Università degli Studi di Roma "La Sapienza"
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Gestionale
Corso di Progettazione del Software
Proff. Toni Mancini e Monica Scannapieco

Progetto **PC.20080703**

versione del 9 luglio 2008

Si vuole progettare e realizzare *DormoDaTe.com*, un sistema per un portale web che consenta agli iscritti di chiedere ospitalità, durante i loro viaggi, a casa di altri iscritti. Lo scopo, oltre che quello banale di ridurre le spese di viaggio, è soprattuto quello di incentivare scambi sociali e senso di ospitalità, e di favorire nuove conoscenze.

Si richiede di effettuare le fasi di Analisi, Progetto, e Realizzazione del sistema in JAVA, utilizzando la metodologia illustrata nel corso.

Requisiti

Il sistema *DormoDaTe.com*[1] deve consentire agli utenti di iscriversi al portale web, dichiarando il loro profilo, ovvero nome, cognome, sesso, età, città di residenza, oltre che informazioni circa la tipologia di ospitalità che sono disposti ad offrire.

In particolare, è necessario che gli iscritti possano inserire nel loro profilo le seguenti informazioni:

- Distanza della loro abitazione dal centro città;
- Distanza dalla stazione autobus/metro/treno più vicina;
- Numero di membri della famiglia (numero di adulti e di bambini);
- Numero di persone che sono disposti ad ospitare contemporaneamente (posti letto); dei posti letto va specificato (nel modo più preciso e strutturato possibile) se si tratta di letti (singoli/doppi) in camere separate, divani in stanze comuni, ecc. (ad esempio, un iscritto può dichiarare di avere un posto in letto singolo più due posti in letto doppio in una camera, ed un posto in soggiorno sul divano).

[1] Il sistema vuole mettersi in diretta concorrenza con il ben più famoso *couchsurfing.com*.

In ogni momento ogni iscritto ha la facoltà di aggiornare le informazioni circa le sue disponibilità ad ospitare altre persone, sulla base dei suoi impegni. In particolare, il sistema deve prevedere ad ogni iscritto la possibilità di dichiarare delle date o dei periodi nei quali non è disponibile a ricevere ospiti, ad es. perché fuori (la ragione non è di interesse per il sistema).

I viaggiatori che desiderano organizzare un viaggio usufruendo del servizio devono poter interrogare il sistema per ottenere l'insieme di tutti gli iscritti della città desiderata che sono disposti a ricevere ospiti nel periodo richiesto e hanno, per le date richieste, posti sufficienti disponibili.

Una volta scelta una soluzione di alloggio, i viaggiatori possono effettuare la relativa richiesta di prenotazione, specificando tutte le informazioni necessarie (date, persone, scelta dei posti letto, ecc.). Si noti che è possibile per un viaggiatore prenotare anche per eventuali accompagnatori: tuttavia il sistema impone che tutti (chi prenota ed eventuali accompagnatori) siano iscritti al sistema (questo per monitorare la qualità e l'affidabilità degli ospiti, cf. seguito). Il relativo ospitante (membro padrone di casa) può decidere liberamente se accettare la richiesta o rifiutarla (in quest'ultimo caso deve specificare una motivazione). Tutte le richieste di prenotazioni, quelle accettate e quelle rifutate, devono essere mantenute in modo persistente dal sistema.

