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.20080918**
versione del 19 settembre 2008

Si vuole progettare e realizzare *Out!*, un sistema informatico su Web che consenta agli utenti di consultare il programma di diversi cinema e teatri cittadini, e di prenotare gli spettacoli a cui desiderano assistere.

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

## Requisiti

*Out!* deve consentire di mantenere informazioni su teatri e cinema e ogni altra sede di spettacoli associata al circuito. In particolare, di ogni sede interessa conoscerne il nome e l'indirizzo. Le sedi possono essere provviste di più sale distinte, ognuna delle quali può ospitare uno spettacolo (ad es., cinema multisala, auditorium, etc.). Il sistema deve anche mantenere informazioni circa i posti a sedere nelle singole sale delle diverse sedi. Di ogni posto interessa conoscere i numeri di fila e colonna.

*Out!* deve inoltre rappresentare gli spettacoli in programmazione presso i diversi enti aderenti al circuito. Di ogni spettacolo interessa conoscerne il titolo, la tipologia (concerto, rappresentazione teatrale, oppure film), il relativo genere e gli artisti che effettuano la performance, oltre che le date, gli orari e le sedi (con relativa sala) in cui viene rappresentato. Si osservi infatti che uno stesso spettacolo viene tipicamente rappresentato più volte; anche se tipicamente uno spettacolo viene rappresentato sempre nella stessa sede, il sistema deve permettere anche di rappresentare spettacoli itineranti. Per generalità dunque, si faccia il modo che il sistema possa mantenere informazioni circa la sede e la sala per ogni data in cui un certo spettacolo viene rappresentato.

Scopo principale del sistema è di consentire agli utenti di consultare il calendario degli spettacoli in programma, e di prenotare via web posti per gli spettacoli a cui desiderano assistere. In particolare, un utente deve poter usare il sistema per:

1. Iscriversi al servizio, fornendo i propri dati di interesse (nome, cognome, codice fiscale) che il sistema deve memorizzare;

2. Prenotare uno o più posti per una data di uno spettacolo. I posti non sono liberi, pertanto il sistema deve assegnare, all'atto della prenotazione, un numero di posti sufficienti non ancora prenotati.[1]

3. Consultare la lista degli spettacoli di una certa tipologia e genere (ad es., spettacoli teatrali/commedie) previsti in un certo giorno.

4. Ricevere dal sistema suggerimenti di nuovi spettacoli da vedere. In particolare, si richiede che il sistema segnali ad un utente l'insieme degli spettacoli programmati nei successivi 7 giorni, che sono dello stesso genere (anche se di tipologia diversa) dell'ultimo spettacolo da egli prenotato.

*Out!* deve consentire anche di memorizzare i prezzi dei biglietti per i diversi spettacoli. Si noti che, dato uno spettacolo, il prezzo di un posto è in generale diverso a seconda della data e della tipologia dei posti richiesti.

In particolare il modello adottato da *Out!* è il seguente. Le sale delle diverse sedi hanno i posti suddivisi in settori (ad es., i settori di una sala di un teatro possono essere "platea", "prima galleria", etc.). Di ogni settore il sistema deve memorizzare nome e insieme di posti.

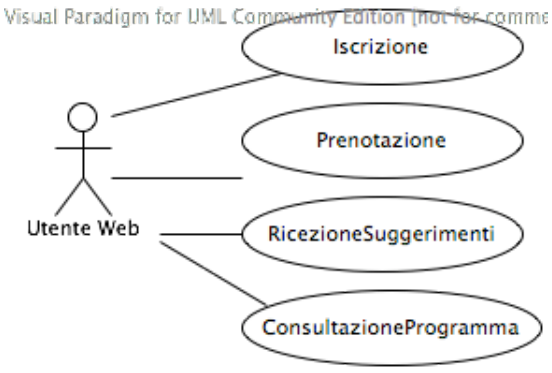
Il prezzo del biglietto di uno spettacolo dipende dal settore (della sala dove lo spettacolo viene rappresentato) al quale il posto prenotato appartiene. Ovviamente, sarà cura dei gestori dei singoli enti (che possono quindi utilizzare il sistema) memorizzare su *Out!* i prezzi (interi e ridotti) dei biglietti per ogni singola data di uno spettacolo e per ogni settore della sala.

