Progetto G39: Emissione biglietti per eventi

Manuele Frigo Vr369360

Stefano Gugole Vr379898

## Requisiti

Si vuole progettare un sistema informativo di una società che gestisce l'emissione di biglietti per eventi che si svolgono in un palazzetto dello sport.
Ogni evento può essere una replica di uno spettacolo proposto in date diverse (ad esempio un concerto che si ripete per un mese) oppure un singolo evento che si svolge solo in una data (ad esempio la partita di pallacanestro del 5/3/2007).
Ogni evento è caratterizzato da un codice univoco, un nome, una tipologia (**opera, concerto classico, concerto rock/pop, prosa, partita di pallavolo, partita di pallacanestro, altro evento sportivo**), la data in cui si svolge, l'ora in cui inizia e la durata indicativa. Quando si tratta di una replica vengono registrate separatamente anche le informazioni che descrivono lo spettacolo: descrizione, artisti coinvolti e il periodo di svolgimento (composto da data inizio e data fine delle repliche). Infine quando si tratta di una partita di pallavolo o pallacanestro vengono registrare le squadre coinvolte e il risultato.
Per ogni evento si memorizzano i settori in cui è organizzato il palazzetto. Ogni settore è caratterizzato da un numero progressivo, un nome, dalla capienza e dal tipo (settore a posti numerati, settore a posti non numerati, settore con posti in piedi). Ogni settore è identificato dal numero progressivo e dall'evento.
Infine si memorizzano i biglietti che vengono venduti per il singolo evento registrando: il codice univoco del biglietto emesso, la data di emissione, l'evento a cui si riferisce, il prezzo, il settore del palazzetto dove è ubicato il posto acquistato e il numero del posto (solo se si tratta di un settore con posti numerati). Il sistema deve permettere agli utenti di effettuare la prenotazione di uno o più biglietti per un certo evento, selezionando oltre che all'evento di una certa data anche il settore del palazzetto prescelto ed eventualmente del posto specifico se si tratta di posti numerati.

### Progetto concettuale

Si veda in allegato il modello E-R.

Si assume che ogni evento è univocamente contraddistinto dal suo codice, che varia di giorno in giorno anche se l'evento è ripetuto per una settimana. Questa scelta è voluta per gestire le eventuali modifiche dell'evento (orario, attori, prezzo,...).

## Progetto logico

```
create domain sect_type as text
check (value='posti numerati' or value='posti non numerati'
or value='posti in piedi');

create domain ti_type as text
check (value='opera' or value='concerto classico' or
value='concerto pop o rock' or value='prosa' or
value='partita di pallavolo' or value='partita di
pallacanestro' or value='altro evento sportivo');

create table evento(
codice varchar(10) primary key,
nome varchar(40) not null,
ora_inizio time not null,
durata time not null,
data_evento date not null,
tipologia ti_type not null,
descrizione varchar(100),
inizio_periodo date,
fine_periodo date,
risultato varchar(5),
squadra1 varchar(10),
squadra2 varchar(10),
CHECK ((inizio_periodo = null AND fine_periodo = null )
OR (data_evento<=fine_periodo AND data_evento >=
inizio_periodo))
);

create table artista(
CF varchar(25) primary key,
nome varchar(14) not null,
cognome varchar(14) not null
);

create table partecipanti(
id serial primary key,
```

```
evento varchar(10) not null references evento(codice),
artista varchar(25) not null references artista(CF)
);

create table settore(
numero numeric,
evento varchar(10) references evento(codice),
nome   varchar(8) not null,
tipo sect_type not null,
capienza numeric not null,
prezzo real not null default 0,
primary key(numero,evento)
);

create table biglietto(
codice varchar(10) primary key,
data_emissione date not null ,
prezzo real not null ,
evento varchar(10) not null references evento(codice),
settore numeric not null,
numero_posto numeric,
foreign key (settore,evento) references
settore(numero,evento)
);
```

**Popolamento database**

```
insert into evento values(
'as123432',
'opera traviata di Verdi',
'20:00',
'3:00',
'20/08/2015',
'opera',
'opera lirica',
'20/08/2015',
'25/08/2015',
null,
null,
null
);

insert into evento values(
'pp129438',
'partita pallavolo vigasio arbizzano',
'15:30',
'1:30',
'13/08/2015',
'partita di pallavolo',
null,
null,
null,
null,
'arbizzano',
'vigasio'
);

insert into artista values('123','Alberto','Rogati');
insert into artista values('asdf','Marco','Bolognesi');
insert into artista values('asdfgf','Massimo','Cancellieri');

insert into partecipanti(evento,artista)
values('as123432','123');
insert into partecipanti(evento,artista)
values('as123432','asdf');
insert into partecipanti(evento,artista)
values('as123432','asdfgf');

insert into settore values (
1,
'as123432',
'galleria',
'posti in piedi',
300);

insert into settore values(
2,
'as123432',
'lodi',
'posti numerati',
50);

insert into settore values(
3,
'as123432',
'milano',
'posti non numerati',
100);

insert into settore values (
1,
'pp129438',
'galleria',
'posti in piedi',
300);

insert into settore values(
2,
'pp129438',
'lodi',
'posti numerati',
50);

insert into settore values(
3,
'pp129438',
'milano',
'posti non numerati',
100);

insert into biglietto values(
'sdfgfd23',
'11/08/2015',
'30.50',
'as123432',
3,
null);

insert into biglietto values(
'asdwer23',
'12/08/2015',
'30.50',
'as123432',
```

```
3,
null);

insert into biglietto values(
'aaasdf23',
'11/08/2015',
'0',
'pp129438',
'3',
null);
```

**Progettazione logica applicazione JSF**

La navigazione comincia nella pagina index.jsf, dove si riempie nel Managed Bean *home* (alias per EventiView.java) l'attributo *eventi* con gli eventi del giorno, se ce ne sono (i veda il metodo cercaEventi presente nella classe database).
Successivamente una datatable disporrà in un elenco gli eventuali eventi della giornata o cmq quelli che risiedono in *home* nell'attributo *eventi*.
Il nome è un collegamento alla descrizione (evento.jsf), in particolare registra in *home* l'evento selezionato. Nella pagina evento, le voci di descrizione sono settate in base alla tipologia di eventi (metodo infoEvento contenuto in *home*).
L'utente può cercare (anche lasciando campi vuoti) un evento impostando il tipo, o la data, o entrambi. Queste scelte vanno a popolare gli attributi di *home* e poi si viene redirezionati in lista-eventi.jsf, in cui con il metodo cercaEventi viene settato la lista eventi e poi stampata in un datatable. Se la lista è vuota viene allertato l'utente.
Sia nell'index che in lista-eventi si può procedere all'acquisto di un biglietto per un dato evento, venendo dirottati alla pagina biglietto.jsf. Questa pagina recupera dati sia da *home* che dalll'altro Managed Bean *tk*. In particolare, ricava da *home* l'evento di interesse e setterà gli attributi nome,cognome, via, cartacredito con i vari campi con i dati utente. Da lì si può procedere all'acquisto effettivo o tornare indietro nelle pagine precedentemente spiegate.
In particolare la scelta del settore e del posto sono a completamento successivo, nel senso che solo quando l'utente avrà selezionato da una tendina il tipo di settore apparirà una seconda tendina per la scelta del numero di settore, e così per il posto numerato eventualmente.
Questa pagina richiama diversi metodi che eseguono query sul database. Questi sono presenti in *tk*, ma a loro volta richiamano metodi presenti nella classe *database*.
Se tutti i dati utente sono settati e se la scelta del settore/posto è completa, si procede all'acquisto nella pagina acquisto.jsf, che richiamando attributi presenti in *tk* mostra un resoconto dei dati di acquisto (utente ed evento) e del numero di posti occupati per settore.

## Struttura dell'applicazione web

I principali file dell'applicazione sono distribuiti in tre cartelle differenti, a simboleggiare package e funzionalità differenti: all'interno di src/java troveremo la cartelle database, che contiene classi bean e la classe database che contiene i metodi di interrogazione postgrelsql. Sempre in questo percorso ma nella cartella view risiedono i due Managed Beans contenuti nelle classi BigliettoView ed EventiView, che si occupano di memorizzare dati e di gestire la comunicazione bidirezionale con le classi contenute in database, che a loro volta interagiscono con Postgrelsql.
Un'ultima locazione è la cartella web, in cui sono situate le pagine dinamiche e l'intero pachetto “Tigra Calendar”.