masterNavigated, masterNavigating, moveRow, next, open, openDetails, post,  
prior, refilter, removeAccessListener, removeDataChangeListener,  
removeMasterNavigateListener, removeNavigationListener, removeOpenListener,  
removeRowFilterListener, removeStatusListener, resetInBounds, row, rowCount,  
saveChanges, setDefaultValues, setDefaultValues, setDisplayErrors,  
setDisplayVariant, setEditable, setEnableDelete, setEnableInsert,  
setEnableUpdate, setLastColumnVisited, setMasterLink, setSort, startEdit,  
startEditCheck, statusMessage, statusMessage, toggleViewOrder, updateRow,  
validate, validate
```

### Methods inherited from class com.borland.dx.dataset.ReadWriteRow

```
clearValues, requiredColumnsCheck, requiredColumnsCheck, setAssignedNull,  
setAssignedNull, setBigDecimal, setBigDecimal, setBinaryStream, setBinaryStream,  
setBoolean, setBoolean, setByte, setByte, setByteArray, setByteArray, setDate,  
setDate, setDate, setDefault, setDefault, setDouble, setDouble,  
setFloat, setFloat, setInputStream, setInputStream, setInt, setInt, setLong,  
setLong, setObject, setObject, setShort, setShort, setString, setString,  
setTime, setTime, setTime, setTime, setTimestamp, setTimestamp, setTimestamp,  
setTimestamp, setUnassignedNull, setUnassignedNull, setVariant, setVariant
```

### Methods inherited from class com.borland.dx.dataset.ReadRow

```
columnCount, copyTo, copyTo, equals, findDifference, findModified, findOrdinal,  
getArrayLength, getBigDecimal, getBigDecimal, getBinaryStream, getBinaryStream,  
getBoolean, getBoolean, getByte, getByte, getByteArray, getByteArray, getColumn,  
getColumn, getColumnCount, getColumnNames, getColumns, getDate, getDate,  
getDouble, getDouble, getFloat, getFloat, getInputStream, getInputStream,  
getInt, getInt, getLong, getLong, getObject, getObject, getShort, getShort,  
getString, getString, getStringPadded, getStringPadded, getTime, getTime,  
getTimestamp, getTimestamp, getVariant, getVariant, getVariants, hasColumn,  
isAssignedNull, isAssignedNull, isCompatibleList, isModified, isNull, isNull,  
isUnassignedNull, isUnassignedNull, toString
```

### Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait
```

### Methods inherited from interface com.borland.dx.dataset.ColumnDesigner

```
getColumn, getColumnCount, hasColumn
```

## Constructor Detail

### CollectionListino

```
public CollectionListino()
```

## Method Detail

### getDescrizione

```
public java.lang.String getDescrizione()
```

Ritorna la descrizione dall'elemento nella collection

**Returns:**

String descrizione elemento

---

### getCategoria

public java.lang.String **getCategoria()**

ritorna la categoria dell'elemento

**Returns:**

String categoria

---

### getPrezzo

public double **getPrezzo()**

ritorna il prezzo

**Returns:**

double prezzo

---

### getDisponibilità

public int **getDisponibilità()**

ritorna la disponibilità del prodotto, -1 per ILLIMITATO

**Returns:**

int disponibilità

---

### setDescrizione

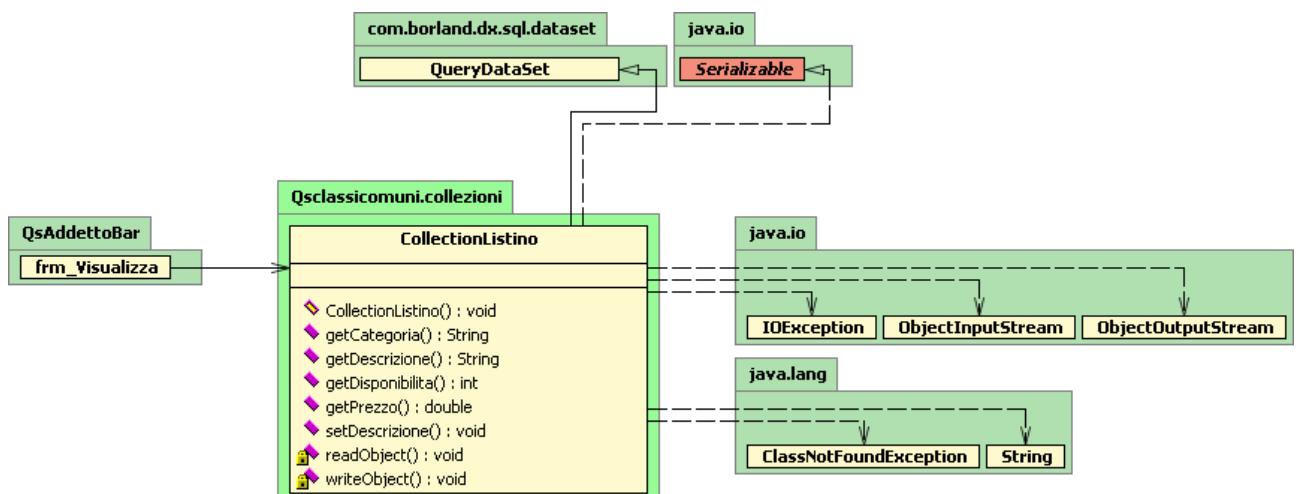
public void **setDescrizione**(java.lang.String descrizione)

Imposta la descrizione

**Parameters:**

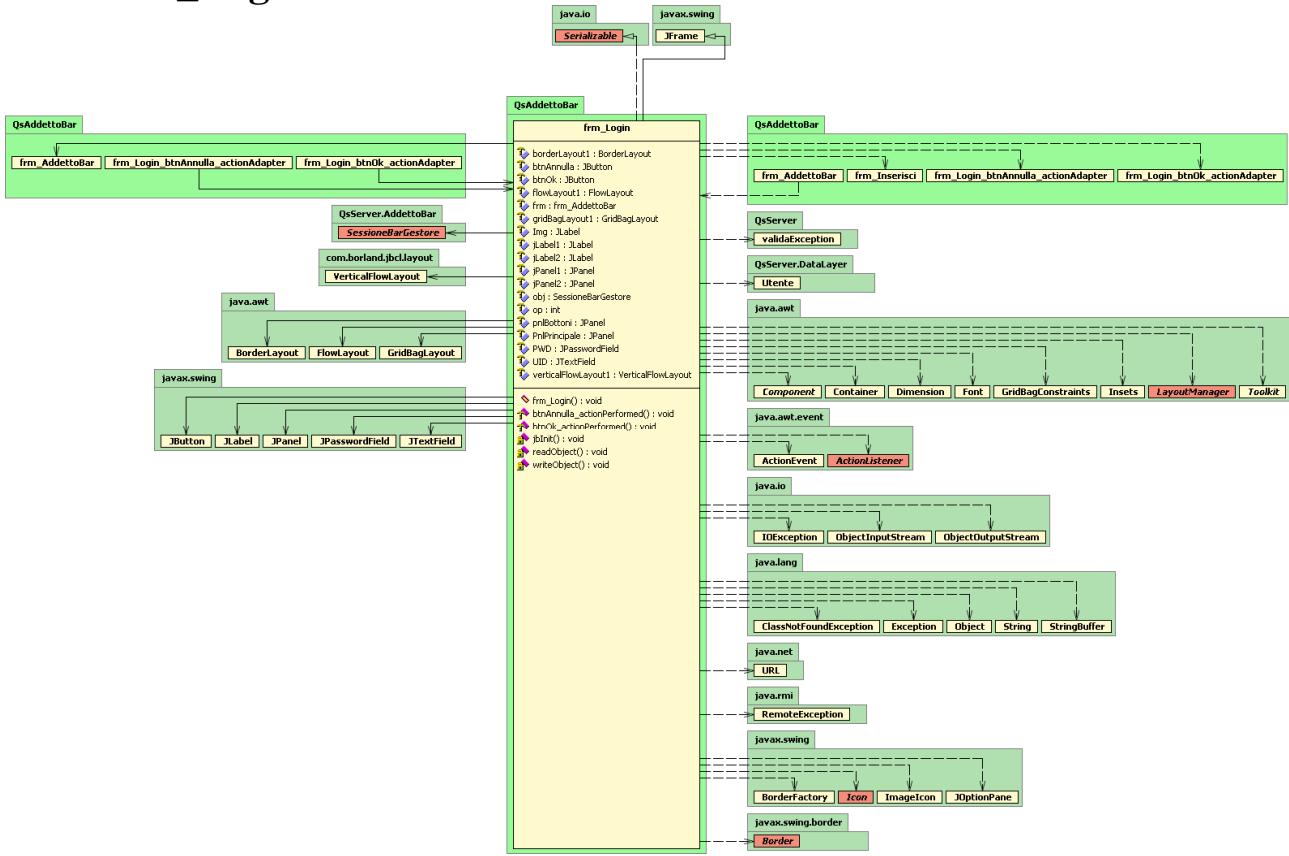
descrizione - String descrizione elemento

