Applicazione Back-Office 'Event Manager'

Introduzione:

(Event Manager) è un Sistema Informativo complesso e distribuito finalizzato a gestire eventi che coinvolgono una grande partecipazione di pubblico, quali concerti, cinema, teatri, conferenze, etc..
Il sistema distribuito presenta una parte di Back-Office per la gestione degli eventi da parte degli amministratori, un Front-End per l'acquisto di un biglietto di un evento da parte di un utente finale, ed un client su dispositivo mobile, utilizzato dai Controllori per verificare la validità degli accessi.

La nostra applicazione di Back-Office, prevederà :

- **L'inserimento di un evento,** e dei suoi relativi dati
- **La visualizzazione degli eventi** inseriti
- **L'eliminazione di evento**, qualora non ci fossero biglietti venduti
- **La visualizzazione clienti** iscritti e i relativi dati
- **L'eliminazione di un client**e
- **La visualizzazione di statistiche** relative agli eventi
- **La creazione di utenze degli addetti** alla sicurezza
- **La modifica e l'eliminazione degli addetti** alla sicurezza

Analisi e Specifica dei Requisiti

Abbiamo deciso di dividere la progettazione in 3 macro-categorie, gestite ognuna da team o persone diverse:

- **Gestione degli Eventi**
- **Gestione dei Clienti**
- **Gestione degli Addetti**

Le Statistiche invece sono state trattate in modo diverso, poiché abbiamo delle statistiche sia per gli Eventi sia per i Clienti, nei prossimi documenti vengono trattate più nel dettaglio.

Modello Funzionale

Gestione Eventi

Descriviamo le interazioni con il sistema da parte degli **Impiegati** e gli **Amministratori** (Vedere Glossario), con un Use Case di UML 2.0:

L'Impiegato del Back-Office, quindi può inserire un evento con i suoi dati, cercare la lista degli eventi, in cui si possono compiere le seguente azioni sulla lista**: Visualizzazione dei Dati, Modifica dei Dati, Eliminazione dell'Evento.**

La V**isualizzazione dei Dati** mostrerà a schermo tutti i dati inseriti dall'impiegato oppure dall'amministratore del sistema.

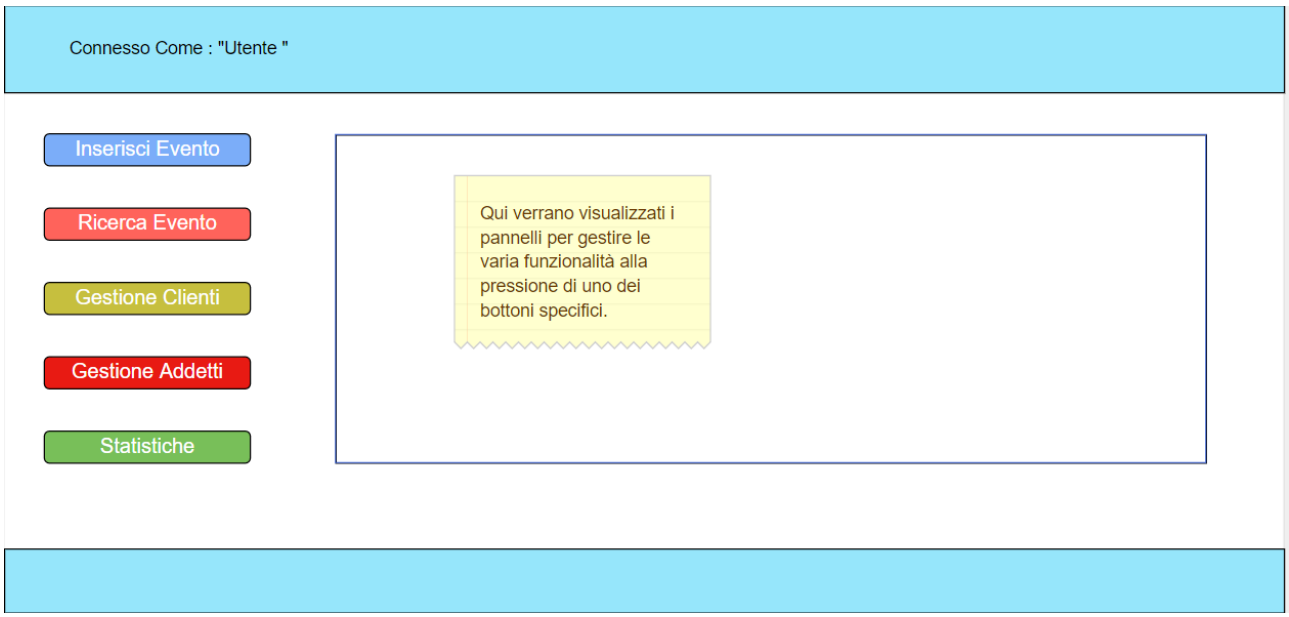
La **Modifica dei Dati,** invece prevede la modifica dei dati salienti di un Evento, i quali sono **Data e Luogo,** dati concordati con il Committente, poiché la modifica di altri dati porterebbe ad uno stravolgimento dell'evento stesso.