La funzionalità di *Out!* di consentire la prenotazione di posti, deve permettere la scelta (da parte dell'utente) di posti a prezzo intero o ridotto.

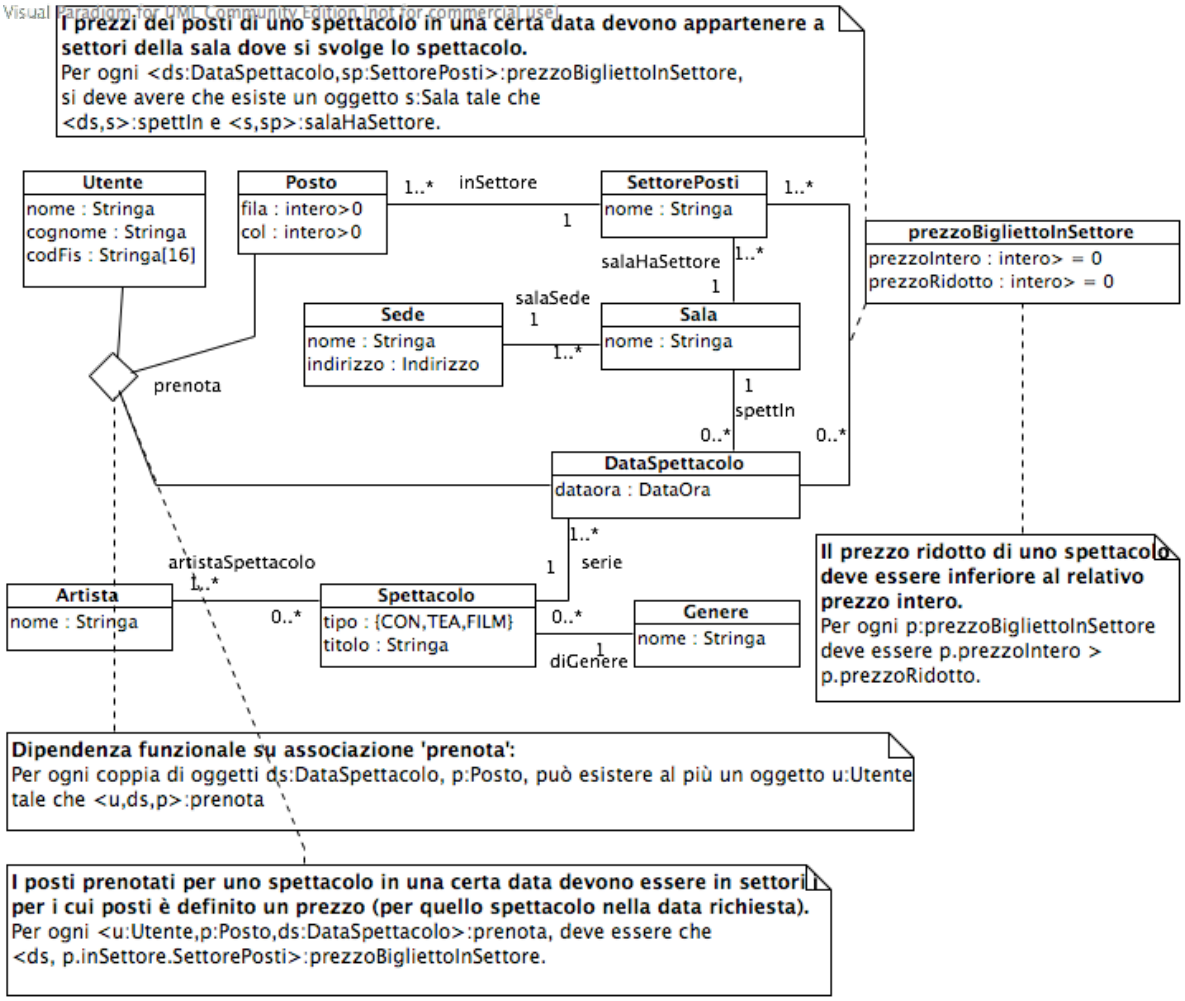
[1] Si assuma per semplicità che i posti vengano assegnati in modo completamente arbitrario, e non dal "migliore" al "peggiore".

# 1 Fase di Analisi

## 1.1 Diagramma degli Use Case



## 1.2 Diagramma delle classi UML




## 1.3 Specifica dei tipi di dato

```
SpecificaTipoDiDato Stringa[n]
  // stringhe di n caratteri
  attributi
    s: Stringa
  operazioni
    Stringa[n](str: Stringa) : Stringa[n]
```

```
      pre: |str| = n
      post: result e' l'istanza del tipo Stringa[n] con result.s = str
FineSpecifica
```

### 1.4 Specifica degli use case

```
SpecificaUseCase Iscrizione
  iscrizione(nome:Stringa, cogn:String, codFis:Stringa[16]): Utente
    pre: nessuna
    post: result e' un nuovo oggetto di classe Utente con
     - result.nome = nome
     - result.cognome = cogn
     - result.codFis = codFis
FineSpecifica


SpecificaUseCase Prenotazione
  prenota(u:Utente, nposti:intero > 0, s:SettorePosti, ds:DataSpettacolo)
    pre:
      - Per i posti del settore 's' e' stato definito un prezzo per l'evento ds:
        esiste il link <s,ds>:prezzoBigliettoInSettore
      - Esistono almeno 'nposti' ancora liberi in 's' per l'evento 'ds':


  |sd.inSettore| −
        Σ{pr:Prenota t.c.pr.DataSpettacolo = ds e pr.Posto.inSettore.SettorePosti = s} > nposti.

   post:
       Vengono creati 'nposti' nuovi link pr:Prenota con:
         - pr.Utente = u
         - pr.DataSpettacolo = ds
         - pr.Posto in Pliberi (scelto arbitrariamente)

         dove Pliberi = {p:Posto | <p,sd>:inSettore e non esiste alcun link pr':prenota
                                    t.c. pr'.Posto = p e pr'.DataSpettacolo = sd}.
         Le precondizioni garantiscono che |Pliberi| >= nposti.
FineSpecifica

SpecificaUseCase RicezioneSuggerimenti
```

```
  suggerisci(u:Utente): Insieme(Spettacolo)
    pre: 'u' deve aver prenotato almeno uno spettacolo in passato:
           esiste pr:Prenota tale che pr.Utente = u
    post:
          Sia ds:DataSpettacolo l'ultimo evento prenotato da u, ovvero l'oggetto
          di classe DataSpettacolo tale che:
              - esiste pr:Prenota tale che pr.Utente = u e pr.DataSpettacolo = ds;
              - Per ogni pr':Prenota tale che pr'.Utente = u, si ha
                pr'.DataSpettacolo.dataora < ds.dataora.

          result = {s:Spettacolo | s.diGenere.Genere=ds.serie.Spettatolo.diGenere.Genere
                                   ed esiste data_s:DataSpettacolo t.c.:
                                      - <s,data_s>:serie e
                                      - data_s.dataora.differenza(adesso,GIORNI)<=7 }

          dove 'adesso' e' l'istanza del tipo DataOra che denota l'istante corrente.
FineSpecifica

SpecificaUseCase ConsultazioneProgramma
  consultaPerTipoEgenere(t:{CON,TEA,FILM}, g:Genere, giorno:Data): Insieme(DataSpettacolo)
    pre: nessuna
    post: result = { ds:DataSpettacolo t.c.:
                               - ds.serie.Spettacolo.diGenere.Genere = g
                               - ds.serie.Spettacolo.tipo = t
                               - ds.dataora.data = giorno }
FineSpecifica
```

### 1.5 Specifica delle classi e diagrammi degli stati e transizioni

Non sono state definite operazioni di classe e diagrammi degli stati e transizioni.