---



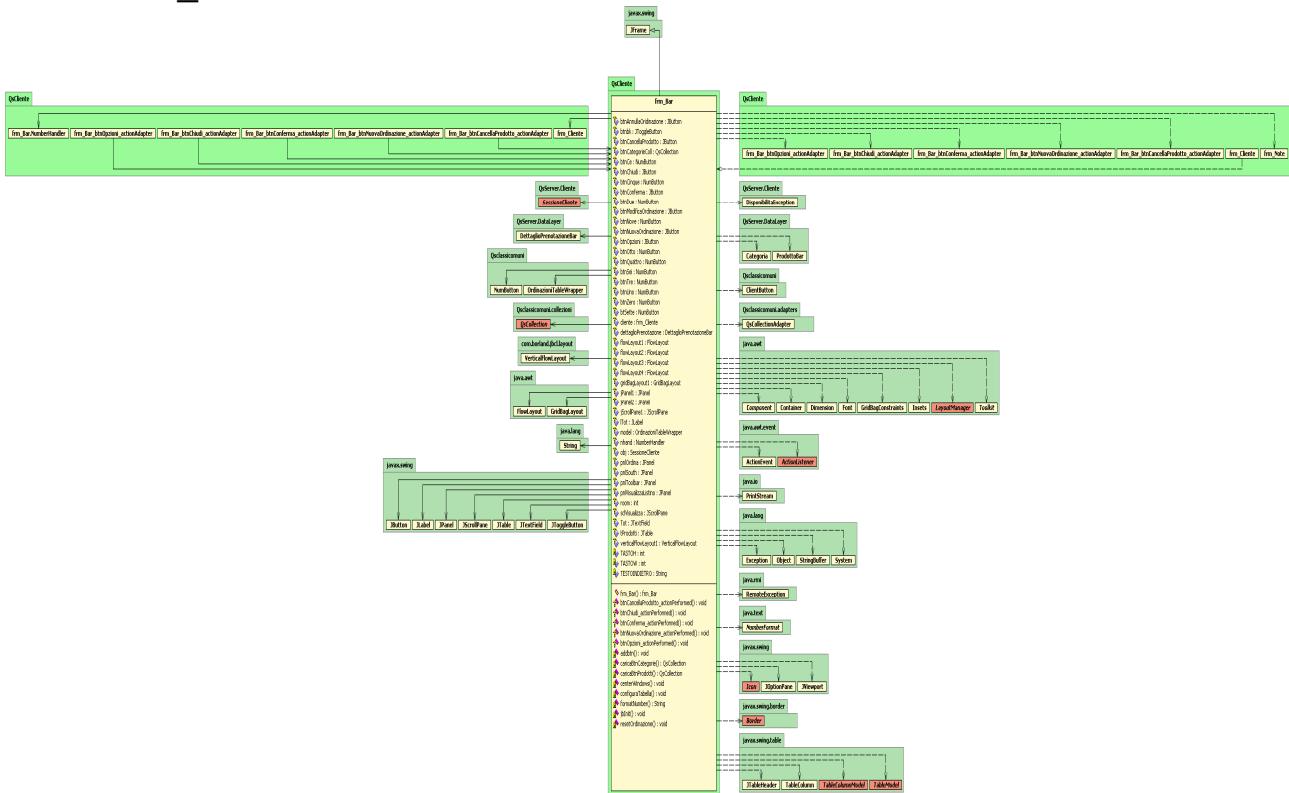
QsAddettoBar

**QsCliente**  
**QsAddettoBar**  
**Class frm\_Login**



## **QsCliente**

# **Class frm\_Bar**







# Università degli Studi di Salerno

Corso di Ingegneria del Software

## Quick Service Test Manual

**Versione 2.0**



Data <15/07/2008>

## **Revision History:**

Version R0.1 9/15/98 Robin Loh. Created

Version R0.2 9/14/98 Joyce Johnstone. revised template

Version R1.0 07/05/2008 Pacifico Marta – aggiunta paragrafi 3.2, dal 3.4 al 5.2

Version R1.0 10/05/2008 Pizza Ernesto Luca – aggiunta paragrafi 3.3, dal 3.6 al 3.7.2

Version R1.0 12/06/2008 Iacoletti Alessandro – aggiunta paragrafi 1, 2, 3.1, dal 3.8 al 3.8.2

Version R1.0 14/07/2008 Vicedomini Vincenzo, Pizza Ernesto, Pacifico Marta, – aggiunta paragrafi 2.3, 3.9

Version R2.0 14/07/2008 Iacoletti Alessandro, Mercurio Antonio, Pizza Ernesto – aggiunta paragrafo 4

Version R2.0 15/07/2008 Iacoletti Alessandro – revisione del documento.

Paragrafo 5

Version R1.0 18/07/2008 Pizza Ernesto – aggiunta paragrafo 5

**Preface:**

This document addresses the requirements of the Quick Service system. The intended audience for this document are the designers and the clients of the project.

**Target Audience:**

Client, Developers

**Quick Service Members:****Project Manager:**

Iacoletti Alessandro

**Team Members:**

Vicidomini Vincenzo

Pacifico Marta

Pizza Ernesto

Mercurio Antonio

Premessa	5
1. Introduzione	6
1.1 Organizzazione del documento	6

1.2	Scopo del documento	7
2.	Test Plan	9
2.1	Premessa	9
2.2	Risorse	10
2.3	Identificazione dei Test Item	10
2.4	Metodologie	11
2.5	Attività di Testing	11
	2.5.1 Visualizza ordine	12
	2.5.2 Nuova ordinazione	12
	2.5.3 Visualizza listino	12
	2.5.4 Inserimento prodotto	13
	2.5.5 Valida	13
3.	Test	14
3.1	Caratteristiche generali	14
3.2	Descrizione dettagliata	14
3.3	Descrizione dei casi di Test	15
3.4	Visualizza ordinazioni	15
3.5	Nuova ordinazione	15
	3.5.1 Test Design Specification	16
	3.5.2 Test Case Specification	18
3.6	Visualizza listino	26
3.7	Inserimento prodotto	28
	3.7.1 Test Design Specification	28
	3.7.2 Test Case Specification	28
3.8	Valida	35
	3.8.1 Test Design Specification	35
	3.8.2 Test Case Specification	37
3.9	Test degli oggetti remoti	44
4.	Test Log	46
4.1	Descrizione dei test log	46
4.2	Test log nuova ordinazione	47
4.3	Test log inserimento prodotto	55
4.4	Test log valida	62
5.	Descrizione classi di testo	69
5.1	Test della Collection	69
5.2	Test della Sessione Bar Gestore	71
5.3	Test della Sessione Cliente	75

## Premessa

Il documento contiene la descrizione delle attività di testing effettuate sull’applicazione denominata *QUICK SERVICE (Documento di testing)*.

Come tale, in base agli standard metodologici, il documento si concentra sulla descrizione della pianificazione di tali attività, delle tecniche utilizzate e dei risultati ottenuti in fase di esecuzione.

In questo senso, il documento mira a valutare l’aspetto qualitativo del software sviluppato attuando tecniche di analisi dinamica, cioè di tecniche utilizzate per osservare il comportamento del sistema in fase di esecuzione sulla base di particolari dati di ingresso.

## 1. Introduzione

## **1.1 Organizzazione del documento**

Il documento può essere suddiviso essenzialmente in due parti fondamentali :

- 1. documenti di pianificazione e specifica** : comprende il *Test Plan* (TP), il *Test Design Specification* (TDS), e il *Test Case Specification* (TCS).
- 2. documenti di esecuzione** : costituito dal *Test Log* (TL).

Il *Capitolo 1* rappresenta un introduzione al documento di *testing*.

Il *Capitolo 2* rappresenta essenzialmente il *Test Plan* che descrive l'oggetto, l'approccio generale, le risorse e lo scheduling delle attività da realizzare e contiene una descrizione dei test item, le caratteristiche da testare, le attività di testing.

Il *Capitolo 3* è rappresentato dalle funzionalità del sistema che noi andremo a testare tramite lo sviluppo di due paragrafi : *Test Design Specification* (TDS), e il *Test Case Specification* (TCS).

Il *Capitolo 4*, infine, rappresenta il *Test Log*, ovvero la banca dati della memorizzazione sistematica, strutturata ed in ordine cronologico di tutti i dettagli rilevanti sulla esecuzione dei test. In particolare le informazioni fondamentali di tale documento sono il successo o l'insuccesso dei test, l'occorrenza e la descrizione di eventi anomali e di testincident.

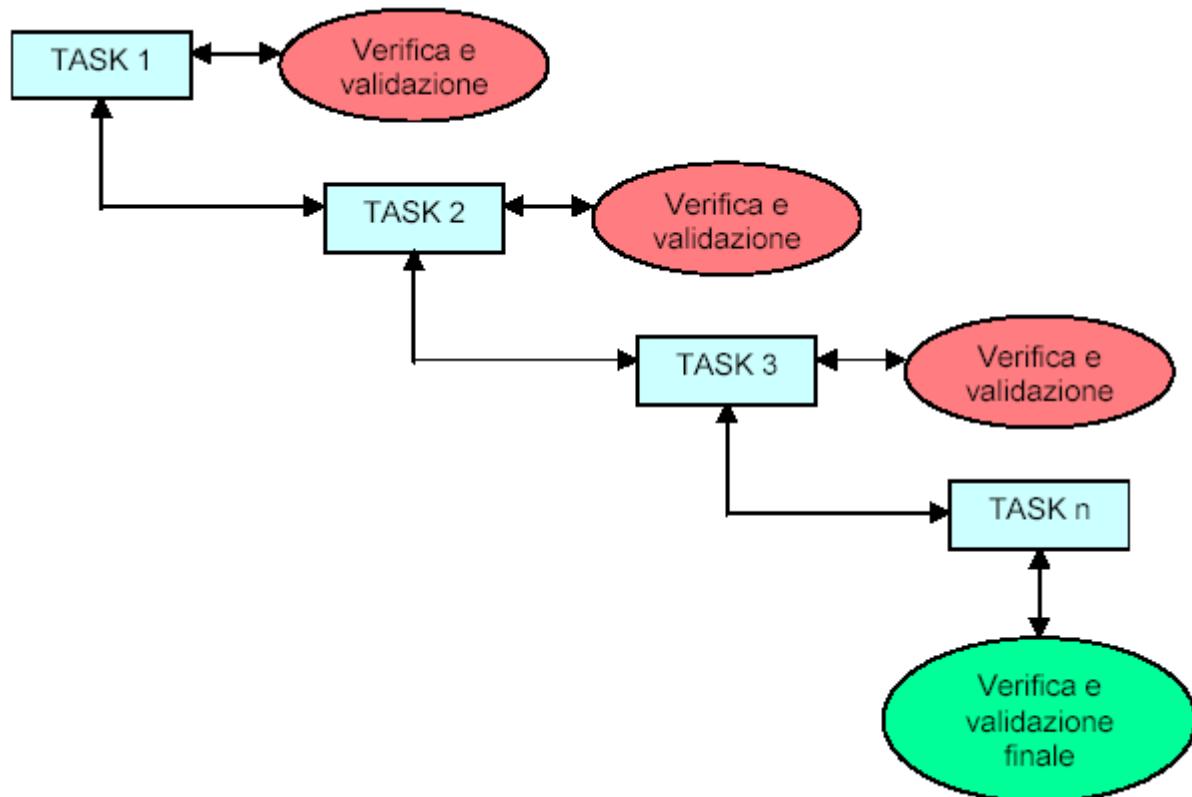
## **1.2. Scopo del documento**

La realizzazione di un buon sistema software richiede che al centro dei processi di sviluppo, manutenzione ed evoluzione del software stesso vi sia il concetto di *qualità*. E' quindi necessario utilizzare metodi quantitativi che permettano di raggiungere tale obiettivo.