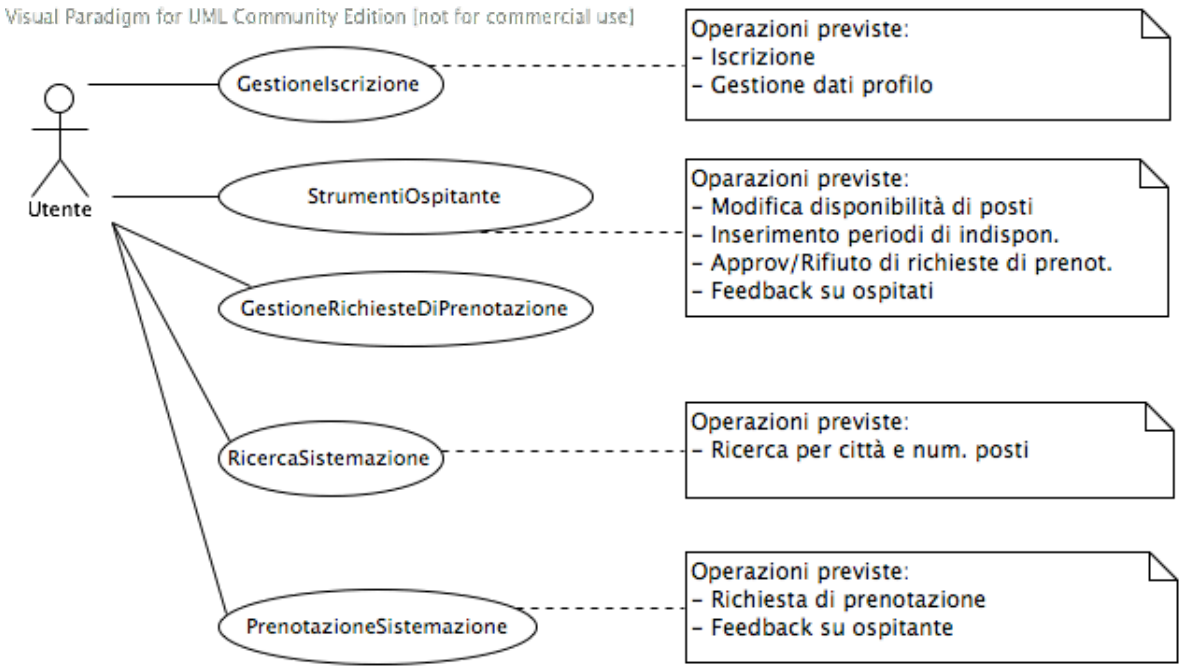
La difficoltà maggiore che si riscontra in sistemi del genere è quella relativa alla sicurezza, visto che, di fatto, gli iscritti offrono ospitalità a casa propria a perfetti estranei (e, dualmente, un viaggiatore va a dormire a casa di sconosciuti). Per questo motivo, sono di grande importanza quei mecccanismi che permettano di avere maggiore confidenza circa la qualità e l'affidabilità dell'ospitante e degli ospitati. In particolare, *DormoDaTe.com* deve permettere dei meccanismi di feedback per valutare la qualità e l'affidabilità sia dei padroni di casa che dei viaggiatori. In dettaglio:

- Ogni viaggiatore, dopo aver alloggiato presso un iscritto ospitante, deve necessariamente esprimere una valutazione su quest'ultimo, in termini di un intero da 0 (scarso) a 5 (ottimo);
- Ogni ospitante, all'atto della partenza di un ospite, può esprimere una valutazione su quest'ultimo, anche qui in termini di un intero da 0 a 5.

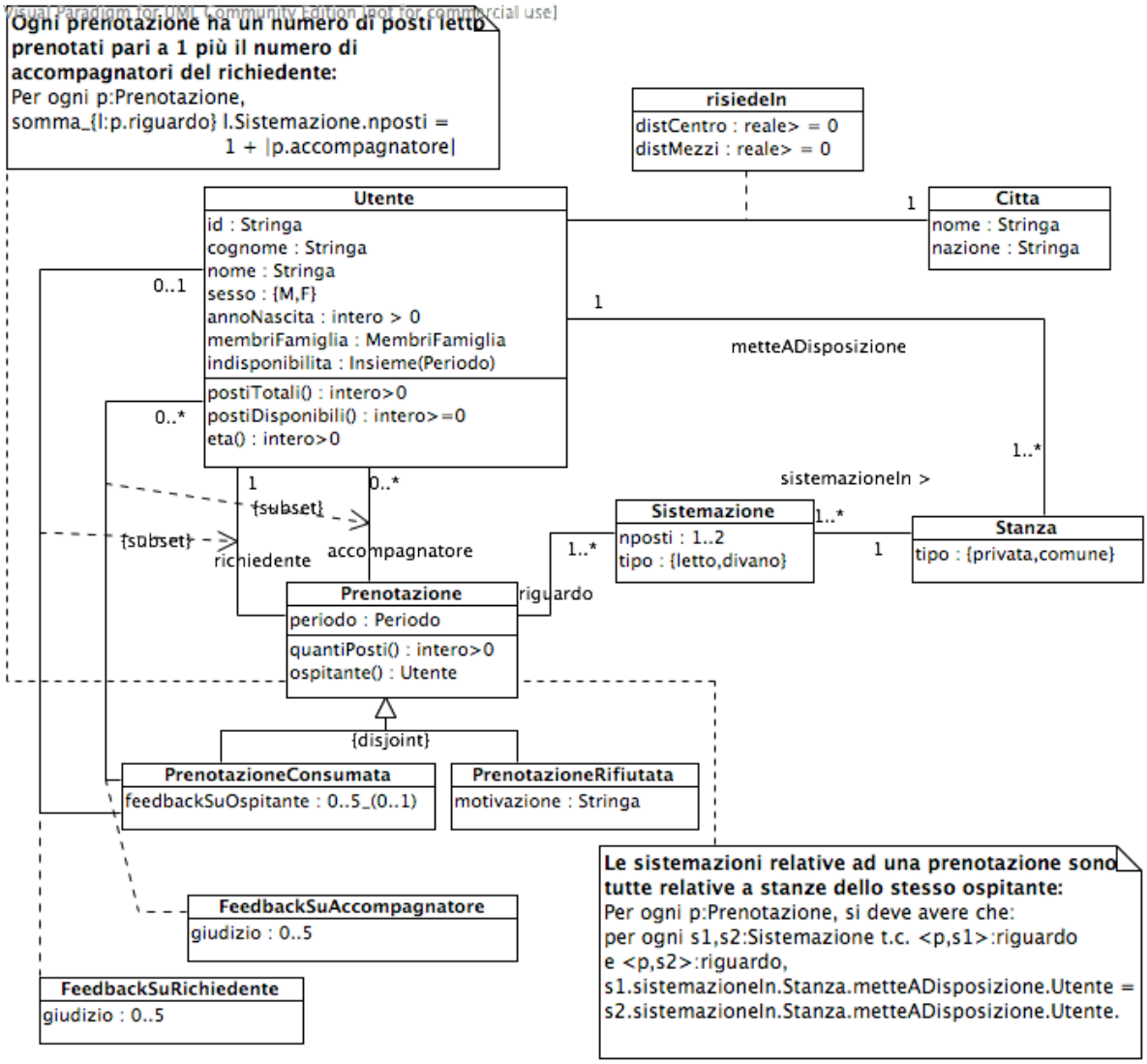
Si noti che, la valutazione di un ospitante su un ospitato è facoltativa, mentre quella opposta è obbligatoria. In particolare, sebbene quest'ultima possa essere effettuata anche in un tempo successivo, il sistema deve fare in modo che un viaggiatore non possa prenotare più alcuna nuova sistemazione se non ha valutato tutti gli ospitanti presso cui ha alloggiato.

