POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Gestionale

Elaborato finale

Gestione ed ottimizzazione del processo di prenotazione alberghiera

Relatore:
Prof. Fulvio Corno

Candidato:
Fanny Acquafresca

Dicembre 2013

Allegato 1
Struttura della proposta di prova finale

Studente proponente (matricola, cognome, nome)	188862 Acquafresca Fanny
Titolo della proposta	Software di gestione alberghiera
Descrizione del problema proposto	Sviluppo di un software che gestisca la prenotazione di camere da parte del personale alberghiero. In particolare il software, tramite un'interfaccia grafica appositamente sviluppata, inserito il periodo desiderato visualizza la disponibilità delle camere ed in base alle esigenze del cliente, effettua la prenotazione della/e camera/e modificandone la disponibilità.
Descrizione della rilevanza gestionale del problema	Riduzione e ottimizzazione dei tempi organizzativi delle prenotazioni alberghiere.
Descrizione dei data-set per la valutazione	Data-set di camere e prenotazioni.
Descrizione preliminare degli algoritmi coinvolti	Algoritmi di ricerca, ottimizzazione.
Descrizione preliminare delle funzionalità previste per l'applicazione software	Inserimento prenotazioni, visualizzazione della disponibilità nel periodo richiesto andamento degli introiti mensili nell'anno corrente
Dati di invio della proposta	05/06/2013

Capitolo 1

Gestione ed ottimizzazione del processo di prenotazione alberghiera

1.1 Descrizione del complesso alberghiero

La struttura considerata, sita in provincia di Siena, è costituita di 13 appartamenti, tra cui due suite, in grado di ospitare complessivamente un massimo di 70 persone. La struttura, strettamente a conduzione familiare, per un periodo di tempo è stata luogo di esperienza lavorativa personale, durante la quale si è sentita la necessità di sviluppare un software, che ottimizzasse la capacità di gestione delle prenotazioni, tutt'ora svolte manualmente dai proprietari.

1.2 Descrizione del software

Il software è stato pensato in modo da avere uno strumento di controllo tra la prenotazione da inserire e la disponibilità delle camere; se l'esito è positivo, ossia nel periodo richiesto vi sono appartamenti liberi, il programma stampa a video il nome dell'appartamento con il relativo numero di posti. A questo punto l'utente può procedere all'inserimento della prenotazione specificando il nome, il numero di ospiti e la tipologia dell'appartamento desiderato (standard o suite). Se il numero di ospiti soddisfa la disponibilità di posti per la tipologia scelta, le camera viene assegnata e la prenotazione viene memorizzata.

L'assegnazione delle camere modifica la disponibilità delle stesse togliendola dalle scelte effettuabili in quel periodo. In definitiva si avranno i seguenti parametri di:

- input
  - data di arrivo e data di partenza
  - nome
  - numero di ospiti
  - tipologia di camera
- output
  - disponibilità della/e camera/e nel periodo richiesto
  - assegnazione della/e camera/e
  - andamento dei ricavi mensili nell'anno corrente

1.3 Potenzialità e Criticità

Le potenzialità principali del programma sono quelle di ridurre i tempi di gestione alberghiera grazie all'uso semplice ed intuitiva dell'interfaccia e di valutare approssimativamente i ricavi mensili nell'arco dell'intero anno.

Una criticità del programma consiste nel fatto che, aumentando il numero di prenotazioni presenti nel database, si può verificare una risposta più lenta da parte del sistema. In secondo luogo si possono creare "buchi di prenotazioni" ossia si hanno camere libere in alcuni giorni tra due prenotazioni; questo è dovuto al fatto che il programma alloca nel miglior modo possibile, nel periodo definito, il numero di persone specificato, senza riorganizzare le prenotazioni inserite in precedenza.

1.4 Descrizione del data-set

I data-set sono stati forniti dall'azienda alberghiera citata e si di distinguono in due serie:

- le prenotazioni del periodo estivo 2013;
- il numero e la tipologia delle camere.

I data-set sono poi stati utilizzati per la creazione del database realizzato con *HeidiSQL* (Figura 1.1).