Il *Testing* rappresenta senza ombra di dubbio una delle più importanti tecniche per il problema della qualità del software, in quanto consente di analizzare, valutare e quindi promuovere il miglioramento della correttezza dell'implementazione con riferimento alle caratteristiche definite in particolare dal modello dei requisiti software.

Il sistema *QUICK SERVICE* ha come obiettivo la realizzazione di una serie di servizi che vanno ad incrementare ed a rendere più agevoli le funzionalità di un albergo (ordinazioni dalla propria camera tramite touch screen, etc...)

Il presente documento, nato dall'accurata analisi del modello dei requisiti software, contiene la specifica del documento di testing per l'applicazione denominata *QUICK SERVICE (Documento di testing)*. Tale documento contiene la descrizione dettagliata della pianificazione e quindi della specifica delle attività di testing, la descrizione dei documenti di esecuzione e rappresenta quindi una essenziale sorgente di informazioni per la valutazione e la certificazione di qualità del sistema software in esame.



## **2. Test Plan**

### **2.1 Premessa**

Una pianificazione attenta ed accurata risulta necessaria per poter ottenere i risultati migliori dal processo di testing. E' ovvio che il raggiungimento di tale obiettivo richieda che la pianificazione parta presto nel processo di sviluppo del sistema software, ricordando però di non ridurre il test ad una semplice fase di tale processo e di far sì che l'attività di analisi e di test siano presenti per tutta la durata del ciclo di vita. In questo modo sarà possibile pianificare il processo per garantire visibilità il prima possibile e continuamente. Il piano di testing progettato rispetta la suddivisione in moduli del sistema software *QUICK SERVICE*, in particolare verrà testato:

- 1. modulo BAR:** insieme dei servizi offerti ai clienti per effettuare ordinazioni al bar dell'albergo.

Il piano di testing ha quindi l'obiettivo di testare il corretto funzionamento dei servizi nel modulo.

In particolare, i servizi del *modulo BAR* che verranno testati saranno:

- *Visualizza ordine;*
- *nuova ordinazione;*
- *visualizza listino;*
- *inserimento prodotto;*
- *valida;*

## 2.2 Risorse

L'esecuzione delle attività di testing verrà effettuata in ambiente distribuito, cioè mediante l'interazione di due o più macchine collegate tra esse da uno switch e da alcuni cavi di rete e richiede su ciascuna di esse l'utilizzo delle seguenti risorse :

- JAVA Virtual Machine;
- Driver JDBC-ODBC

## 2.3 Identificazione dei test item

Gli oggetti fondamentali delle attività di testing pianificate saranno le seguenti applicazioni (codice oggetto) realizzate in *jbuilder X*:

- *Quick Service - Server*;
- *Quick Service - Cliente*: interfaccia per l'accesso al S.I.O. da parte del cliente;
- *Quick Service – Addetto Bar*: interfaccia per l'accesso al S.I.O. da parte dell'addetto bar.

Le caratteristiche da testare per il controllo del corretto funzionamento di ciascuna funzionalità saranno :

- **robustezza**: capacità del sistema di reagire fornendo input non valido per il dominio applicativo;
- **usabilità**: analisi di ogni forma di iterazione corrisposta da messaggi di aiuto (in caso di errore) o di notifica (in caso di operazioni eseguite con successo) ;
- **sicurezza**: verrà testato che un utente col solo interfacciamento grafico non possa intaccare dati non inerenti alla specifica dell'operazione o accedere a dati non consentiti;
- **correttezza**: verrà testato che le operazioni vengano eseguite correttamente così come dalla specifica dei requisiti sono stati designati.

## 2.4 Metodologie di testing

Nella fase di testing abbiamo utilizzato due diverse metodologie :

- Per gli oggetti remoti sono state usate le classi di test implementate direttamente con il framework JUnit. La scelta di tale framework è dovuta a diverse ragioni:
  1. E' un progetto open-source quindi è liberamente utilizzabile.
  2. Permette di separare i test dal codice.
  3. E' facile da integrare nel processo di compilazione.
  4. È integrabile negli ambienti di sviluppo IDE.
- Per il test del reale comportamento del sistema abbiamo usato un tipo di test manuale, ovvero la semplice compilazione dei campi che vengono mostrati dall'interfaccia grafica

## **2.5 Attività di testing**

Le attività di testing del modulo gestione prenotazioni che il piano progettato realizzerà sono:

- **testing di visualizza ordine;**
- **testing di nuova ordinazione;**
- **testing di visualizza listino;**
- **testing di inserimento prodotti;**
- **testing di valida;**

### **2.5.1. Testing di Visualizza ordine**

Questa attività di testing non richiede input quindi si testerà solamente che i dati relativi alla ordinazione effettuata dal cliente venga visualizzata correttamente.

### **2.5.2. Testing di Nuova ordinazione**

L'*Operatore* seleziona il prodotto/i e conferma la scelta.

A tale attività verranno forniti i seguenti tipi di input:

- Nuova ordinazione mediante la selezione di un prodotto con i campi Quantità e Note non compilati;
- Campo Quantità non compilato;
- Campo Note non compilato;

- Errata compilazione del campo Quantità;
- Nuova ordinazione mediante la selezione di due prodotti, con i campi Quantità e note non compilati;
- Nuova ordinazione mediante la selezione di nessun prodotto;

#### **2.5.3. Testing di Visualizzazione listino**

Questa attività di testing non richiede input quindi si testerà solamente che i dati relativi al listino dei prodotti venga visualizzata correttamente dal cliente.

#### **2.5.4. Testing di Inserimento prodotto**

L'Addetto Bar inserisce un nuovo prodotto nel listino.

A tale attività verranno forniti i seguenti tipi di input:

- Nome Prezzo corretti riferiti al prodotto da inserire;
- Nome non corretto riferito al prodotto da inserire;
- Prezzo non corretto riferito al prodotto da inserire;
- Dati inseriti per un prodotto presente nel listino;

#### **2.5.5. Testing di Valida**

L'Addetto Bar accede alla maschera di login, inserisce Login e Password e conferma la validazione.

A tale attività verranno forniti i seguenti tipi di input:

- Valori corretti per Login e Password;
- Valori corretti e scorretti (alternamente) per Login e password.

Se la Login e la password sono corretti, verrà testato che il sistema abiliti l'accesso al S.I.O. Se la Login e la Password non sono corretti, verrà testato che il sistema visualizzi un messaggio di errore e consenta la ripetizione dell'inserimento dei dati.

## 3. Test

### 3.1. Caratteristiche generali

Il testing verrà applicato sul sistema software completo e pienamente integrato, con l'obiettivo di valutare l'adesione del sistema ai requisiti specificati: si provvederà quindi alla realizzazione del *testing di sistema*.

La tecnica di testing che verrà utilizzata per testare il sistema *Quick Service* sarà il *testing funzionale*. Le caratteristiche di tale tecnica ci permetteranno di definire i casi di test e i relativi oracoli sulla base della sola conoscenza dei requisiti del sistema e dei suoi componenti specificati nel *Modello dei requisiti software*. I casi di test saranno suddivisi in classi di equivalenza utilizzando in modo da poter ridurre e semplificare il numero di test case necessari per la realizzazione del testing.

## **3.2 Descrizione dettagliata**

Effettuiamo la descrizione dettagliata delle attività di testing, mettendo in evidenza la suddivisione del dominio di ingresso di ciascun test in *classi di equivalenza*. E' opportuno osservare che ogni funzionalità verrà eseguita almeno una volta e in particolare il numero di esecuzioni dipenderà dai dati di ingresso e di uscita, da precondizioni e postcondizioni.

## **3.3 Descrizione dei casi di test**

Procediamo ora con la descrizione dei casi di test sulla base delle classi di equivalenza definite nel *Test Design Specification*.

Ovviamente verranno rispettati i seguenti vincoli :

- verranno individuati tanti casi di test da coprire tutte le classi di equivalenza valide in modo che ciascuno comprenda il maggior numero possibile di classi valide ancora scoperte;
- verranno individuati tanti casi di test da coprire tutte le classi di equivalenza non valide in modo che ciascuno copra una ed una sola delle classi non valide;
- durante la composizione dei casi di test, saranno prese in considerazione *solo le situazioni effettivamente verificabili*.

## **3.4 Testing di *Visualizza ordinazioni***

In questa attività di testing andiamo solamente a controllare che i dati visualizzati siano effettivamente quelli relativi alle ordinazioni effettuate dal cliente. Non è quindi necessario effettuare una classificazione formale delle classi di equivalenza.

### **3.5 Testing Nuova Ordinazione**

Questa funzione permette al cliente di effettuare una nuova ordinazione al bar.