1 Fase di Analisi

1.1 Diagramma degli Use Case



1.2 Diagramma delle classi UML



Nota: I vincoli di molteplicità non presenti devono intendersi 0..*.

1.3 Specifica dei tipi di dato

```
SpecificaTipoDiDato Periodo
  attributi
```

```
    adulti: intero > 0  // almeno un adulto, l'utente stesso
    bambini: intero >= 0
FineSpecifica

SpecificaTipoDiDato Periodo
  attributi
    inizio: Data
    fine: Data

    Periodo(i:Data, f:Data): Periodo
      pre:  i<=f
      post: result e' l'istanza del tipo con result.inizio = i e result.fine = f

    siSovrapponeA(p:Periodo): booleano
      pre: nessuna
      post:
          se this.fine <= p.inizio oppure this.inizio >= p.fine, allora result = false;
          altrimenti result = true.
FineSpecifica
```

1.4 Specifica degli use case

```
SpecificaUseCase GestioneIscrizione
  iscrizione(id:Stringa, c:Stringa, n:Stringa, s:{M,F}, annoNascita:intero>0, ...) : Utente
    pre: nessuna
    post: result e' un nuovo oggetto di classe Utente con...
    // Da completare per esercizio
FineSpecifica

SpecificaUseCase StrumentiOspitante
  aggiungiSistemazioneIn(u:Utente, tipo:{letto,divano}, n:1..2, in:Stanza)
    pre: in.metteADisposizione.Utente = u
    post: sia s:Sistemazione un nuovo oggetto con s.tipo......
             viene aggiunto il link....
             // Da completare per esercizio

  aggiungiNuovaStanza(u:Utente, tipo:{privata,comune}, nposti:1..2, tipoSist:{letto,divano})
    pre: ...
    post: ...
```

```
            // Da completare per esercizio
 rimuoviSistemazione(...)
           // Da completare per esercizio

 inserisciIndisponibilita(u:Utente, p:Periodo)
    pre: nessuna
    post: u.indisponibilita = pre(u.indisponibilita) U {p}

 approvaRichiestaDiPrenotazione(u:Utente, p:Prenotazione)
   pre: p.ospitante() = u,
          p e' nello stato 'pendente',
          oggi < p.periodo.inizio,
          per ogni s:Sistemazione t.c. <p,s>:riguardo si ha: s.e'Libera(p.periodo)

   post: viene generato l'evento 'approva' su p

 rifiutaRichiestaDiPrenotazione(u:Utente, p:Prenotazione, motivaz:Stringa)
   pre: p.ospitante() = u,
          p e' nello stato 'pendente',
          oggi < p.periodo.inizio
   post: viene generato l'evento 'rifiuta' su p (p diventa di classe PrenotazioneRifiutata).
          p.motivazione = motivaz.

 esprimiFeedbackSuOspite(u:Utente, p:Prenotazione, o:Ospite, g:0..5)
   pre: p.ospitante() = u,
          p e' nello stato 'consumata',
          <o,p> in richiedente oppure <o,p> in accompagnatore
   post:
          se <o,p>: richiedente viene creato un link l di tipo feedbackSuRichiedente con
          l.giudizio = g
          altrimenti viene creato un link l di tipo feedbackSuOspitante con
          l.giudizio = g
FineSpecifica

SpecificaUseCase RicercaSistemazione
  ricercaPerCitta(c:Citta, n:intero>0, per:Periodo): Insieme(Utente)
    pre: nessuna
    post:
       result = { u:Utente | <u,c>:risiedeIn e u.postiDisponibili(per)>=n }
FineSpecifica
```

```
SpecificaUseCase PrenotazioneSistemazione
  prenotazione(rich:Utente, accomp:Insieme(Utente), dest:Utente,
                                per:Periodo, sist:Insieme(Sistemazione))
    pre: !(rich in accomp), rich != dest,  !(dest in accomp),
           oggi < per.inizio,

           // le sistemazioni richieste sono nel giusto numero

                              Σ_{s:sist} s.nposti = 1 + |accomp|

          // le sistemazioni richieste sono tutte di 'dest' e tutte libere
          per ogni s:sist si ha:
            - s.sistemazioneIn.Stanza.metteADisposizione.Utente = dest;
            - s.e'Libera(per)

           // il richiedente ha gia' valutato tutte le sue precedenti esperienze
           non esiste p:Prenotazione t.c.
              - p.richiedente.Utente = rich;
              - p e' nello stato 'consumata'
              - p.feedbackSuOspitante e' indefinito

    post:
              viene creato p:Prenotazione (nello stato 'pendente'), con:
                - p.periodo = per;
              Vengono creati i link:
                - <p,rich>:richiedente
                - <p,a>:accompagnatore per ogni a in accomp
                - <p,s>:riguardo, per ogni s:sist.

  feedbackSuOspitante(ospite:Utente, p:Prenotazione, g:0..5)
    pre: <o, p>:richiedente, p e' nello stato 'consumata',
         p.feedbackSuOspitante e' indefinito
    post: p.feedbackSuOspitante = g
FineSpecifica
```