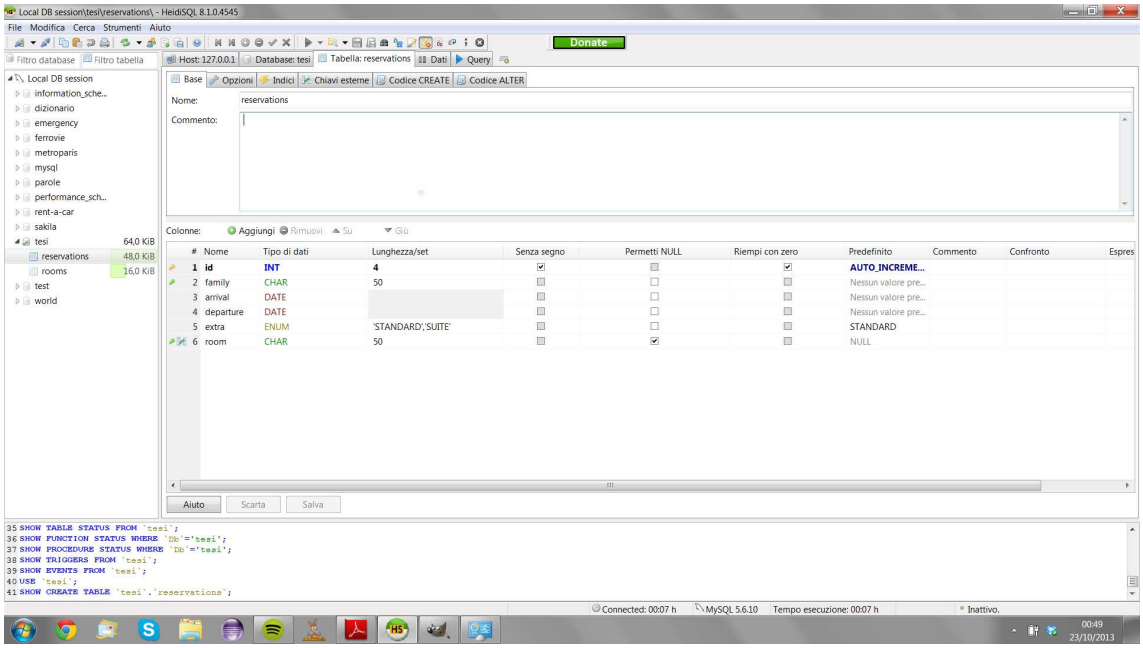

Figura 1.1: Programma per la gestione dei database.

Il database e le connessioni reciproche tra i data-set sono state realizzate come visibile in figura 1.2;

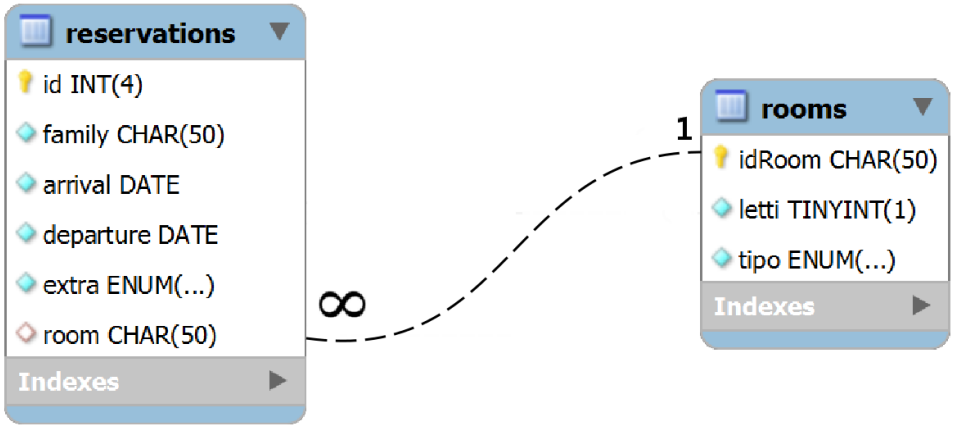

Figura 1.2: Visualizzazione dei collegamenti tra i data-set.

il numero 1 presente accanto alla tabella *rooms* indica che la prenotazione può associarsi univocamente ad una stanza; d'altro canto una stanza può essere associata a più *reservations* (come indicato dal simbolo ∞). Ogni volta che nel programma una prenotazione va a buon fine, questa viene automaticamente inserita nel data-set di *reservations*.

1.5 Descrizione degli algoritmi e strutture di dati

Gli algoritmi principali, utilizzati nella realizzazione del programma, sono sostanzialmente tre:

- algoritmo di ricerca delle camere disponibili
- algoritmo di prenotazione
- algoritmo di assegnazione camere

Le principali strutture dati utilizzate nel software ed in particolar modo negli algoritmi sopra citati, sono di tipo *List* e *Map*.

1.5.1 Algoritmo di ricerca delle camere disponibili

Nella classe *Controller1.java*, che gestisce l'interfaccia, si definisce il comando:

```
allFreeRooms= hotel.calcolaFree(arrival, departure);
```

che richiama l'algoritmo di ricerca presente nella classe *HotelModel.java*:

```
public List<Room> calcolaFree(Date da, Date a) {
  List<Room> allFreeRooms = new LinkedList<Room>(this.getCamere()
      .values());
  if (!this.getPrenotazioni().isEmpty()) {
    List<Reservation> allRes = new ArrayList<Reservation>(this
        .getPrenotazioni().values());
    for (Reservation res : allRes) {
      if (res.getArrival().equals(da)) {
        allFreeRooms.remove(res.getRoom());
            }
      if (res.getArrival().after(da) && !res.getArrival().equals(da)
          && res.getArrival().before(a)) {
        allFreeRooms.remove(res.getRoom());
            }
      if (res.getArrival().before(da) && res.getDeparture().after(da)
          && !res.getDeparture().equals(da)) {
        allFreeRooms.remove(res.getRoom());
      }}}
  return allFreeRooms;
}
```

Listing 1.1: Algoritmo di ricerca camere disponibili

L'algoritmo riceve in ingresso la data di arrivo e di partenza della prenotazione da inserire. Il periodo così definito viene confrontato con le prenotazioni esistenti in modo da evitare sovrapposizioni. L'algoritmo elimina la disponibilità delle camere già occupate e restituisce in uscita la lista delle camere libere.