Per effettuare una ordinazione, il cliente, deve scegliere uno o più prodotti, tra quelli disponibili, definendone la quantità e le eventuali note. Una volta effettuata la scelta il tutto viene confermato con l'ordinazione.

Nell'ordinazione deve essere selezionato almeno un prodotto.

È possibile modificare o cancellare l'ordinazione fatta entro un “Tot” di tempo prestabilito, utile per un eventuale ripensamento del cliente.

#### **3.5.1 Test Design Specification**

Il Cliente per effettuare una nuova ordinazione deve:

- Selezionare la voce “nuova ordinazione”.
- Scegliere almeno un prodotto dalla lista dei prodotti del bar.
- Indicare la quantità di prodotto desiderato.
- Inviare l'ordinazione.

Procediamo ora all'individuazione delle classi di equivalenza.

#### **Selezione del prodotto desiderato**

Viene selezionato con il semplice tocco dello schermo il prodotto che si desidera.

*Condizione di ingresso:* Deve essere visualizzato il listino dei prodotti disponibili (ciò è possibile invocando la funzione di “Visualizzazione Listino”)

- ✓ *Classi di equivalenza valide*

**CE1 : Almeno 1 Prodotto selezionato.**

- ✓ *Classi di equivalenza non valide*

**CE2 : Nessun Prodotto selezionato.**

### **Inserimento quantità del prodotto**

Il campo corrispondente alla quantità di prodotto desiderata è settato di default ad 1.

La quantità del prodotto viene immessa utilizzando un tastierino numerico, è quindi impossibile specificare una quantità negativa.

- ✓ *Condizione di ingresso 1:* la quantità deve essere superiore a 0 ed inferiore ad X (disponibilità del prodotto).
- ✓ *Condizione di ingresso 2:* deve essere stato selezionato un prodotto dal listino.

- ✓ *Classi di equivalenza valide*

**CE3 : ( quantità > 0 )**

**CE4 : ( quantità < X )**

**CE5 : (quantità = '')**

**CE10 : (quantità <= disponibilità)**

- ✓ *Classi di equivalenza non valide*

**CE6 : ( quantità = 0 )**

**CE7 : ( quantità > X )**

**CE11 : (quantità < disponibilità)**

## Inserimento note

Il campo note è riservato a quelle persone che desiderano avere delle piccole modifiche durante la preparazione del prodotto ordinato al bar.

Es. Il cliente desidera un caffè senza zucchero. Per fare ciò il cliente seleziona il prodotto e tra le note scrive “Senza zucchero”.

- ✓ *Condizione di ingresso:* Deve essere stato selezionato un prodotto dal listino.

- ✓ *Classi di equivalenza valide*

**CE8 : ( note ≠ "" )**

**CE9 : ( note = "" )**

### 3.5.2. Test Case Specification

<b>Test Case C01</b>	Nuova Ordinazione mediante la sola selezione di un prodotto. Non vengono compilati i campi di Quantità e Note.	<i>Data : 11 / 06 / 2008</i> <i>Versione : 0.1</i>
Caso d'Uso: <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> Caffè <b>Quantità :</b> 1 <b>Note :</b> <>Null<>	
Oracolo	<b>Viene ordinato un Caffè zuccherato.</b>	

Copertura	<i>Classi valide: CE1, CE5, CE9, CE10</i> <i>Classi non valide: Nessuna.</i>
-----------	---

Test Case C02	Nuova Ordinazione mediante la selezione di un prodotto. Non viene compilato il campo Quantità. Nelle note viene scritto : “Senza Zucchero”.	<i>Data : 11 / 06 / 2008</i> <i>Versione : 0.1</i>
Caso d’Uso: <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all’interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> Caffè <b>Quantità :</b> 1 <b>Note :</b> Senza zucchero	
Oracolo	<b>Viene ordinato un Caffè non zuccherato.</b>	

Copertura	<i>Classi valide: CE1, CE5, CE8, CE10</i> <i>Classi non valide: Nessuna.</i>
-----------	---

Test Case C03	Nuova Ordinazione mediante la selezione di un prodotto. Viene compilato il campo Quantità con 2. Il campo Note non viene compilato.	Data : 11 / 06 / 2008  Versione : 0.1
Caso d'Uso: <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> Caffè <b>Quantità :</b> 2 <b>Note :</b> <<Null>>	
Oracolo	<b>Vengono ordinati due Caffè zuccherati.</b>	

Copertura	<i>Classi valide: CE1, CE3, CE4, CE9, CE10</i> <i>Classi non valide: Nessuna.</i>
-----------	--

Test Case C04	Nuova Ordinazione mediante la selezione di un prodotto. Viene compilato il campo Quantità con 0. Il campo Note non viene compilato.	Data : 11 / 06 / 2008 Versione : 0.1
Caso d'Uso: <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> Caffè <b>Quantità :</b> 0 <b>Note :</b> <<Null>>	

Oracolo	<b>Dati inseriti errati. Quantità di prodotto inferiore a 1. Non viene effettuata l'ordinazione.</b>
Copertura	<i>Classi valide: CE1, CE9, CE10</i> <i>Classi non valide: CE6.</i>

<b>Test Case</b> <b>C05</b>	Nuova Ordinazione mediante la selezione di un prodotto. Viene compilato il campo Quantità con 100000. Il campo Note non viene compilato.	<i>Data : 11 / 06 / 2008</i>
Caso d'Uso: <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	<i>Versione : 0.1</i>
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> Caffè <b>Quantità :</b> 100000 <b>Note :</b> <<Null>>	

Oracolo	<b>Dati inseriti errati. Quantità di prodotto richiesta eccessiva. Non viene effettuata l'ordinazione.</b>
Copertura	<p><i>Classi valide: CE1, CE9, CE10.</i></p> <p><i>Classi non valide: CE7.</i></p>

<b>Test Case</b>	Nuova Ordinazione mediante la selezione di due prodotti. Non vengono compilati i campi Quantità e Note.	<i>Data : 11 / 06 / 2008</i>
<b>C06</b>		<i>Versione : 0.1</i>
<b>Caso d'Uso:</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	
<b>NUO_ORD</b>		
<b>Priorità</b>	Alta.	
<b>Set Up</b>	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
<b>Input</b>	<b>Prodotto :</b> Caffè, Cornetto <b>Quantità :</b> 1,1 <b>Note :</b> <<Null>>	

Oracolo	<b>Vengono ordinati un Caffè Zuccherato e un Cornetto non farcito.</b>
Copertura	<p><i>Classi valide: CE1, CE5, CE9,CE10.</i></p> <p><i>Classi non valide: Nessuna.</i></p>

Test Case C07	Nuova Ordinazione mediante la selezione di nessun prodotto. Non vengono compilati i campi Quantità e Note.	<i>Data : 11 / 06 / 2008</i>  <i>Versione : 0.1</i>
Caso d'Uso: <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> <<Null>> <b>Quantità :</b> <<Null>> <b>Note :</b> <<Null>>	

Oracolo	<b>Dati inseriti errati. Si deve selezionare almeno un prodotto. Non viene effettuata l'ordinazione.</b>
Copertura	<p><i>Classi valide: CE5, CE9, CE10.</i></p> <p><i>Classi non valide: CE2.</i></p>

<b>Test Case</b>	Nuova Ordinazione mediante la selezione di un prodotto. La quantità inserita è superiore alla disponibilità del bar.	<i>Data : 11 / 06 / 2008</i>
<b>Caso d'Uso:</b> <b>NUO_ORD</b>	Permette al Cliente di effettuare una nuova ordinazione presso il bar.	<i>Versione : 0.1</i>
Priorità	Alta.	
Set Up	Il prodotto è memorizzato all'interno del listino del bar.	
<b>Descrizione Test</b>		
Input	<b>Prodotto :</b> Coca Cola <b>Quantità :</b> 85 <b>Note :</b> <<Null>>	

Oracolo	<b>Ordinazione non effettuata, quantità superiore alla disponibilità.</b>
Copertura	<i>Classi valide: CE5, CE9.</i> <i>Classi non valide: CE2, CE11.</i>

### 3.6 Testing di *Visualizza listino*

In questa attività di testing andiamo solamente a controllare che i dati visualizzati siano effettivamente quelli relativi al listino dei prodotti. Non è quindi necessario effettuare una classificazione formale delle classi di equivalenza.

### 3.7 Testing di *Inserimento prodotto*

Questa funzionalità permette all'*Addetto bar* di inserire un nuovo prodotto nel listino

### 3.7.1 Test Design Specification

L'*Addetto bar* inserisce i seguenti dati: *Nome*, *Prezzo* e *Disponibilità* relativi al prodotto da inserire nel listino, mentre per quanto riguarda il *Tipo* viene utilizzata una “combo box” che visualizza le tipologie di prodotto esistenti e che quindi non richiede nessun controllo di validità in fase di testing e inoltre il *Codice*, essendo di tipo “contatore”, non deve essere inserito dall’utente visto che sarà il numero successivo a quello del codice dell’ultimo prodotto immesso nel database.

Procediamo ora all’individuazione delle classi di equivalenza.

#### Inserimento nome

- ✓ *Condizione di ingresso:* il nome deve essere una stringa non vuota

- ✓ *Classi di equivalenza valide*

**CE1: (caratteri (nome) != "")**

**CE2: (nome non presente nel database)**

- *Classi di equivalenza non valide*

**CE3: (caratteri (nome) = "")**

**CE4: (nome già presente nel database)**

#### Inserimento prezzo

- *Condizione di ingresso:* il prezzo deve essere un numero maggiore o uguale a 0.

- *Classi di equivalenza valide*

**CE5: (numero (prezzo) >= 0)**

- *Classi di equivalenza non valide*

**CE6: (numero (prezzo) < 0)**

### Inserimento disponibilità

- *Condizione di ingresso:* la disponibilità deve essere maggiore o uguale di 0 eccetto il caso in cui è *illimitata*, tale condizione si specifica inserendo “-1”.