L'**Eliminazione dell'Evento,** potrà essere effettuato solamente se non sono stati ancora venduti biglietti per quello specifico evento selezionato.

CockBurn:

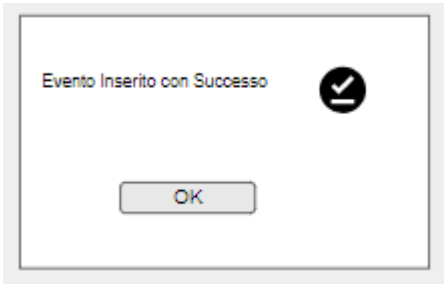
Procediamo ora con le tabelle di CockBurn per mostrare i passi tra utente e sistema, e in che modo interagiscono tra loro.
Mostreremo i Mock-Up per aiutarci a descrivere meglio queste interazioni:

MockUp: MockUp_MenuPrincipale



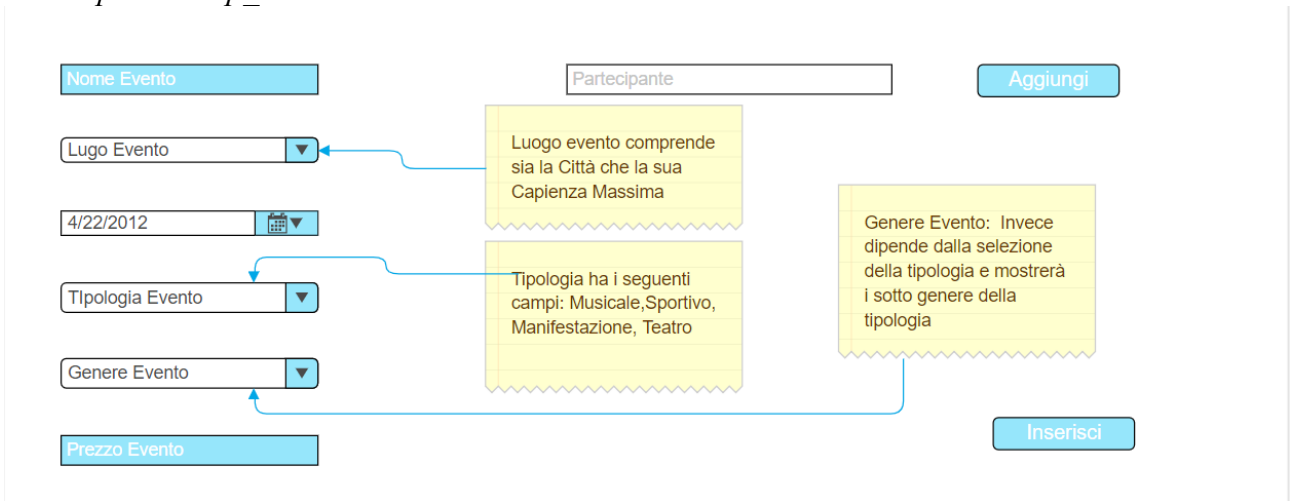
MockUp: MockUp_ErroreDati

MockUp: MockUp_SuccessoInserimento



NOTA: I MockUP di errore e di successo sono uguali per tutto il sistema, quello che cambierà sarà solo il testo al suo intero, abbiamo quindi deciso di rappresentarli solo due, che faranno da esempi