1.5.2 Algoritmo di prenotazione

Nella classe *Controller1.java*, che gestisce l'interfaccia, si definisce il comando:

```
List<Reservation> r = hotel.addPrenotazione(family.getText(), arrival,
    departure, ospiti.getValue(), extra.getValue());
```

che richiama l'algoritmo di prenotazione presente nella classe *HotelModel.java*:

```

  public List<Reservation> addPrenotazione(String family, Date arrival,
      Date departure, int persone, String extra) {
    int totPosLib = 0;
    List<Room> free = calcolaFree(arrival, departure);
    List<Reservation> temp = new LinkedList<>();
    if (!free.isEmpty()) {
      if (persone < 63) {
        for (Room r : free) {
          if (r.getTipo().toString().equals(extra))
            totPosLib += r.getLetti();
        }
        if (persone <= totPosLib) {
            List<Room> assegnate = assegnaCamere(free, persone,
                extra);
            if (!assegnate.isEmpty()) {
              for (Room t : assegnate) {
                Reservation a = new Reservation(idRes, family,
                    arrival, departure, t);
                temp.add(a);
                prenotazioni.put(idRes, a);
                idRes++;
              }}}
      } else {
        for (Room r : free) {
          totPosLib += r.getLetti();
        }
        if (persone <= totPosLib) {
          List<Room> assegnate = assegnaCamere(free, persone);
          if (!assegnate.isEmpty()) {
            for (Room t : assegnate) {
              Reservation a = new Reservation(idRes, family,
                  arrival, departure, t);
              temp.add(a);
              prenotazioni.put(idRes, a);
              idRes++;
            }}}}}
    return temp;
  }
```

Listing 1.2: Algoritmo di prenotazione

L'algoritmo riceve in ingresso i dati della prenotazione (*Nome*, *Data di arrivo*, *Data di partenza*, *Numero ospiti*, *Extra*), viene calcolato il numero totale di posti liberi nella struttura (considerando l'extra) e se tale numero è superiore al numero di ospiti si procede con l'esecuzione degli algoritmi di assegnazione delle camere; se il feedback è positivo, l'algoritmo restituisce la prenotazione avvenuta. Nel caso di prenotazioni superiori a 63 posti, poiché le camere restanti sono di tipologia "suite" viene ignorata la richiesta dell'*Extra*.

1.5.3 Algoritmo di assegnazione camere

Nella classe *HotelModel.java*, all'interno dell'algoritmo di prenotazione (1.2), si definisce il comando:

```
List<Room> assegnate = assegnaCamere(free, persone, extra);
```

che richiama l'algoritmo di assegnazione delle camere:

```
private List<Room> assegnaCamere(List<Room> free, int persone,
                                                    String extra) {
List<Room> assegnate = new LinkedList<>();
    Collections.sort(free, new Comparator<Room>() {
      @Override
      public int compare(Room a, Room a1) {
        return -(a.getLetti() - a1.getLetti());
      }});
    int res = persone;
    while (res > 0) {
      Room temp = null;
      int avanzo = 0;
      int i = 0;
      for (Room a : free) {
        if (a.getTipo().toString().equals(extra)) {
          if (res >= a.getLetti() && avanzo==0) {
            if (res - a.getLetti() >= 2 || res - a.getLetti() == 0) {
              temp = a;
              break;
            }}
        if(res<a.getLetti()){
            if (avanzo == 0) {
              temp = a;
              avanzo = a.getLetti() - res;
            }
            if ((i = a.getLetti() - res) < avanzo) {
              avanzo = i;
              temp = a;
            }}}}
      assegnate.add(temp);
      res -= temp.getLetti();
      free.remove(free.indexOf(temp));  }
  return assegnate;
  }
```

Listing 1.3: Algoritmo di assegnazione camere

L'algoritmo riceve in ingresso la lista di camere disponibili (ordinate in modo decrescente per numero di letti), il numero di persone e l'*Extra*; scorre la lista delle camere libere e, se la camera corrisponde all'*Extra* richiesto, viene confrontata con le altre dello stesso tipo al fine di assegnare la camera o l'insieme di camere il cui numero di posti totale disponibili è il più vicino possibile al numero di ospiti. L'algoritmo restituisce in uscita la lista delle camere assegnate.