- *Classi di equivalenza valide*

**CE7: ( (numero(disponibilità) >= 0) || (numero(disponibilità) = -1) )**

- *Classi di equivalenza non valide*

**CE8: ( (numero(disponibilità) < 0) && (numero(disponibilità) != -1) )**

### 3.7.2. Test Case Specification

<b>Test Case C01</b>	<b>Inserimento dei dati relativi a un prodotto non registrato nel database. I dati inseriti sono non nulli.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	<i>Versione : 0.1</i>
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> Aranciata <b>Prezzo:</b> 2,50 <b>Disponibilità:</b> 45	
Oracolo	<b>Il sistema consente la registrazione del prodotto nel listino.</b>	
Copertura	<i>Classi valide: CE1, CE2, CE5, CE8</i> <i>Classi non valide : Nessuna</i>	

<b>Test Case C02</b>	<b>Inserimento dei dati relativi a un prodotto non registrato nel S.I.O. Il prezzo è valido, il nome è nullo.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	<i>Versione : 0.1</i>
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> <>Null<> <b>Prezzo:</b> 2,50 <b>Disponibilità:</b> 45	
Oracolo	<b>Input non valido: Nome</b>	
Copertura	<i>Classi valide: CE2, CE5, CE7</i> <i>Classi non valide : CE3</i>	

<b>Test Case C03</b>	<p><b>Inserimento dei dati relativi a un prodotto non registrato nel S.I.O.</b>  <b>Dati inseriti non nulli, il prezzo è non valido.</b></p>	<i>Data : 11 / 06 / 2008</i>  <i>Versione : 0.1</i>
Caso d'uso:  <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> Aranciata <b>Prezzo:</b> -2,50 <b>Disponibilità:</b> 45	
Oracolo	<b>Input non valido: Prezzo.</b>	
Copertura	<i>Classi valide: CE1, CE2, CE7</i> <i>Classi non valide : CE6</i>	

<b>Test Case C04</b>	<p><b>Inserimento dei dati relativi a un prodotto il cui nome è già registrato nel S.I.O.</b></p> <p><b>Dati inseriti non nulli.</b></p>	<i>Data : 11 / 06 / 2008</i> <i>Versione : 0.1</i>
Caso d'uso: <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> Coca Cola <b>Prezzo:</b> 3,00 <b>Disponibilità:</b> 45	
Oracolo	<b>Nome già presente nel database Il sistema non consente la memorizzazione dei dati.</b>	
Copertura	<i>Classi valide: CE1, CE5, CE7</i> <i>Classi non valide : CE4</i>	

<b>Test Case C05</b>	<b>Inserimento dei dati relativi a un prodotto, disponibilità illimitata. Dati inseriti non nulli.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> Aranciata <b>Prezzo:</b> 3,00 <b>Disponibilità:</b> -1	
Oracolo	<b>Il sistema consente la memorizzazione dei dati.</b>	
Copertura	<i>Classi valide: CE1, CE5, CE7 Classi non valide : nessuna</i>	

<b>Test Case C06</b>	<b>Inserimento dei dati relativi a un prodotto, disponibilità minore di 0. Dati inseriti non nulli.</b>	<i>Data : 11 / 06 / 2008</i>
		<i>Versione : 0.1</i>
Caso d'uso: <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> Aranciata <b>Prezzo:</b> 3,00 <b>Disponibilità:</b> -1234	
Oracolo	<b>Il sistema non consente la memorizzazione dei dati.</b> <b>Disponibilità non valida.</b>	
Copertura	<i>Classi valide: CE1, CE5</i> <i>Classi non valide : CE8</i>	

<b>Test Case C07</b>	<b>Inserimento dei dati relativi a un prodotto.</b> <b>Dati inseriti non nulli tranne disponibilità.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>INS_PRO</b>	Si occupa di inserire un nuovo prodotto nel listino.	
Priorità	Alta	
Set UP	Il prodotto non deve essere già memorizzato all'interno del listino bar.	
<b>Descrizione Test</b>		
Input	<b>Nome:</b> Aranciata <b>Prezzo:</b> 3,00 <b>Disponibilità:</b> <<Null>>	
Oracolo	<b>Il sistema consente la memorizzazione dei dati.</b>	
Copertura	<i>Classi valide: CE1, CE5, CE7</i> <i>Classi non valide : CE8</i>	

## 3.8 Testing di *valida*

Questa funzionalità permette all'*Addetto bar* di compiere operazioni riservate, con l'accesso al sistema tramite Login e Password.

### 3.8.1 *Test Design Specification*

L'*Addetto bar* inserisce login e password nell'apposita sezione della maschera di login.

Procediamo ora all'individuazione delle classi di equivalenza.

#### Inserimento login

- ✓ *Condizione di ingresso 1:* la login è una stringa alfanumerica composta da 6 caratteri.
- ✓ *Condizione di ingresso 2:* la login deve essere presente all'interno del S.I.O.
- *Classi di equivalenza valide*
  - CE1 :** ( (#caratteri( login ) = 5 )
  - CE2 :** ( login presente nel S.I.O. )
- *Classi di equivalenza non valide*
  - CE3 :** ( (#caratteri( login ) < 5 )
  - CE4 :** ( (#caratteri( login ) > 5 )
  - CE5 :** ( login non presente nel S.I.O. )

#### Inserimento password

- ✓ *Condizione di ingresso 1:* la password è una stringa alfanumerica composta da 6 caratteri.
- ✓ *Condizione di ingresso 2:* la password deve essere presente all'interno del S.I.O.

- *Classi di equivalenza valide*

**CE6 : ( #caratteri( password ) = 5 )**

**CE7 : ( password presente nel S.I.O. )**

- *Classi di equivalenza non valide*

**CE8 : ( #caratteri( password ) < 5 )**

**CE9 : ( #caratteri( password ) > 5 )**

**CE10 : ( password non presente nel S.I.O. )**

### 3.8.2 Test Case Specification

<b>Test Case C01</b>	<b>Valida addetto bar mediante login corretta, password corretta e la coppia login, password presente nel S.I.O.</b>	<i>Data : 11 / 06 / 2008</i> <i>Versione : 0.1</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La coppia (login, password) = (admin, admin) è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> admin <b>Password:</b> admin	
Oracolo	<b>Il sistema consente l'accesso all'addetto bar</b>	
Copertura	<i>Classi valide: CE1, CE2, CE6, CE7</i> <i>Classi non valide : Nessuna</i>	

<b>Test Case C02</b>	<b>Valida addetto bar mediante login non corretta (numero inferiore di caratteri), e password registrata nel S.I.O.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La password “admin” è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> adm <b>Password:</b> admin	
Oracolo	<b>Input non valido: login</b>	
Copertura	<i>Classi valide: CE6, CE7</i> <i>Classi non valide :CE3, CE5</i>	

<b>Test Case C03</b>	<b>Valida addetto bar mediante login non corretta (numero superiore di caratteri), e password registrata nel S.I.O.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La password <b>admin</b> è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> vitolosalvatore <b>Password:</b> admin	
Oracolo	<b>Input non valido: login</b>	
Copertura	<i>Classi valide: CE6, CE7</i> <i>Classi non valide :CE4, CE5</i>	

<b>Test Case C04</b>	<b>Valida addetto bar mediante login registrate nel S.I.O. e password non corretta (numero inferiore di caratteri).</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La login <b>admin</b> è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> admin <b>Password:</b> sal	
Oracolo	<b>Input non valido: password</b>	
Copertura	<i>Classi valide: CE1, CE2</i> <i>Classi non valide :CE8, CE10</i>	

<b>Test Case C05</b>	<b>Valida addetto bar mediante login registrate nel S.I.O. e password non corretta (numero superiore di caratteri).</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La login <b>admin</b> è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> admin <b>Password:</b> salvatore1983	
Oracolo	<b>Input non valido: password</b>	
Copertura	<i>Classi valide: CE1, CE2</i> <i>Classi non valide :CE9, CE10</i>	

<b>Test Case C06</b>	<b>Valida addetto bar mediante login registrata nel S.I.O. e password corretta ma non registrata nel S.I.O.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La login <b>admin</b> è registrata nel S.I.O. La password <b>lello</b> non è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> admin <b>Password:</b> lello	
Oracolo	<b>Il sistema non consente l'accesoo all'addetto bar</b>	
Copertura	<i>Classi valide: CE1, CE2, CE6 Classi non valide :CE10</i>	

<b>Test Case C07</b>	<b>Valida addetto bar mediante login corretta ma non registrata nel S.I.O. e password corretta registrata nel S.I.O.</b>	<i>Data : 11 / 06 / 2008</i>
Caso d'uso: <b>VALIDA</b>	Si occupa di eseguire le funzioni necessarie per l'autenticazione dell'addetto bar	
Priorità	Alta	
Set UP	La login <b>lello</b> non è registrata nel S.I.O. La password <b>admin</b> è registrata nel S.I.O.	
<b>Descrizione Test</b>		
Input	<b>Login:</b> lello <b>Password:</b> admin	
Oracolo	<b>Il sistema non consente l'accesso all'addetto bar</b>	
Copertura	<i>Classi valide: CE1, CE6, CE7 Classi non valide :CE5</i>	

### 3.9 Test degli oggetti remoti

Ogni test è composto da test case individuali. Ogni test case è una classe che estende TestCase.

Per convenzione la classe di test ha lo stesso nome della classe da testare cui viene aggiunto il prefisso “Test” (ad es. HelloWorld.java diventa TestHelloWorld.java) e si trova all'interno dello stesso package della classe cui viene aggiunto il suffisso “test” (ad es. se il package della classe da testare è hello il package della classe di test sarà hello.test). Analogamente i metodi della classe di test hanno lo stesso nome dei metodi della classe da testare ma iniziano con il prefisso “test”.