MockUp: MockUp_InserisciEvento



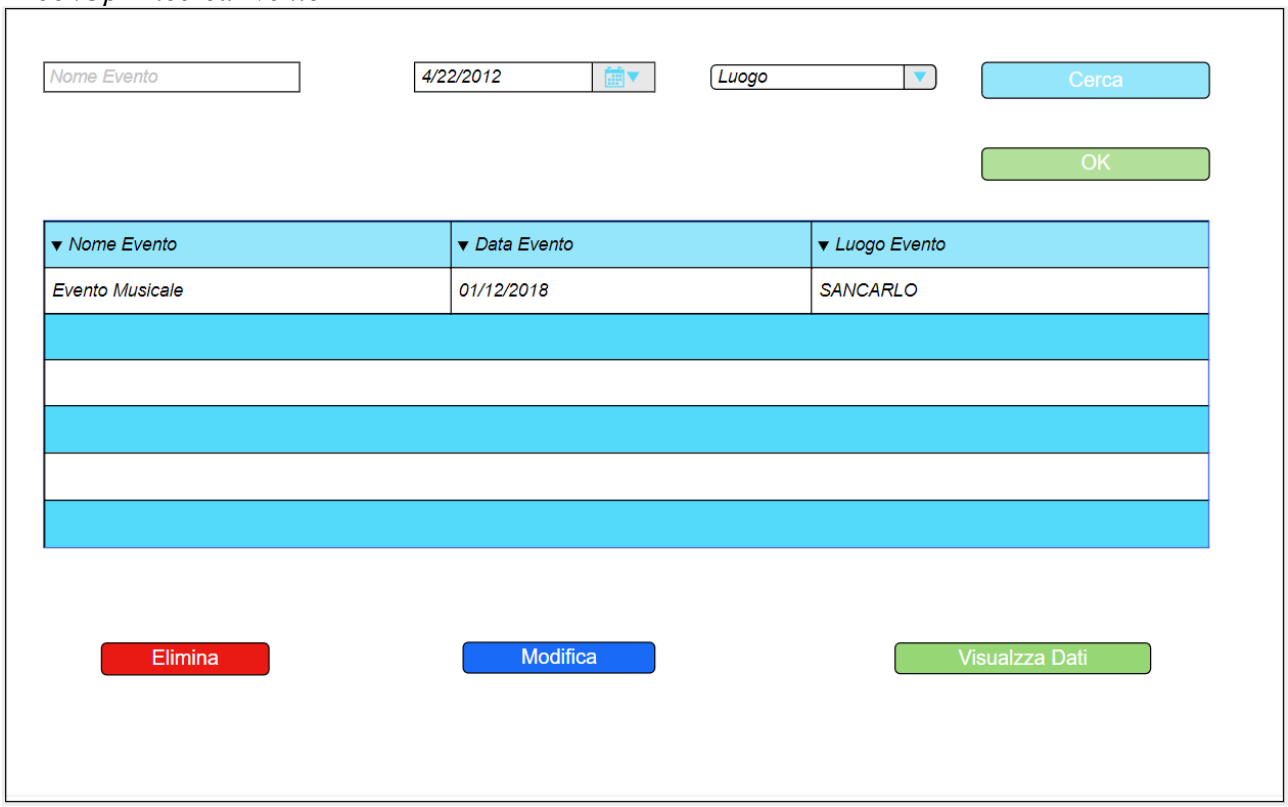
CockBurn per lo Use Case : Inserimento Evento

Use Case: #1	*Inserimento Evento*		

Goals Context	Un Impiegato vuole inserire un Evento nel sistema		
Scope & Level			
Preconditions	L'utente deve essere un **Impiegato BackOffice** o **Amministratore**		
Success End Condition	L'Impiegato/Amminsitratore riesce ad inserire correttamente l'evento		
Failed End Condition	Dati errati o utente non è del BackOffice		
Primary Actor	**Impiegato BackOffice**		
Trigger	L'utente preme il pulsate 'Inserisci Evento', nel *MockUp_MenuPrincipale*		
DESCRIPTION	**Step n°**	**ImpiegatoBackOffice**	**Sistema**
	1	Utente preme il pulsante 'Inserisci Evento' Nel *MockUp_MenuPrincipale*	
	2		Apre la schermata di inserimento : *MockUp_InserimentoEvento*
	3	Inserisce i campi dell'evento	
	4		Mostra *MockUp_SuccessoInserimento*
SUBVARIATIONS	*Step*	**ImpiegatoBackOffice**	**Sistema**
	3.1	Utente non inserisce i dati, inseriti in modo	

		errato oppure evento è stato già creato	
			Mostra MockU_ErroreDati

MockUp RicercaEvento



Verranno visualizzati gli eventi precedentemente inseriti, possiamo scegliere di inserire un campo o più campi per rendere la ricerca più specifica, come si vede i tre pulsati sotto, permettono di effettuare le tre funzionalità descritte precedentemente.