1.5 Specifica delle classi e diagrammi degli stati e transizioni

La classe Utente

```
SpecificaClasse Utente
  eta(): intero > 0
    pre: this.annoNascita < oggi.anno
    post: result = oggi.anno - this.annoNascita

  postiTotali(): intero > 0
    pre: nessuna
    post:
```

$$\texttt{result} = \Sigma_{\texttt{mad:this.metteADisposizione}} \Sigma_{\texttt{sin:mad.Stanza.sistemazioneIn}} \texttt{sin.Sistemazione.nposti}$$

```
  postiDisponibili(per:Periodo): intero >= 0
    pre: oggi <= per.inizio
    post:
      se esiste p in this.incompatibilita tale che
           p.siSovrapponeA(per) = true, result = 0;

     altrimenti detto
     SL = { s:Sistemazione | s.sistemazioneIn.Stanza.metteADisposizione.Utente = this
                             e s.e'Libera(per) }
     l'insieme delle sistemazioni dell'utente this libere per tutto il periodo 'per'
```

$$\texttt{result} = \Sigma_{\texttt{s} \in \texttt{SL}} \texttt{s.nposti}$$

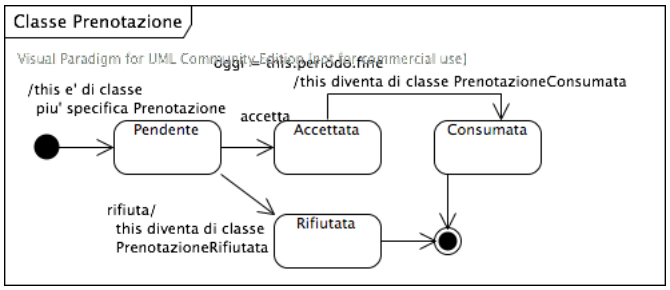
```
FineSpecifica
```

La classe Prenotazione

```
SpecificaClasse Prenotazione
  quantiPosti(): intero > 0
    pre: nessuna
    post: result = 1 + |this.accomp|

  ospitante(): Utente
    pre: nessuna
    post: result = l.Sistemazione.sistemazioneIn.Stanza.metteADisposizione.Utente
          per un qualsivoglia l:this.riguardo.
FineSpecifica
```

Gli oggetti della classe Prenotazione evolvono secondo il seguente diagramma degli stati e transizioni:



La sottoclasse PrenotazioneConsumata della classe Prenotazione rappresenta lo stato ‘consumata’ degli oggetti di tale classe.

La classe Sistemazione

```
SpecificaClasse Sistemazione
  e’Libera(per:Periodo): booleano
    pre: nessuna
    post: result e’ true se e solo se non esiste p:Prenotazione t.c.:
            - <p,this>:riguardo
            - p.periodo.siSovrapponeA(per)
            - p non e’ nello stato ‘rifiutata’
FineSpecifica
```