La verifica dei test case è effettuata mediante le condizioni di assert (ad esempio: assertEquals, assertTrue, assertNotNull).

Supponiamo di voler scrivere un test case per la seguente classe:

```
package hello;

class HelloWorld {

    public String sayHello(){
        return "Hello World";
    }

    public static void main(String[] args){
        HelloWorld world=new HelloWorld();
        System.out.println(world.sayHello());
    }
}
```

In base a quanto detto creiamo la nuova classe TestHelloWorld contenente un metodo per ogni metodo della classe da testare (nel nostro caso il solo metodo testSayHello). Il risultato è il seguente:

```
package hello.test;
```

```

import hello.HelloWorld;
import junit.framework.TestCase;
import junit.framework.AssertionFailedError;

public class TestHelloWorld extends TestCase{

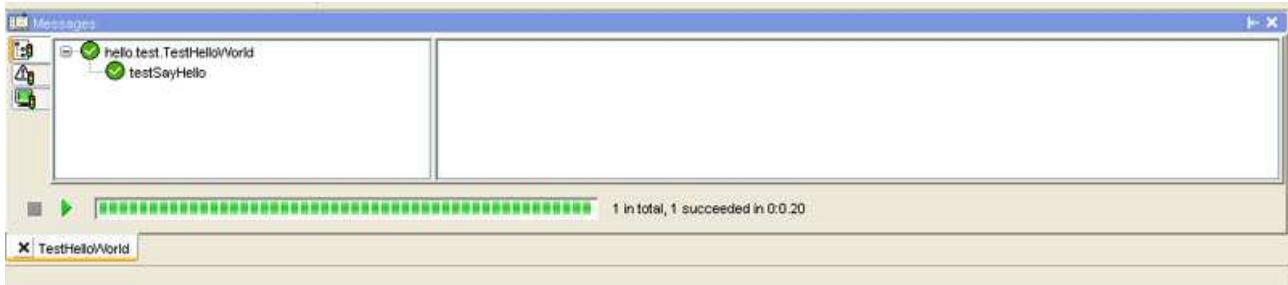
    public TestHelloWorld(String name){
        super(name);
    }

    public static void main(String[] args){
        junit.textui.TestRunner.run(TestHelloWorld.class);
    }

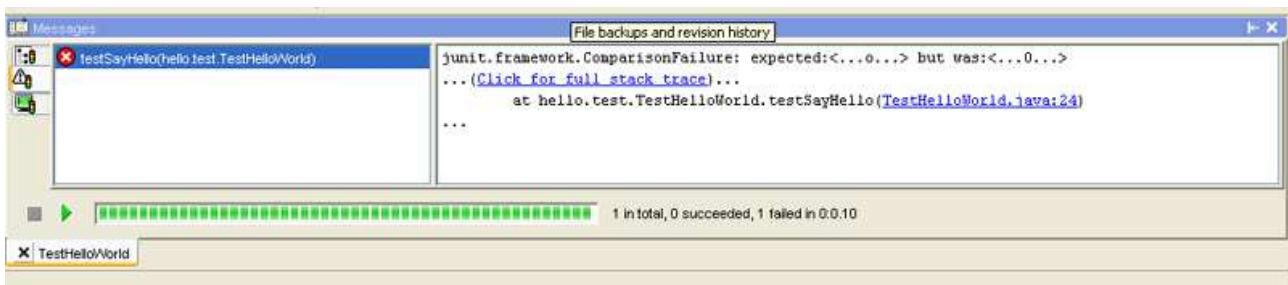
    public void testSayHello(){
        HelloWorld world = new HelloWorld();
        assertEquals("Hello World",world.sayHello());
    }
}

```

Dal momento che il framework JUnit è integrato nell'ambiente di sviluppo da noi utilizzato, è possibile controllare l'esecuzione dei test direttamente da JBuilder. La seguente figura mostra l'esito positivo dell'esecuzione del test per la classe dell'esempio precedente:



Supponiamo ora di modificare la stringa restituita dal metodo sayHello della classe da testare in "Hello WOrld". Il testcase fallirà e la seguente figura mostra l'esito dell'esecuzione:



## 4. Test log

### 4.1 Descrizione del test log

In questa sezione descriviamo i risultati dell'esecuzione dei casi di test, riportando per ognuno la descrizione delle seguenti caratteristiche :

- \_ ***Stato iniziale***: situazione del sistema immediatamente prima dell'esecuzione del test;
- \_ ***Scenario***: descrizione dettagliata dei passi di esecuzione. Per ogni passo di esecuzione viene registrato il completamento o meno dell'operazione effettuata, e in caso di anomalie del sistema l'ID dell'errore che si è presentato ;
- \_ ***Stato finale***: situazione del sistema al termine dell'esecuzione del test ;
- \_ ***Conclusioni***: descrizione dell'esito del test. Viene stabilito se il test ha avuto successo o meno, e se il completamento è rimasto in sospeso ;

## 4.2 Test log Nuova Ordinazione

<b>Test Case C01</b>	Nuova Ordinazione mediante la sola selezione di un prodotto. Non vengono compilati i campi di Quantità e Note.	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.	<b>Versione : 0.1</b>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario principale</i>  CASO D'USO: NUO_ORD	Scelta prodotto: <b>Caffè</b>	<b>SI</b>	
	Scelta quantità: <b>1</b>	<b>SI</b>	
	Inserimento note: <>Null<>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema informa che l'ordinazione è avvenuta con successo.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b>	Nuova Ordinazione mediante la selezione di un prodotto. Non viene compilato il campo Quantità. Nelle note viene scritto : “Senza Zucchero”.	<i>Data : 13 / 07 / 2008</i>	
<b>C02</b>		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>	Scelta prodotto: <b>Caffè</b>	<b>SI</b>	
<b>CASO D'USO: NUO_ORD</b>	Scelta quantità: <b>1</b>	<b>SI</b>	
	Inserimento note: <b>Senza zucchero</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema informa che l'ordinazione è avvenuta con successo.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b>  C03	Nuova Ordinazione mediante la selezione di un prodotto. Viene compilato il campo Quantità con 2. Il campo Note non viene compilato.	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>	Scelta prodotto: <b>Caffè</b>	<b>SI</b>	
CASO D'USO: NUO_ORD	Scelta quantità: <b>2</b>	<b>SI</b>	
	Inserimento note: <>Null>>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema informa che l'ordinazione è avvenuta con successo.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b>  <b>C04</b>	Nuova Ordinazione mediante la selezione di un prodotto. Viene compilato il campo Quantità con 0. Il campo Note non viene compilato.	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>	Scelta prodotto: <b>Caffè</b>	<b>SI</b>	
CASO D'USO: <b>NUO_ORD</b>	Scelta quantità: <b>0</b>	<b>SI</b>	
	Inserimento note: <<Null>>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non consente di scegliere quantità “0” poichè il tasto “0” è disabilitato.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b> <b>C05</b>	Nuova Ordinazione mediante la selezione di un prodotto. Viene compilato il campo Quantità con 100000. Il campo Note non viene compilato.	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>	Scelta prodotto: <b>Caffè</b>	<b>SI</b>	
CASO D'USO: <b>NUO_ORD</b>	Scelta quantità: <b>100000</b>	<b>SI</b>	
	Inserimento note: <>Null>>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non consente di scegliere quantità superiori a 99.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b>  <b>C06</b>	Nuova Ordinazione mediante la selezione di due prodotti. Non vengono compilati i campi Quantità e Note.	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<b>Scenario alternativo</b>  CASO D'USO: NUO_ORD	Scelta prodotto: <b>Caffè</b> , <b>Cornetto</b>	<b>SI</b>	
	Scelta quantità: <b>1,1</b>	<b>SI</b>	
	Inserimento note: <<Null>>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema informa che l'ordinazione è avvenuta con successo.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b>  C07	Nuova Ordinazione mediante la selezione di nessun prodotto. Non vengono compilati i campi Quantità e Note.	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>	Scelta prodotto: <<Null>>	SI	
CASO D'USO: NUO_ORD	Scelta quantità: <<Null>>	SI	
	Inserimento note: <<Null>>	SI	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema informa che è impossibile inviare un'ordinazione nulla.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case</b>	Nuova Ordinazione mediante la selezione di un prodotto. La quantità inserita è superiore alla disponibilità del bar.	<i>Data : 13 / 07 / 2008</i>	
<b>C08</b>		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	Il prodotto è memorizzato all'interno del listino del bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>	Scelta prodotto: <b>Coca Cola</b>	<b>SI</b>	
<b>CASO D'USO: NUO_ORD</b>	Scelta quantità: <b>85</b>	<b>SI</b>	
	Inserimento note: <>Null<>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema informa che è impossibile inviare l'ordinazione, perché la quantità richiesta è superiore alla disponibilità del prodotto.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