CockBurn per l'Use Case : 'Ricerca Evento'

USE CASE #2	Ricerca Evento		
Goals Context	Un Impiegato vuole visualizzare gli eventi precedentemente inserimenti		
Scope & Level			
Preconditions	L'utente deve essere loggato nel sistema		
Success End Condition	La lista viene visualizzata		

	correttamente		
Failed End Condition			
Primary Actor	Impiegato BackOffice		
Trigger	L'utente preme il pulsante 'Ricerca Evento' nel menu		
DESCRIPTION	**Step n°**	**Impiegato BackOffice**	**Sistema**
	1	Preme pulsate 'Ricerca Evento'	
	2		Mostra *MockUp_RicercaEvento*
	3	Preme tasto 'Cerca'	
	4		Mostra la lista di tutti gli eventi
SUBVARIATIONS	**Step n°**		
	1	Preme pulsanete 'Ricerca Eventi'	
	2		Mostra *MockUp_RicercaEvento*
	3,1	Inserisce 'Nome Evento' nell'apposito campo	
	4,1		Mostra la lista degli eventi che rientrano nei critieri di ricerca per nome.
	Step n°		
	1	Preme pulsanete 'Ricerca Eventi'	
	2		Mostra *MockUp_RicercaEvento*
	3,2	Seleziona una Data Evento nel DatePicker	
	4,2		Mostra la lista degli eventi che ci sono in quella specifica data
	Step n°		
	1	Preme pulsanete 'Ricerca Eventi'	
	2		Mostra *MockUp_RicercaEvento*

	3,3	Seleziona un Luogo Evento nell'apposita ComboBox	
	4,3		Mostra la lista degli eventi che ci sono nel luogo selezionato
	Step n°		
	1	Preme pulsanete 'Ricerca Eventi'	
	2		Mostra *MockUp_RicercaEvento*
	3,1	Compila tutti e 3 i campi: Nome Evento,Data Evento e Luogo Evento	
	4,1		Mostra la lista degli eventi che rientrano nei parametri inseriti

NOTA: Possiamo scegliere qualsiasi combinazione di dati per la ricerca.

CockBurn per l'Use Case: 'Modifica Evento'

USE CASE #3	Modifica Evento		
Goals Context	Un Impiegato vuole modificare la data e/o luogo di evento selezionato, precedentemente visualizzato grazie alle ricerca		
Scope & Level			
Preconditions	L'utente deve essere loggato nel sistema, e deve avere effettuato la Ricerca un evento. Vedere USE CASE#2		
Success End Condition	L'Evento viene modificato		

	correttamente		
Failed End Condition			
Primary Actor	Impiegato BackOffice		
Trigger	L'utente preme il pulsante 'Modifca Evento' nel MockUp_RicercaEvento		
DESCRIPTION	**Step n°**	**Impiegato BackOffice**	**Sistema**
	1	Seleziona Evento	
	2	Preme pulsate 'Modifica Evento'	
	3		*Disattiva tutti i pulsati e gli elementi tranne : pulsate 'OK', DataEvento (DatePicker) e LuogoEvento(ComboBox)*
	4	Selezionata una data e un luogo	
	5	Preme Pulsante 'OK'	
	6		Mostra '*MockUp_Successo'*
SUBVARIATIONS 'Dati errati'	**Step n°**		
	1	Seleziona Evento	
	2	Preme pulsate 'Modifica Evento'	
	3		*Disattiva tutti i pulsati e gli elementi tranne : pulsate 'OK', DataEvento (DatePicker) e LuogoEvento(ComboBox)*
	4	Selezionata una data e un luogo	
	5	Preme Pulsante 'OK'	
	6		Mostra 'MockUp_ErroreDati'