Se il numero di ospiti è maggiore di 63 (avendo già controllato, ad un livello superiore, la possibilità di accoglierli nella struttura), verrà applicato un algoritmo analogo che non tiene conto dell'*Extra*:

```
List<Room> assegnate = assegnaCamere(free, persone);
```

1.6 Diagramma delle classi delle parti principali

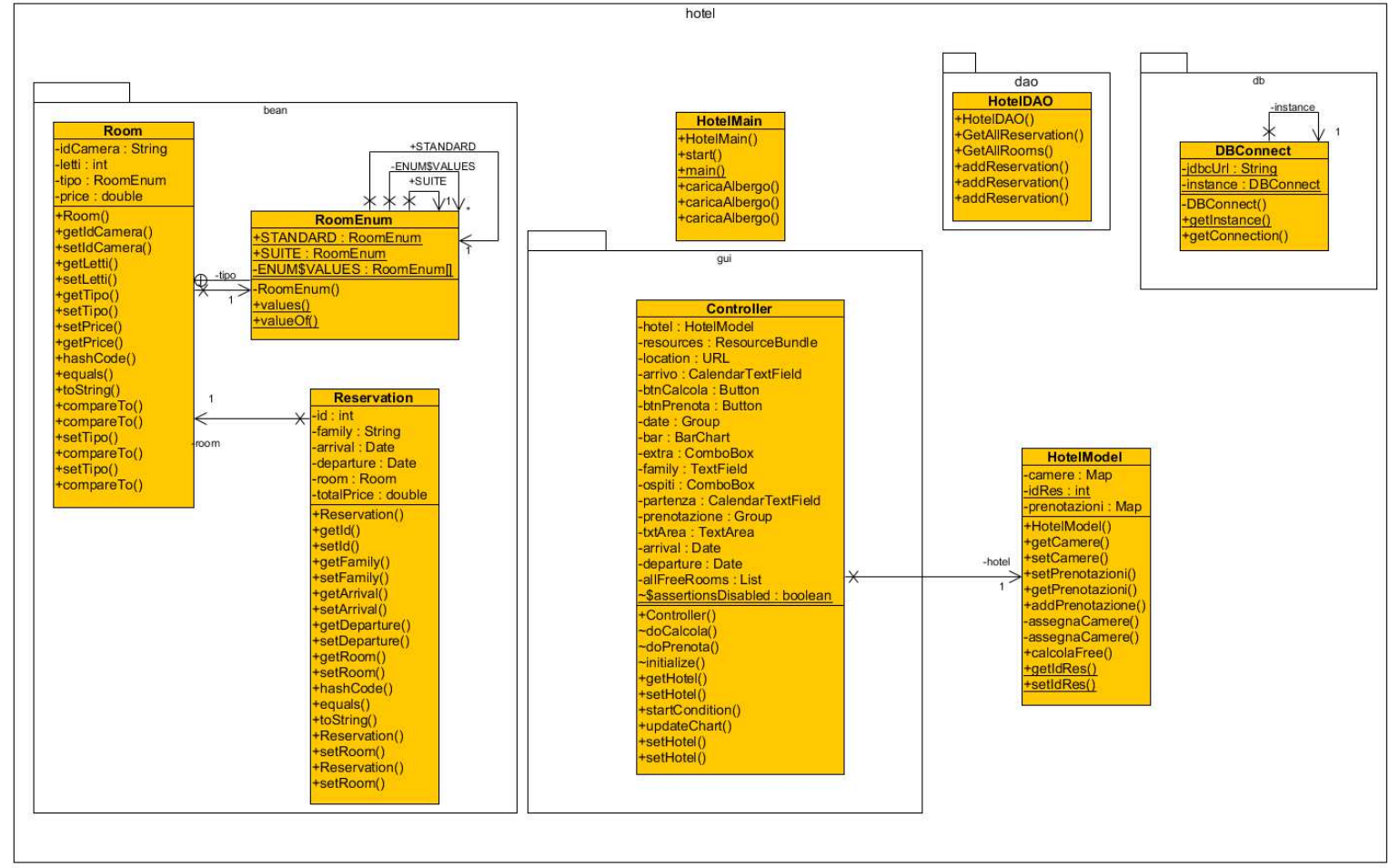

Figura 1.3: Diagramma delle classi delle parti principali.

1.7 Videate dell'applicazione e link al video dimostrativo

Figura 1.4: Schermata videata principale.

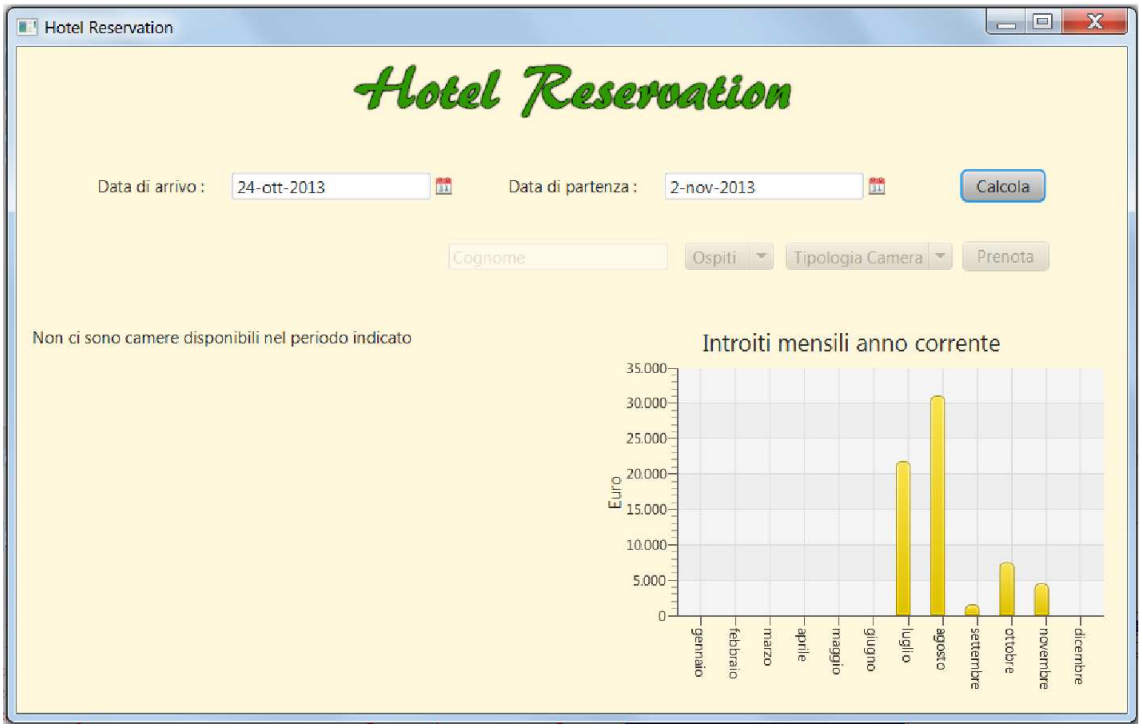

Figura 1.5: Videata con periodo di prenotazione senza camere disponibili.

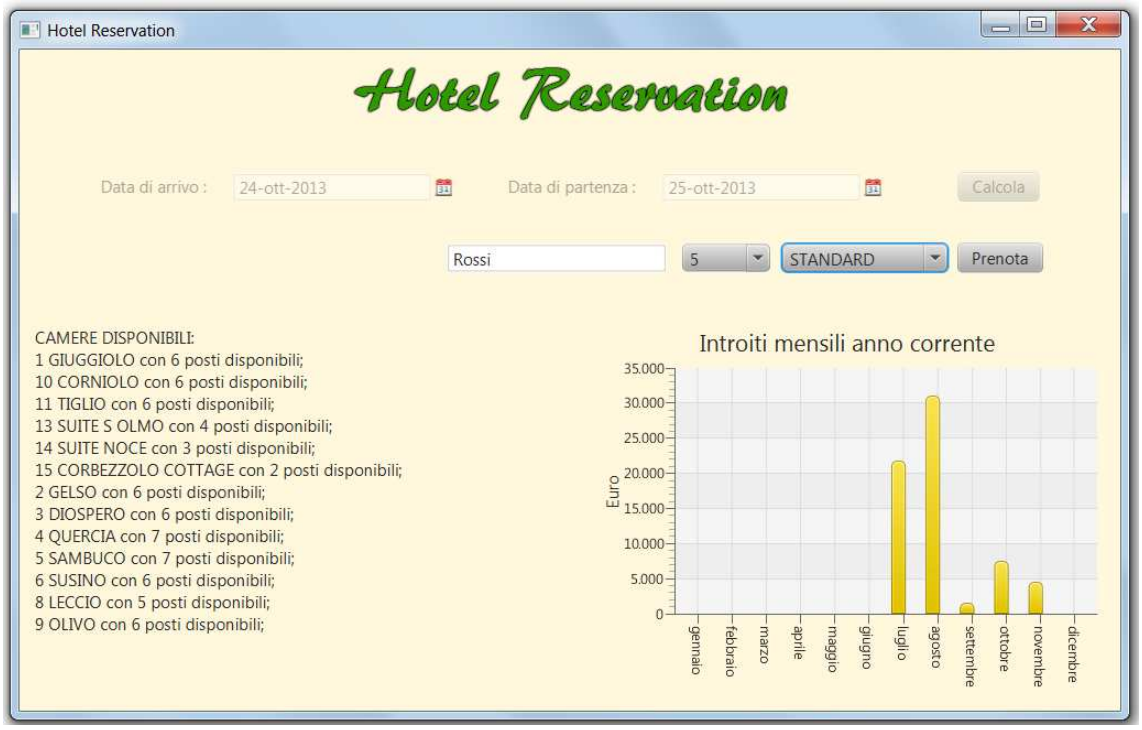

Figura 1.6: Disponibilità camere e inserimento dati di prenotazione.

Figura 1.7: Conferma di prenotazione di una sola camera.

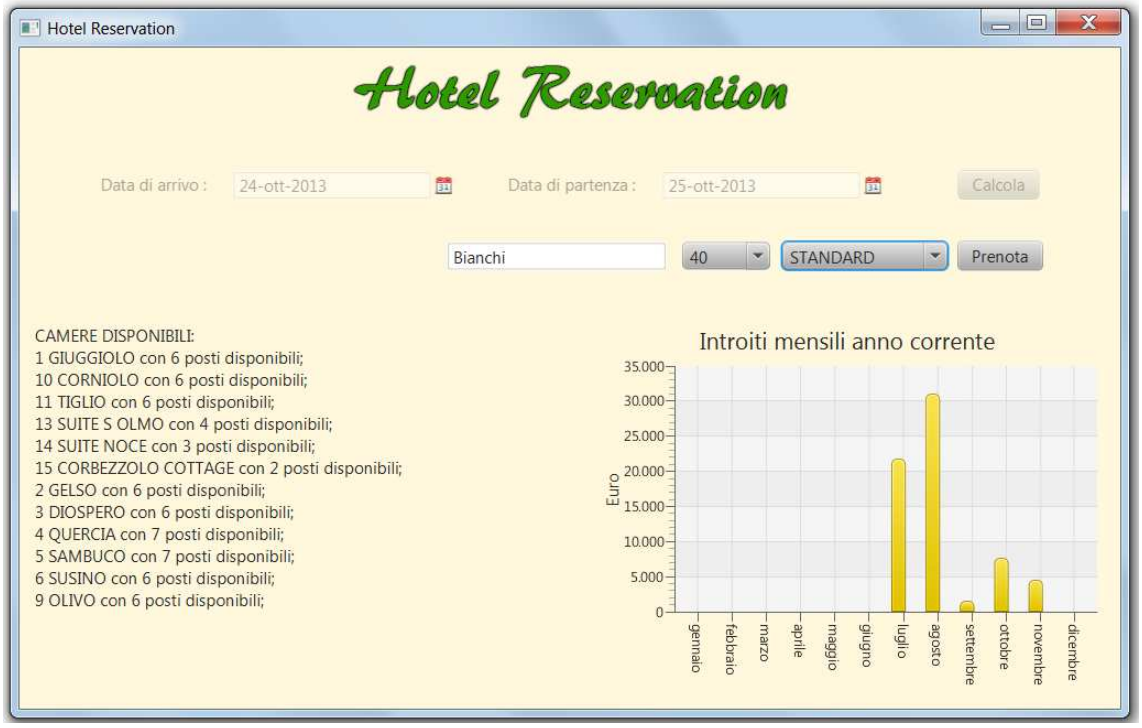

Figura 1.8: Inserimento dati prenotazione multipla.

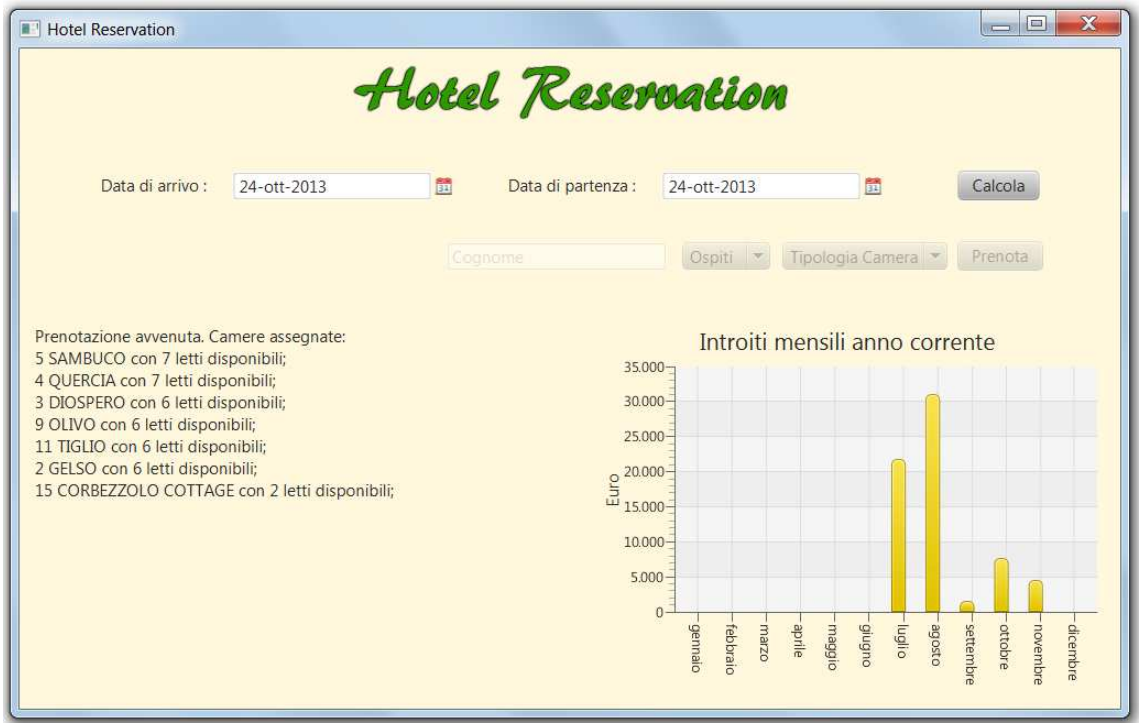

Figura 1.9: Conferma di prenotazione di camere multiple.

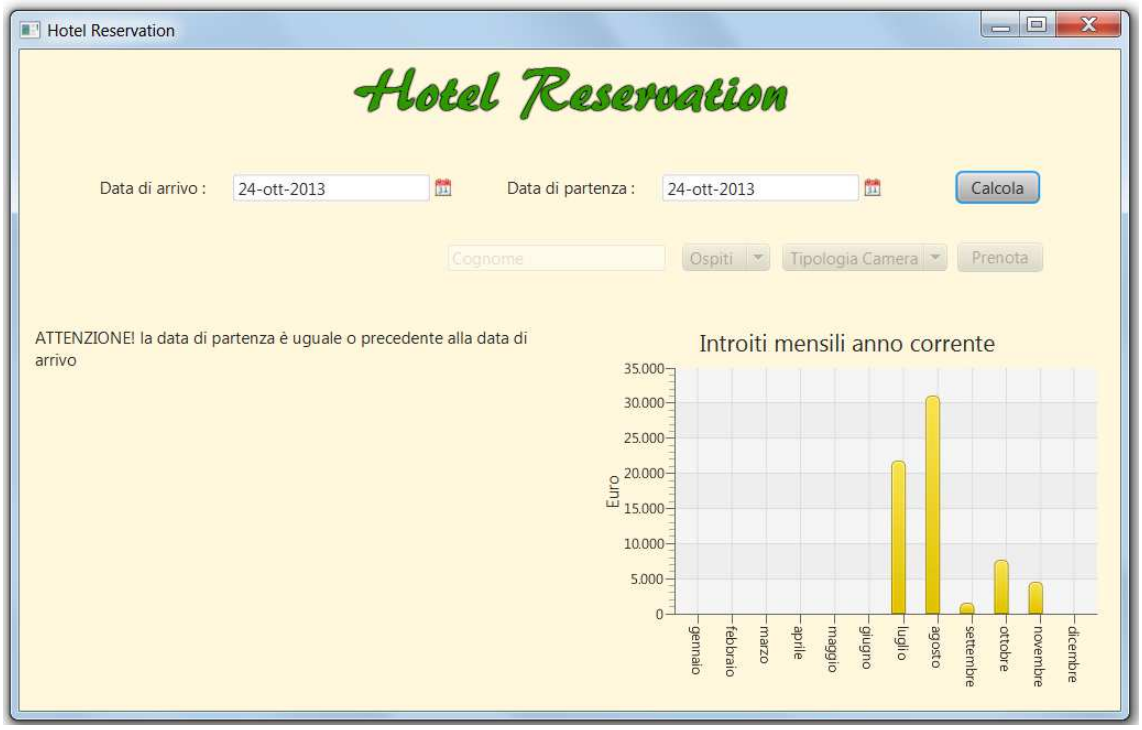

Figura 1.10: Errore inserimento periodo di prenotazione.

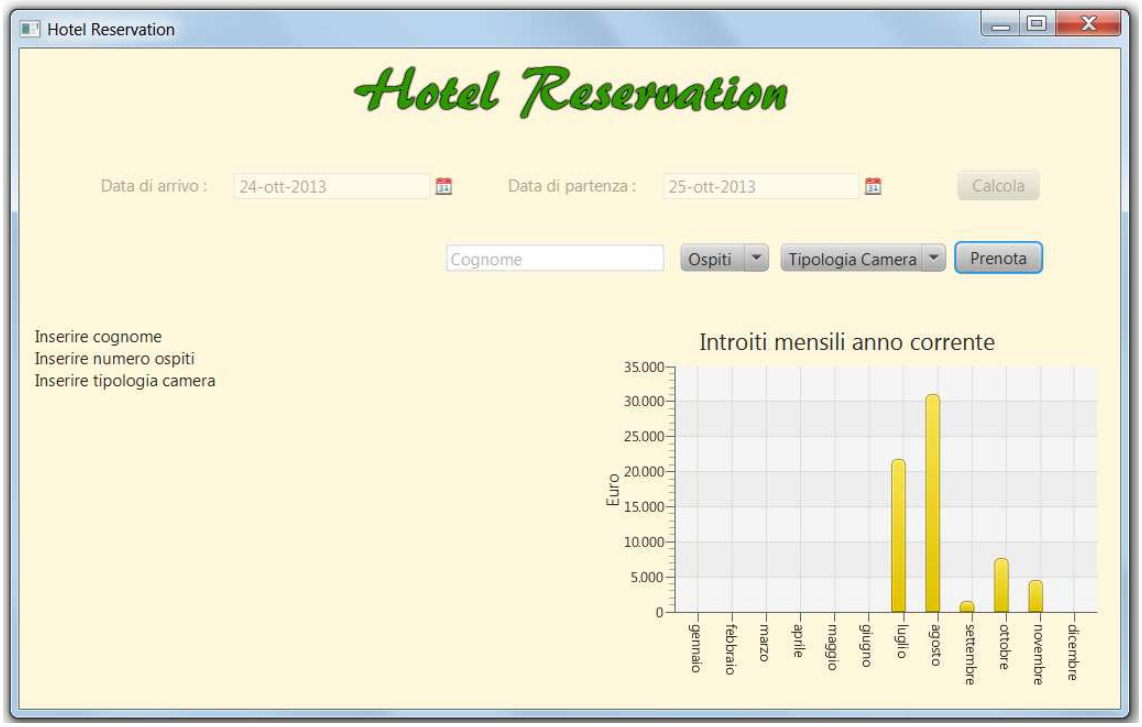

Figura 1.11: Errore inserimento dati.

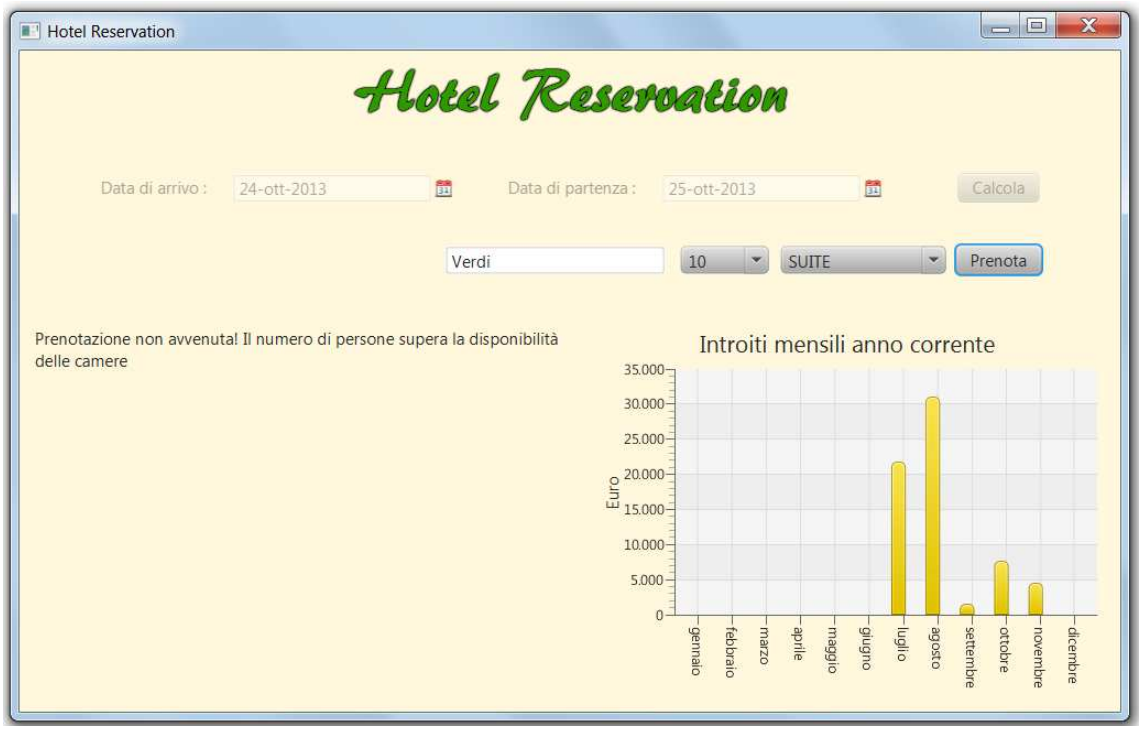

Figura 1.12: Errore di disponibilità: il numero di persone supera il numero di posti disponibili.

Di seguito riportiamo il link dimostrativo dell'applicazione realizzata:
`http://youtu.be/JJKv16Ab4rA`

1.8 Conclusioni

Come si può vedere nel paragrafo 1.7 e nel video dimostrativo , il software ha un'interfaccia semplice ed intuitiva, che permette di inserire velocemente le prenotazioni e di controllare le camere disponibili. Questo apporta un notevole miglioramento della gestione della struttura riducendo i tempi amministrativi. Il confronto tra i dati simulati e i dati reali, la maggior parte delle volte, ha dato un esito positivo. Miglioramenti futuri possono vertere sull'ottimizzazione dell'algoritmo di assegnazione camere, in modo da non creare i già citati "buchi di prenotazione". D'altro canto, la modifica di questo algoritmo aumenterebbe i tempi di risposta del programma, in quanto l'allocazione di una nuova camera, comporterebbe la riallocazione delle prenotazioni esistenti, al fine di creare un calendario che sia il più compatto possibile. Il programma realizzato soddisfa appieno le ipotesi iniziali e costituisce un buon punto di partenza per sviluppi successivi.