### 4.3 Test log Inserimento Prodotto

<b>Test Case C01</b>	Inserimento dei dati relativi a un prodotto non registrato nel database. I dati inseriti sono non nulli.	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.	<i>Versione : 0.1</i>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario principale</i> CASO D'USO: INS_PRO	Inserimento nome: <b>Aranciata</b>	<b>SI</b>	
	Inserimento prezzo: <b>2,50</b>	<b>SI</b>	
	Inserimento disponibilità: <b>45</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il prodotto è stato inserito nel database.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C02</b>	<b>Inserimento dei dati relativi a un prodotto non registrato nel S.I.O. Il prezzo e la disponibilità sono validi, il nome è nullo.</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i> CASO D'USO: INS_PRO	Inserimento nome: <>Null>>	SI	
	Inserimento prezzo: 2,50	SI	
	Inserimento disponibilità: 45	SI	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il prodotto non viene inserito (nome nullo).</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C03</b>	<b>Inserimento dei dati relativi a un prodotto non registrato nel S.I.O. Dati inseriti non nulli, il prezzo è non valido.</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.	<i>Versione : 0.1</i>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i> CASO D'USO: INS_PRO	Inserimento nome: <b>Aranciata</b>	<b>SI</b>	
	Inserimento prezzo: <b>-2,50</b>	<b>SI</b>	
	Inserimento disponibilità: <b>45</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non permette l'inserimento (prezzo non valido).</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C04</b>	<b>Inserimento dei dati relativi a un prodotto il cui nome è già registrato nel S.I.O.</b> <b>Dati inseriti non nulli.</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.	<i>Versione : 0.1</i>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i> CASO D'USO: INS_PRO	Inserimento nome: <b>Coca</b> <b>Cola</b>	<b>SI</b>	
	Inserimento prezzo: <b>3,00</b>	<b>SI</b>	
	Inserimento disponibilità: <b>45</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non permette l'inserimento (prodotto già presente).</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C05</b>	Inserimento dei dati relativi a un prodotto non registrato nel S.I.O. Dati inseriti non nulli, la disponibilità è illimitata.	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i> CASO D'USO: INS_PRO	Inserimento nome: <b>Aranciata</b>	<b>SI</b>	
	Inserimento prezzo: <b>2,50</b>	<b>SI</b>	
	Inserimento disponibilità: <b>-1</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema permette l'inserimento (disponibilità illimitata).</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C06</b>	<b>Inserimento dei dati relativi a un prodotto il cui nome non registrato nel S.I.O. Dati inseriti non nulli, disponibilità non valida.</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.	<i>Versione : 0.1</i>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>  CASO D'USO: INS_PRO	Inserimento nome: <b>Aranciata</b>	<b>SI</b>	
	Inserimento prezzo: <b>2,50</b>	<b>SI</b>	
	Inserimento disponibilità: <b>-1234</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non permette l'inserimento (disponibilità negativa).</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C07</b>	Inserimento dei dati relativi a un prodotto il cui nome non è registrato nel S.I.O. Dati inseriti non nulli, tranne disponibilità.	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	Il prodotto non deve essere già memorizzato all'interno del listino bar.	<i>Versione : 0.1</i>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>  CASO D'USO: INS_PRO	Inserimento nome: <b>Aranciata</b>	<b>SI</b>	
	Inserimento prezzo: <b>2,50</b>	<b>SI</b>	
	Inserimento disponibilità: <b>&lt;&lt;Null&gt;&gt;</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema permette l'inserimento con disponibilità 0.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

#### 4.4 Test log Valida

<b>Test Case C01</b>	<b>Valida addetto bar mediante login corretta, password corretta e la coppia login, password presente nel S.I.O.</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	La coppia (login, password) = (admin, admin) è registrata nel S.I.O.	<i>Versione : 0.1</i>	
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario principale</i> <b>CASO D'USO: LOG_PWD</b>	Inserimento nel campo login: <b>admin</b>	<b>SI</b>	
	Inserimento nel campo password: <b>admin</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema ha consentito l'accesso.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C02</b>	<b>Valida addetto bar mediante login non corretta (numero inferiore di caratteri), e password registrata nel S.I.O.</b>	<i>Data : 13 / 07 / 2008</i>		
<b>Stato iniziale</b>	La password <b>adm</b> non è registrata nel S.I.O.			
<b>Descrizione esecuzione</b>				
Scenario	Descrizione passi	Positivo	IDErr	
<i>Scenario alternativo</i> <b>CASO D'USO: LOG_PWD</b>	Inserimento nel campo login: <b>adm</b>	<b>SI</b>		
	Inserimento nel campo password: <b>admin</b>	<b>SI</b>		
<b>Rapporto dettagliato dei risultati</b>				
<b>Stato finale</b>	<b>Il sistema non ha consentito l'accesso perché il numero di caratteri della password è inferiore a 5.</b>			
<b>Pending</b>	NO			
<b>Passed</b>	SI			

<b>Test Case C03</b>	<b>Valida addetto bar mediante login non corretta (numero superiore di caratteri), e password registrata nel S.I.O.</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	La password <b>admin</b> è registrata nel S.I.O.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>  CASO D'USO: <b>LOG_PWD</b>	Inserimento nel campo login: <b>vitolosalvatore</b>	<b>SI</b>	
	Inserimento nel campo password: <b>admin</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non ha consentito l'accesso perché il numero di caratteri della login è superiore a 5.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C04</b>	<b>Valida addetto bar mediante login registrate nel S.I.O. e password non corretta (numero inferiore di caratteri).</b>	<i>Data : 13 / 07 / 2008</i>		
<b>Stato iniziale</b>	La login <b>admin</b> è registrata nel S.I.O.			
<b>Descrizione esecuzione</b>				
	<b>Scenario</b>	<b>Descrizione passi</b>	<b>Positivo</b>	<b>IDErr</b>
<i>Scenario alternativo</i> <b>CASO D'USO:LOG_PWD</b>	Inserimento nel campo login: <b>admin</b>		<b>SI</b>	
	Inserimento nel campo password: <b>sal</b>		<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>				
<b>Stato finale</b>	<b>Il sistema non ha consentito l'accesso perché il numero di caratteri della password è inferiore a 5.</b>			
<b>Pending</b>	NO			
<b>Passed</b>	SI			

<b>Test Case C05</b>	<b>Valida addetto bar mediante login registrate nel S.I.O. e password non corretta (numero superiore di caratteri).</b>	<i>Data : 13 / 07 / 2008</i>	
<b>Stato iniziale</b>	La login <b>admin</b> è registrata nel S.I.O.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i> <b>CASO D'USO:LOG_PWD</b>	Inserimento nel campo login: <b>admin</b>	<b>SI</b>	
	Inserimento nel campo password: <b>salvatore1983</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non ha consentito l'accesso perché il numero di caratteri della password è superiore a 5.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C06</b>	<b>Valida addetto bar mediante login registrata nel S.I.O. e password corretta ma non registrata nel S.I.O.</b>	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	La login <b>admin</b> è registrata nel S.I.O.  La password <b>lello</b> non è registrata nel S.I.O.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>  CASO D'USO: <b>LOG_PWD</b>	Inserimento nel campo login: <b>admin</b>	<b>SI</b>	
	Inserimento nel campo password: <b>lello</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non ha consentito l'accesso perché la password è errata.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

<b>Test Case C07</b>	<b>Valida addetto bar mediante login corretta ma non registrata nel S.I.O. e password corretta registrata nel S.I.O.</b>	<i>Data : 13 / 07 / 2008</i>	
		<i>Versione : 0.1</i>	
<b>Stato iniziale</b>	La login <b>lello</b> non è registrata nel S.I.O. La password <b>admin</b> è registrata nel S.I.O.		
<b>Descrizione esecuzione</b>			
Scenario	Descrizione passi	Positivo	IDErr
<i>Scenario alternativo</i>  CASO D'USO: <b>LOG_PWD</b>	Inserimento nel campo login: <b>lello</b>	<b>SI</b>	
	Inserimento nel campo password: <b>admin</b>	<b>SI</b>	
<b>Rapporto dettagliato dei risultati</b>			
<b>Stato finale</b>	<b>Il sistema non ha consentito l'accesso perché la login è errata.</b>		
<b>Pending</b>	NO		
<b>Passed</b>	SI		

## 5. Descrizione classi di testo

### 5.1 Test della collection

```
package Qsclassicomuni.test;

import Qsclassicomuni.adapters.*;
import Qsclassicomuni.collezioni.*;
import junit.framework.*;

/**
 * <p>Title: TestQsCollection</p>
 * <p>Description: Testa le collection usate</p>
 * <p>Copyright: Copyright (c) 2008</p>
 * <p>Company: UNISA (Corso ingegneria del Software)</p>
 * @author Gruppo De Sio
 * @version 1.0
 */

public class TestQsCollection
    extends TestCase {
    private QsCollection qsCollection;

    /**
     * Inizializza le classi da testare
     * @throws Exception
     */

    protected void setUp() throws Exception {
        super.setUp();
        qsCollection = new QsCollectionAdapter();
        qsCollection.aggiungi( "VOCE 0" );
        qsCollection.aggiungi( "VOCE 1" );
    }

    /**
     * Il metodo testa l'aggiunta di un elemento alla collection
     */
    public void testAggiungi() {
        int i;
        for (i = 2; i < 10; i++) {
            qsCollection.aggiungi( "VOCE " + i );
        }
        System.out.println(qsCollection.dimensione());
        /*@todo fill in the test code*/
    }

    /**
     * Il metodo testa il prelievo di un elemento da una collection.
     * Il valore della variabile oracolo deve essere lo stesso di output.
    
```

```

*/
public void testPreleva() {
    int i = 0;
    String oracolo = "VOCE 0";
    String output = (String) qsCollection.preleva(i);
    assertEquals("return value", oracolo, output);
    /**@todo fill in the test code*/
}

/**
* Il metodo testa la modifica di un elemento in una collection.
* Il test avrà successo se la variabile oracolo avrà lo stesso valore del
* risultato di qsCollection.preleva(0).
*/
public void testModifica() {
    int i = 0;
    String oracolo = "VOCE 00";
    qsCollection.modifica(0, oracolo);
    assertEquals("Modified value", oracolo,
                (String) qsCollection.preleva(0));
}

/**
* Il metodo testa la richiesta della dimensione della collection.
* Il test avrà successo se la variabile oracolo avrà lo stesso valore di output.
*/
public void testDimensione() {
    int oracolo = 2;
    int output = qsCollection.dimensione();
    System.out.println(output);
    assertEquals("return value", oracolo, output);
    /**@todo fill in the test code*/
}

/**
* Il metodo testa la rimozione di un elemento da una collection.
*/
public void testRimuovi() {
    int oldSize = qsCollection.dimensione();
    qsCollection.rimuovi(0);
    assertTrue("new size", qsCollection.dimensione() < oldSize);

}
/**
* Il metodo svuota le collection e libera la memoria.
* @throws Exception
*/
protected void tearDown() throws Exception {
    qsCollection = null;
    super.tearDown();
}
}