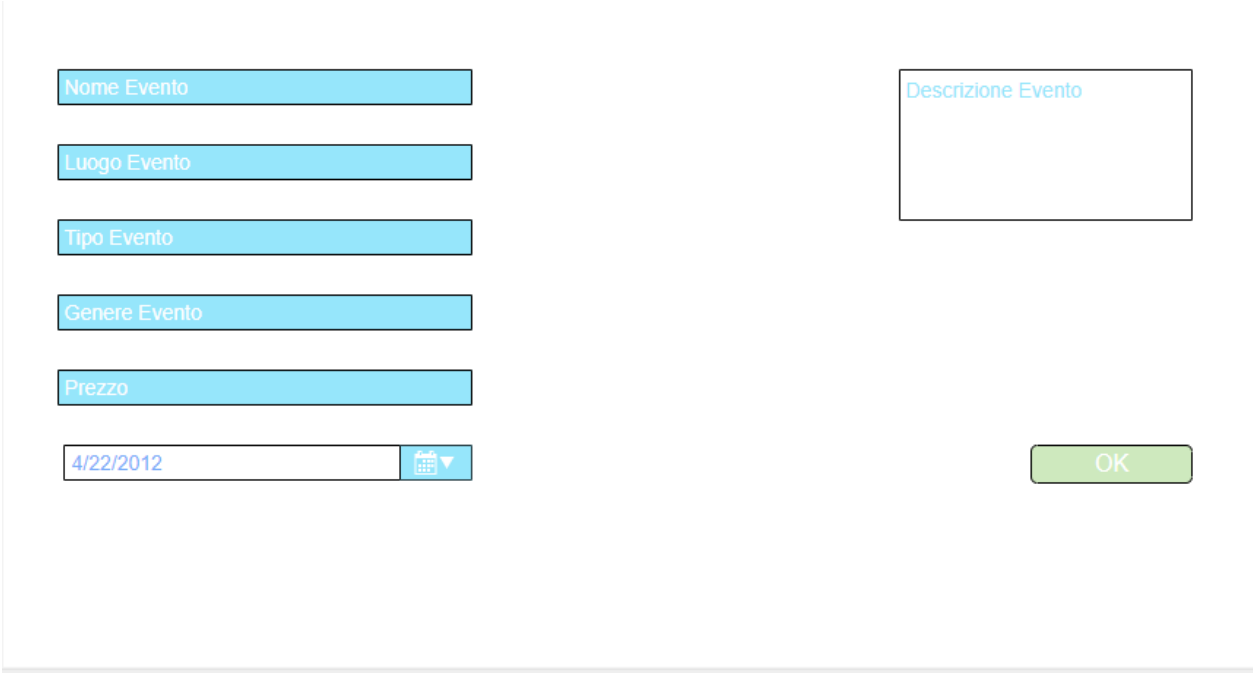
CockBurn per l'Use Case: ' Eliminazione Evento'

USE CASE #4	Elimazione Evento		

Goals Context	L'impiegato vuole eliminare un evento selezionato vedere USE CASE #2		
Scope & Level			
Preconditions	L'impiegato deve essere loggato e deve essere stata effettuata la ricerca di evento/i (Vedere USE CASE #2)		
Success End Condition	L'evento selezionato viene elimanto con successo		
Failed End Condition	L'evento non viene eliminato perché ha dei biglietti venduti		
Primary Actor	Impiegato BackOffice		
Trigger	L'impiegato preme il pulsate 'Elimina' nel *MockUp_RicercaEvento*		
DESCRIPTION	**Step n°**	**Impiegato BackOffice**	**Sistema**
	1	Seleziona Evento	
	2	Preme il pulsante 'Elimina Evento'	
	3		Controlla se non sono stati venduti biglietti
	4		Elimina evento.
	5		Mostra *MockUP_SucessoEliminazione*
	6	Preme 'OK' nel *MockUP_SuccessoEliminazione*	
SUBVARIATIONS 'Biglietti eventi selezionati >0'			
	1	Seleziona Evento	
	2	Preme il pulsate 'Elimina Evento'	
	3		Controlla se non stati venduti biglietti

	4		Mostra *'MockUp_ErroreEliminazione'*
	5	Preme 'OK' nell *'MockUp_ErroreEliminazione'*	

MockUP: 'Visualizza Dati Evento'



Qui verranno visualizzati i dati relativi all'evento selezionato, i campi saranno disabilitati per le modifiche, il tasto 'OK' porterà alla schermata precedente, *(MockUP_RicercaEvento).*