```

## 5.2 Test Sessione Bar Gestore

```
package QsServer.test;

import java.rmi.*;
import java.util.*;

import QsServer.*;
import QsServer.AddettoBar.*;
import QsServer.DataLayer.*;
import Qsclassicomuni.collezioni.*;
import com.borland.dx.sql.dataset.*;
import junit.framework.*;

/**
 * <p>Title: TestSessioneBarGestoreImp</p>
 * <p>Description: Testa la sessione per la gestione del bar</p>
 * <p>Copyright: Copyright (c) 2008</p>
 * <p>Company: UNISA (Corso ingegneria del Software)</p>
 * @author Gruppo De Sio
 * @version 1.0
 */

public class TestSessioneBarGestoreImp
    extends TestCase {
    private SessioneBarGestoreImp sessioneBarGestoreImp = null;
    private Database database;
    private ResourceBundle sqlRes;

    /**
     * Inizializza le classi da testare
     * @throws Exception
     */

    protected void setUp() throws Exception {
        super.setUp();
        /**@todo verify the constructors*/
        database = new Database();
        database.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor(
            "jdbc:odbc:QuickService", "", "", false,
"sun.jdbc.odbc.JdbcOdbcDriver"));
        sqlRes = ResourceBundle.getBundle("QsServer.SqlRes");
        sessioneBarGestoreImp = new SessioneBarGestoreImp(database,
sqlRes);
    }

    /**
     * Il metodo svuota le collection e libera la memoria.
     * @throws Exception
     */
}
```

```

protected void tearDown() throws Exception {
    sessioneBarGestoreImp = null;
    database.closeConnection();
    database = null;
    super.tearDown();
}

/**
 * Il metodo testa la creazione di una nuova sessione per il gestore del bar.
 * @throws RemoteException
 */

public void testSessioneBarGestoreImp() throws RemoteException {
    Database db = database;
    ResourceBundle sqlR = sqlRes;
    sessioneBarGestoreImp = new SessioneBarGestoreImp(database,
sqlRes);
    /**@todo fill in the test code*/
}

/**
 * Il metodo testa il caricamento categorie.
 * Se la dimensione è maggiore di 0 allora esistono categorie.
 * @throws RemoteException
 */
public void testCaricaCategorieDimensione() throws RemoteException
{
    QsCollection output = sessioneBarGestoreImp.caricaCategorie();
    assertTrue("ci sono categorie", output.dimensione() > 0);
    /**@todo fill in the test code*/
}

/**
 * Testa che il metodo restituisca correttamente i dati dell'utente.
 * Il test avrà successo se la variabile oracolo avrà lo stesso valore di output.
 */
public void testGetDatiUtente() {
    Utente oracolo = null;
    Utente output = sessioneBarGestoreImp.getDatiUtente();
    assertEquals("return value", oracolo, output);
    /**@todo fill in the test code*/
}

/**
 * Il metodo testa l'inserimento di un nuovo prodotto nel database.
 * @throws RemoteException
 */

```

```

public void testInserisciProdotto() throws RemoteException {
    QueryDataSet qds = new QueryDataSet();
    qds.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database,
        sqlRes.getString("Prodotti"), null, true,
        Load.ALL));
    qds.open();
    int size = qds.rowCount();
    int oracolo = size + 1;
    ProdottoBar p = new ProdottoBar(1, "Prova", 5.00, 0);
    sessioneBarGestoreImp.inserisciProdotto(p);
    qds.refresh();
    assertEquals(oracolo, qds.rowCount());
    assertTrue(qds.getString("Descrizione") != null);
    qds.close();
    qds = null;
    /**@todo fill in the test code*/
}

/**
 * Il metodo testa il caso in cui nella valida nome utente e password siano vuoti.
 * Il test avrà successo se la variabile oracolo avrà lo stesso valore di output.
 * @throws validaException
 */
public void testValidaNoUIDPWD() throws validaException {
    String UID = "";
    String PWD = "";
    boolean oracolo = false;
    boolean output;
    try {
        output = sessioneBarGestoreImp.valida(UID, PWD);
    }
    catch (validaException ex) {
        output = false;
    }
    assertEquals("return value", oracolo, output);
    /**@todo fill in the test code*/
}

/**
 * Il metodo testa la valida utente.
 * @throws validaException
 */

public void testValidaUIDPWD() throws validaException {
    String UID = "admin";
    String PWD = "admin";
    boolean oracolo = true;
    boolean output;
    try {
        output = sessioneBarGestoreImp.valida(UID, PWD);
    }
}

```

```
        catch (validaException ex) {
            output = false;
        }
        assertEquals("return value", oracolo, output);
        /**@todo fill in the test code*/
    }
}
```

### 5.3 Test Sessione Cliente

```
package QsServer.test;

import java.rmi.*;
import java.util.*;

import QsServer.Cliente.*;
import QsServer.DataLayer.*;
import Qsclassicomuni.adapters.*;
import Qsclassicomuni.collezioni.*;
import com.borland.dx.sql.dataset.*;
import junit.framework.*;

/**
 * <p>Title: TestSessioneClienteImp</p>
 * <p>Description: Tesa la sessione del cliente</p>
 * <p>Copyright: Copyright (c) 2008</p>
 * <p>Company: UNISA (Corso ingegneria del Software)</p>
 * @author Gruppo De Sio
 * @version 1.0
 */

public class TestSessioneClienteImp
    extends TestCase {
    Database database;
    ResourceBundle sqlRes;
    Camera camera;
    private SessioneClienteImp sessioneClienteImp = null;
    private double tot;

    /**
     * Inizializza le classi da testare
     * @throws Exception
     */

    protected void setUp() throws Exception {
        super.setUp();
        /*@todo verify the constructors*/
        database = new Database();
        camera = new Camera(2, 0, true, false, 5, 5);
        database.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor(
            "jdbc:odbc:QuickService", "", "", false,
"sun.jdbc.odbc.JdbcOdbcDriver"));
        sqlRes = ResourceBundle.getBundle("QsServer.SqlRes");
        sessioneClienteImp = new SessioneClienteImp(database, sqlRes,
camera);
    }
}
```

```

/**
 * Il metodo svuota le collection e libera la memoria.
 * @throws Exception
 */

protected void tearDown() throws Exception {
    sessioneClienteImp = null;
    super.tearDown();
}

/**
 * Il metodo testa la creazione di una nuova sessione cliente.
 * @throws RemoteException
 */

public void testSessioneClienteImp() throws RemoteException {
    sessioneClienteImp = new SessioneClienteImp(database, sqlRes,
camera);
    /**@todo fill in the test code*/
}

/**
 * Testa che il metodo restituica i dati di una camera correttamente.
 */
public void testGetDatiCamera() {
    Camera actualReturn = sessioneClienteImp.getDatiCamera();
    assertTrue("is not null", actualReturn != null);
    /**@todo fill in the test code*/
}

/**
 * Il metodo testa l'inserimento di nuove ordinazioni.
 * @throws DisponibilitaException
 */
public void testOrdinazioneBar() throws DisponibilitaException {
    int room = 2;
    QsCollection collect = new QsCollectionAdapter();
    QueryDataSet qds = new QueryDataSet();
    qds.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database,
        sqlRes.getString("Prodotti") + " WHERE disponibilita >0 OR
disponibilita =-1", null, true,
        Load.ALL));
    qds.open();

    for (int i = 0; i < 10 && qds.next(); i++) {
        collect.aggiungi(new DettaglioPrenotazioneBar(qds.getInt(0),
qds.getString(2),1,
                qds.getDouble(3), ""));
    }

    double tot = +qds.getDouble(3);
}

```

```

        qds.next();
    }
    qds.close();
    sessioneClienteImp.ordinazioneBar(room, collect, tot);
    /**@todo fill in the test code*/
}

/***
 * Il metodo testa il caso in cui l'inserimento di nuove prenotazioni fallisce.
 * @throws DisponibilitaException
 */
public void testOrdinazioneBarFallita() throws
DisponibilitaException {
    int room = 2;
    boolean oracolo = true;
    QsCollection collect = new QsCollectionAdapter();
    QueryDataSet qds = new QueryDataSet();
    qds.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database,
    sqlRes.getString("Prodotti") + " WHERE disponibilita <0",
null, true,
    Load.ALL));
    qds.open();

    for (int i = 0; i < 10 && qds.next(); i++) {
        collect.aggiungi(new DettaglioPrenotazioneBar(qds.getInt(0),
qds.getString(2), 1,
                qds.getDouble(3), ""));
    }

    double tot = +qds.getDouble(3);
    qds.next();
}
qds.close();
try {
    sessioneClienteImp.ordinazioneBar(room, collect, tot);
}
catch (DisponibilitaException ex){
    oracolo = false;
    assertFalse(oracolo);
}
/**@todo fill in the test code*/
}
/***
 * IL metodo testa la visualizzazione delle categorie.
 * Se la dimensione è maggiore di 0 allora vengono visualizzate le categorie.
 * @throws RemoteException
 */
public void testVisualizzaListinoCategorie() throws
RemoteException {
    int codCat = -1;
    QsCollection actualReturn =
sessioneClienteImp.visualizzaListino(codCat);

```

```
    assertTrue("the size is >0", actualReturn.dimensione() > 0);
    /**@todo fill in the test code*/
}

/***
 * Il metodo testa la visualizzazione dei prodotti.
 * Se la dimensione è maggiore di 0 allora vengono visualizzati i prodotti.
 * @throws RemoteException
 */
public void testVisualizzaListinoProdotti() throws RemoteException
{
    int codCat = 1; // categoria presente
    QsCollection actualReturn =
sessioneClienteImp.visualizzaListino(codCat);
    assertTrue("the size is >0", actualReturn.dimensione() > 0);
    /**@todo fill in the test code*/
}

}
```