CockBurn per l'Use Case : ' Visualizza Dati Evento'

USE CASE #5	Visualizza Dati Evento		
Goals Context	L'impiegato visualizza i dati dell'evento selezionato		
Scope & Level			
Preconditions	L'impiegato deve essere loggato e deve essere stata effettuata la ricerca di evento/i, e deve selezionare un		

	evento (Vedere USE CASE #2)		
Success End Condition	L'evento selezionato viene elimanto con successo		
Failed End Condition	Non e' stato selezionato nessun evento dall'Impiegato		
Primary Actor	Impiegato BackOffice		
Trigger	L'impiegato preme il pulsate *'VisualizzaDati Evento'*		
DESCRIPTION	**Step n°**	**Impiegato BackOffice**	**Sistema**
	1	Seleziona Evento	
	2	Preme 'Visualizza Dati Evento' (MockUP_RicercaEvento)	
	3		Mostra MockUp_VisualizzaDatiEvento
	4	Preme pulsate 'OK'	
	5		Mostra 'MockUp_RicercaEvento

Glossario:

- **Impiegato BackOffice:** E' l'utente che lavora nel BackOffice della Società, il suo compito è quello di gestire gli eventi, effettuare le operazioni CRUD su di essi, e sugli Addetti alla sicurezza.Questo utente potrà creare la sua utenza per accedere al gestionale sul sito della piattaforma.
- **Amministratore :** Ha gli stessi privilegi dell'Impiegato BackOffice, ma può visualizzare i dati dei Clienti e cancellare quando necessario.
-

Modelli di Dominio

In questa sezione analizzeremo i modelli del dominio, estendendo le nozioni acquisite dalla analisi dei requisiti, utilizziamo sempre UML 2.0, questa volta usermo i **Class Diagram, Sequence Diagram, Activity Diagram.**

In questa fase si cercherà di individuare gli oggetti software necessari per modellare con più dettaglio quello descritto nella sezione di Analisi, per fare questo abbiamo usato l'Euristica Three-Object (Entiry,Control,Buondary), seguendo le sequenti regole:

1. Gli oggetti *Buondary* parlando solo con gli oggetti *Control,* e gli *Attori* (Impiegato BackOffice)
2. Gli oggetti *Entity* parlano solo con gli oggetti *Control.*
3. Gli oggetti *Control,* parlano con tutti tranne che con gli *Attori.*

NOTA: I getter e setter nei Class Diagram, sono stati omessi, per rendere il diagramma più leggibile

Inserimento Evento

Dalla fase di analisi, e dalla modellazione dei MockUp, abbiamo trovato i primi potenziali oggetti software, quali '*Evento'* e il form per inserire un evento, 'InserisciEventoForm', possiamo dunque definire '*Evento'* come un oggetto **Entity**, e '*InserisciEventoForm*', come un oggetto **Buondary**, notando che questi due tipi di oggetto non possono comunicare tra loro, se vogliamo seguire la *best practice* Three-Object, dovremo utilizzare un'altra classe che comunica sia con l'oggetto **Entity** sia con l'oggetto **Buondary**, un oggetto di tipo **Control**, è quello che stiamo cercando, dunque, questa classe che chiameremo *'InserisciEventoControl',* che comunicherà con entrambi gli oggetti.
A questo punto, vorremmo salvare questi dati in una base di dati, non bene specifica per ora, quindi useremo il ***Design Pattern DAO***, per comunicare con una base di dati, e in questo modo separeremo i vari livelli di astrazione, rendendo il software più mantenibile e più propenso ai cambiamenti.
Ci troviamo dunque con un altro oggetto che non fa parte della categoria che abbiamo prima elencato, ma sappiamo che i **Control**, possono comunicare con tutti i tipi di classe, e quindi facciamo comunicare proprio a quest'ultimo con la nostra Classe DAO, che sarà un interfaccia, che sarà implementata da classi concrete a seconda dell'implementazione della base di dati scelta.
Sommario delle classe usate:

- *MenuPrincipaleForm:* Questo oggetti di tipo **Buondary,** rappresenta il *MockUp_MenuPrincipale,* dovrà creare i vari form che serviranno a compiere le varie funzionalità del sistema.
- *InserisciEventoForm:* Classe di tipo **Buondary**, la quale prevederà i campi necessari per l'inserimento dei dati di un evento, con questa classe interagirà l'**Attore**, che nel nostro caso è l'impiegato backoffice, questa classe una volta raccolti i dati, prevederà ad inviarli al **Control**.
- *InserisciEventoControl:*Classe **Control**, controllerà la correttezza dei dati secondo i vincoli che verranno successivamente descritti in dettaglio, e creerà l'evento con i dati correnti, comunicando con la classe **DAO**.
- *EventoDAO:* Classe **DAO** comunica solo che il **Control** e l'implementazione della basi di dati scelta.
- *Evento:* Classe **Entity** modella l'evento da inserire.

Abbiamo aggiunto delle enumerazioni per i tipo di dato con i dominio limitato:

- *LuogoEnum:* Rappresenta il luogo dell'evento, conterrà : 'Nome Luogo', 'Città Luogo', 'Capienza Massima'.
- *CittaEnum:* Presente in *LuogoEnum* sarà l'elenco di città su cui lavorerà la piattaforma
- *TipologiaEnum:* Elenco delle quattro tipologie di eventi disponibili sulla piattaforma, i quali ogni elemento di questa enumerazione avrà la sua enumerazione di sotto generi.

Dal Sequence Diagram abbiamo quindi trovato le principali classi che dovranno essere implementate e come comunicano tra loro, ora vedremo la rappresentazione statica di questi oggetti con un **Class Diagram.**

Sono state aggiunte le classi *enumerazioni*, e le relazioni con le classi.
Vincoli OCL :

- context EVENTO inv : DataEvento >= Today.Date
- context EVENTO inv : NomeEvento.lenght()>=8
- context EVENTO inv : Prezzo > 0.0
- context EVENTO inv : Partecipanti->size() > 0
- context EVENTO inv : TipologiaEvento
  if Evento.TipologiaEvento = MUSICALE
  then Evento.GenereEvento = (ROCK or POP or JAZZ or CLASSICA)
  if Evento.TipologiaEvento = TEATRO
  then Evento.GenereEvento = (DRAMA or MUSICAL or COMMEDIA or CABARET or OPERA)
  if Evento.TipologiaEvento = SPORTIVO
  then Evento.GenereSportivo = (CALCIO or BASKET or GOLF or NUOTO or RUGBY or CICLISMO or TENNIS or ATLETICA)
  if Evento.TipologiaEvento = MANIFESRAZIONE
  then Evento.GenereEvento = (FIERA or CIRCO or PARCO or ALTRO).

Oltre questi vincoli, tutti gli attributi dell'evento devono essere diversi da NULL, fatta eccezione per la descrizione, la quale è opzionale, e la DataEvento e LuogoEvento non devono esistere nella lista degli eventi già presenti.
Abbiamo scelto di rendere *GenereEvento* di tipo String, per via delle 4 enumerazioni possibili.

Ricerca Evento:

Per questa funzionalità abbiamo usato la stessa tecnica del *Three-Object-Type*, questa funzionalità comprenderà anche la: *Visualizzazione dei dati di un Evento, Eliminazione Evento, Modifica Evento*.
Fa riferimento quindi al *MockUp_RicercaEvento.*
Il Sequence Diagram, ci sarà utile per identificare una prima versione degli oggetti necessari.
Sommario classi usate:

- *RicercaEventoControl:* Questo oggetti di tipo **Control**, creare il form per la ricerca evento, e comunicherà con le classi DAO per raccogliere i dati della ricerca.
- *RicercaEventoForm:* Ci permetterà di 'leggere' le azioni dell' **Attore**, questa classe **Buondary**, ci permetterà di effettuare le tre funzionalità legate alla ricerca di evento(Visualizzazione,Modifica ed Eliminazione di un Evento).
- *ListEventiForm:* visualizzerà la lista di eventi ricercati, e conterrà il riferimento all'evento.

Vincoli:

- Prima di poter modificare, eliminare o visualizzare un evento, l'Attore deve averlo selezionato dall apposita lista di eventi.
- Per quanto riguarda la modifica, i dati seguendo i vincoli dell'inserimento, per i campi DataEvento e LuogoEvento.

Rappresentiamo ora le classi in un Class Diagram

Proseguiamo con i Sequence Diagram e Activity Diagram per le altre tre funzionalità: