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** fin ora inseriti
- **L'eliminazione di evento**, qualora non ci fossero biglietti venduti
- **La visualizzazione clienti** iscritti e i relativi dati
- **L'eliminazione di un cliente**
- **La visualizzazione di statistiche** relative agli eventi
- **La creazione di utenze degli addetti** alla sicurezza
- **La visualizzazione degli addetti** alla sicurezza presenti nel sistema
- **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:

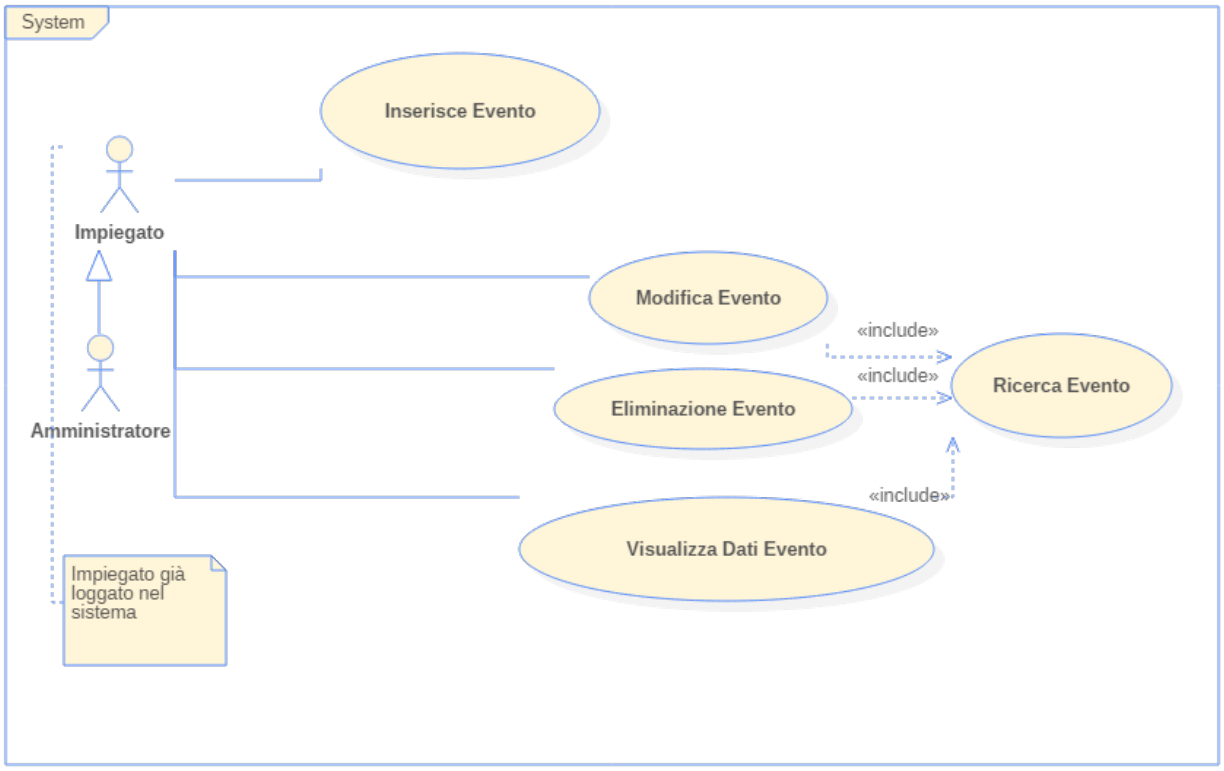
- **Gestione degli Eventi**
- **Gestione dei Clienti**
- **Gestione degli Addetti**

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

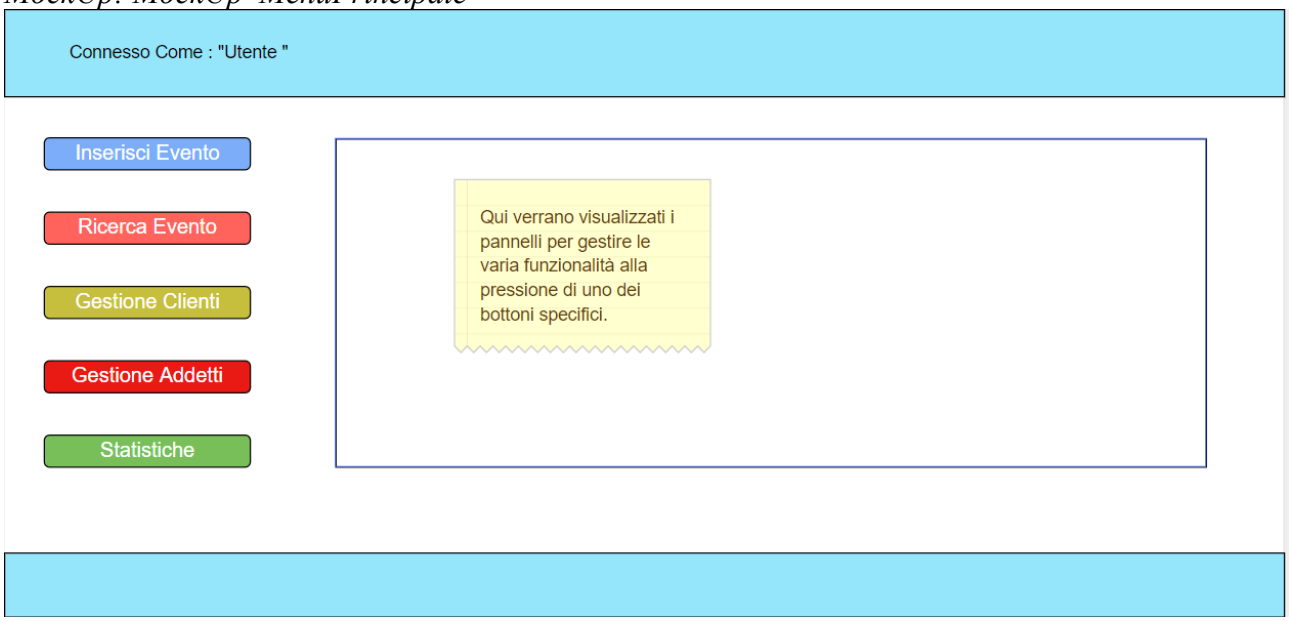
La **Modifica dei Dati**, invece prevede la modifica dei dati salienti di un Evento, i quali sono **Data** e **Luogo**, dati concordati con il Committente, poiché la modifica di altri dati porterebbe ad uno stravolgimento dell'evento stesso.

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

CockBurn:

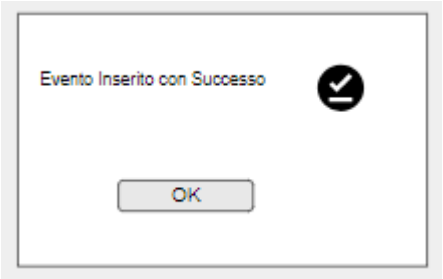
Procediamo ora con le tabelle di CockBurn per mostrare i passi tra utente e sistema, e in che modo interagiscono tra loro.
Mostreremo i Mock-Up per aiutarci a descrivere meglio queste interazioni:

MockUp: MockUp_MenuPrincipale

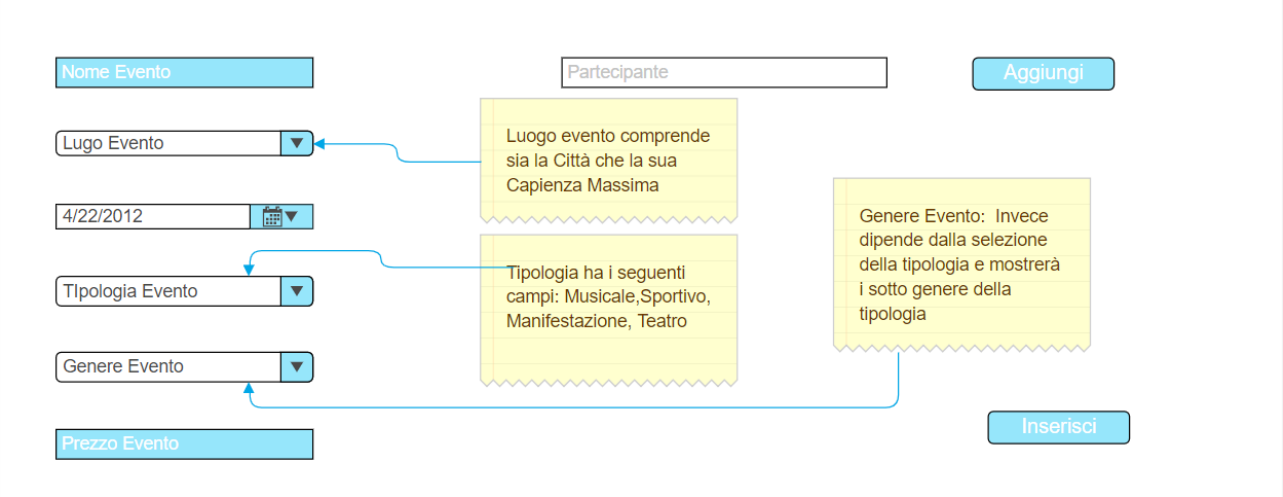


MockUp: MockUp_ErroreDati

MockUp: MockUp_SuccessoInserimento



MockUp: MockUp_InserisciEvento

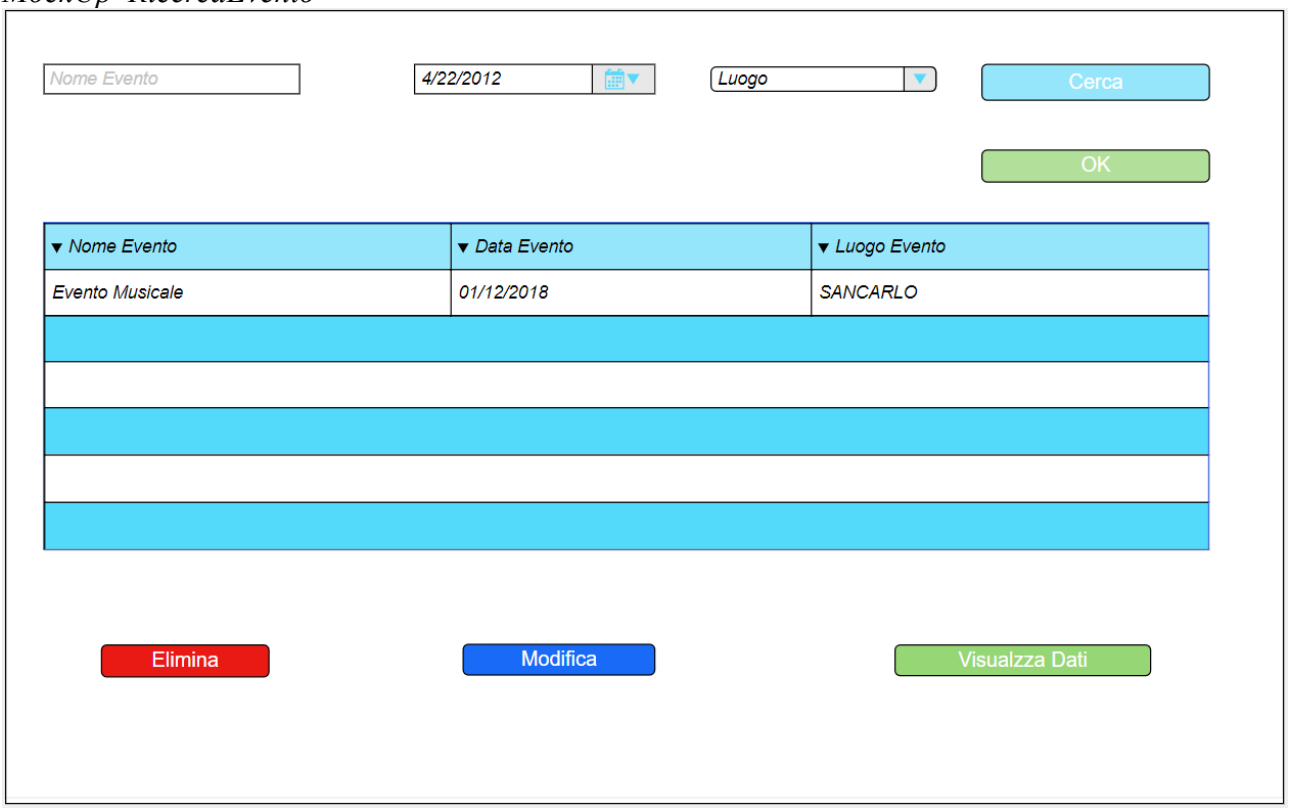


CockBurn per lo Use Case : Inserimento Evento

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*
SUBVARIATION #1	*Step*	**ImpiegatoBackOffice**	**Sistema**
	3.1	Utente non inserisce i dati, inseriti in modo errato oppure evento è stato già creato	
			Mostra MockUp_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
SUBVARIATION #1	**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.

SUBVARIATION #2	**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
SUBVARIATION #3	**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
SUBVARIATION #4	**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	

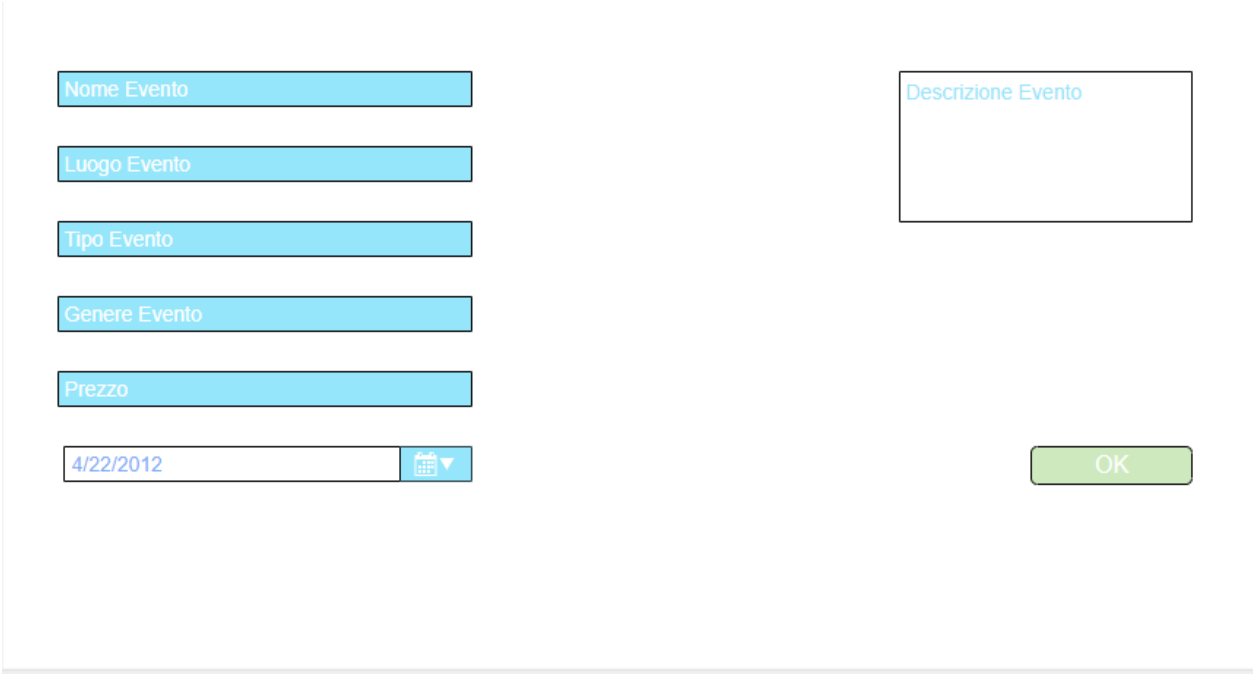
	2	Preme pulsate 'Modifica Evento'	
	3		*Disattiva tutti i pulsati e gli elementi tranne : pulsate 'OK', DataEvento (DatePicker) e LuogoEvento(ComboBox)*
	4	Selezionata una data e un luogo	
	5	Preme Pulsante 'OK'	
	6		Mostra 'MockUp_ErroreDati'

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'



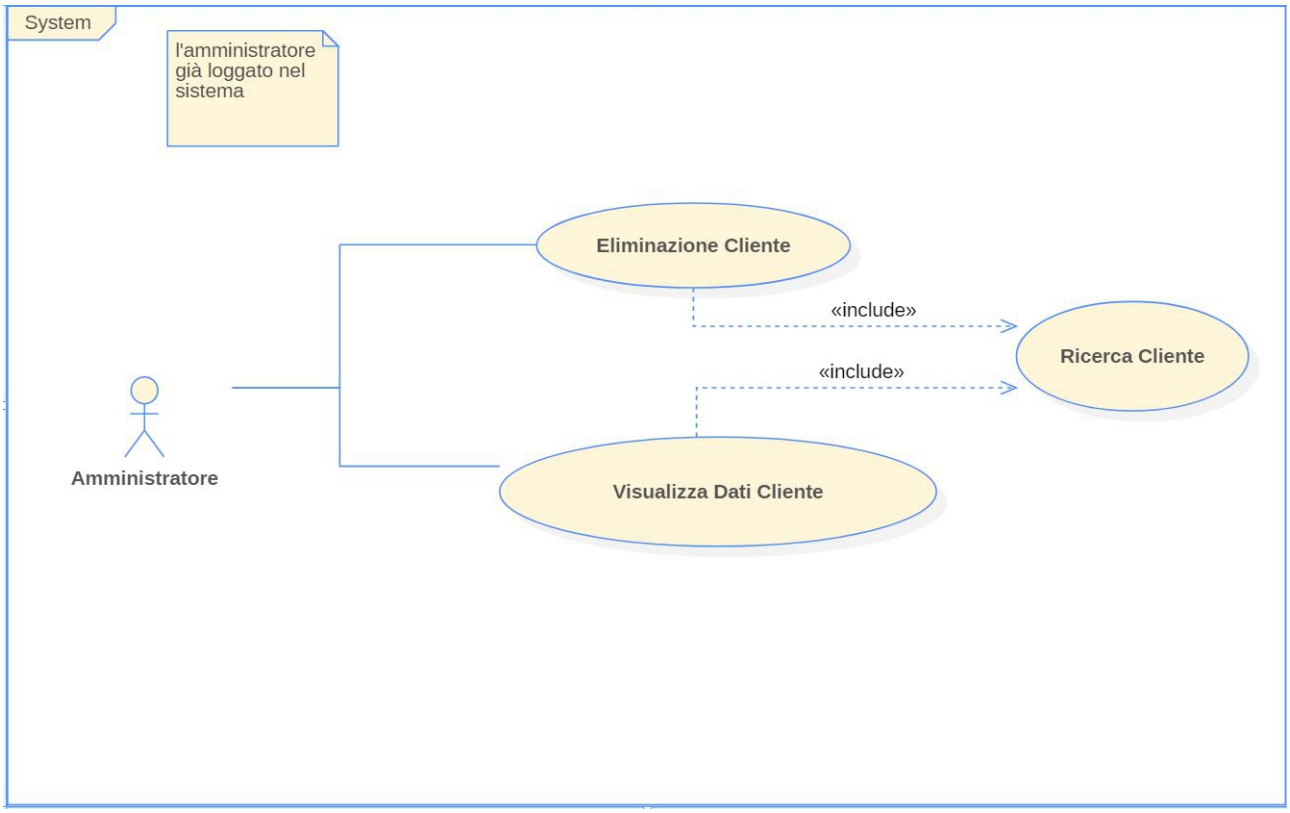
Qui verranno visualizzati i dati relativi all'evento selezionato, i campi saranno disabilitati per le modifiche, il tasto 'OK' porterà alla schermata precedente, *(MockUP_RicercaEvento).*

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

Gestione Clienti

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

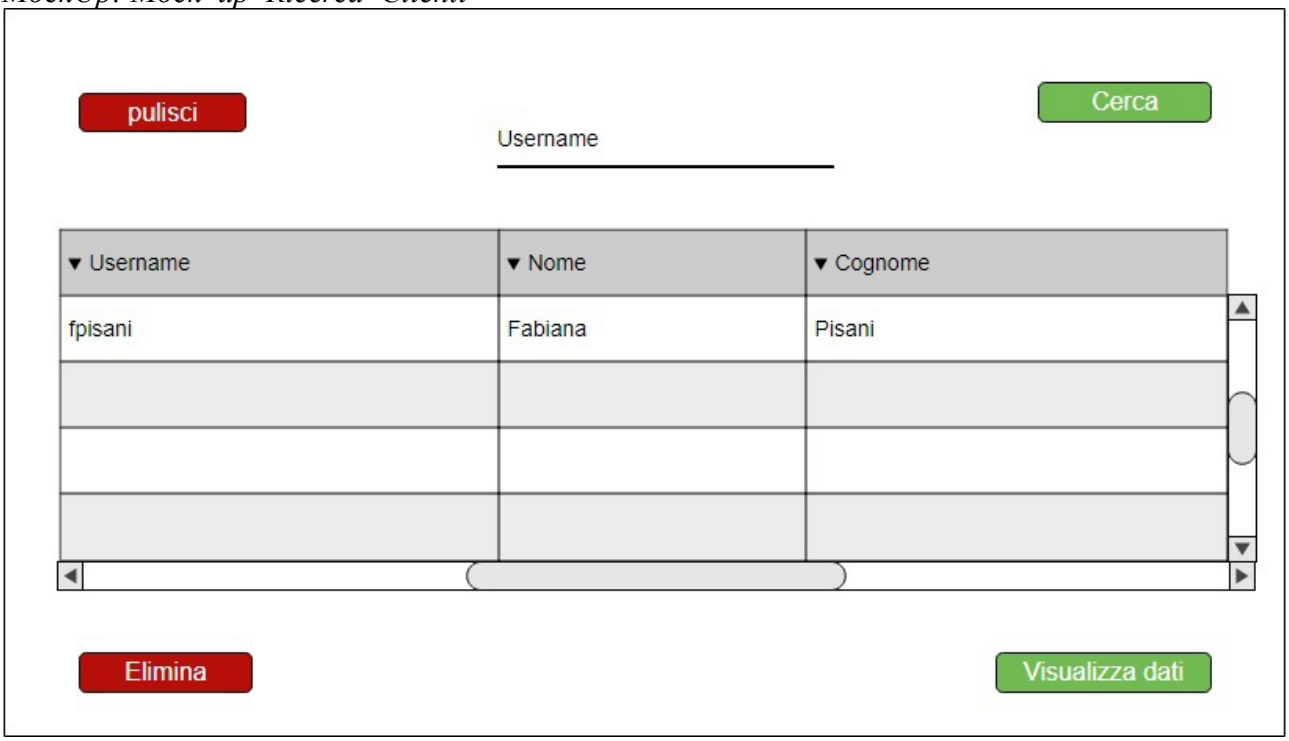


L'Amministratore del Back-Office, può cercare la lista dei clienti, in cui si possono compiere le seguenti azioni sulla lista**: Visualizzazione dei Dati di un cliente o eliminazione del cliente.**

La V**isualizzazione dei Dati di un cliente** mostrerà a schermo tutti i dati inseriti dall'utente in fase di registrazione su un altro sistema con annesso grafico a torta, la cui compilazione è composta dal numero di biglietti acquistati in base al tipo di evento questo per una statistica di gradimento per i tipi di evento.

L'**Eliminazione del cliente,** consiste nella cancellazione dal database di un cliente ed è per questo motivo che solo l'amministrazione può effettuare questa operazione.

Cockburn:
MockUp: Mock_up_Ricerca_Clienti



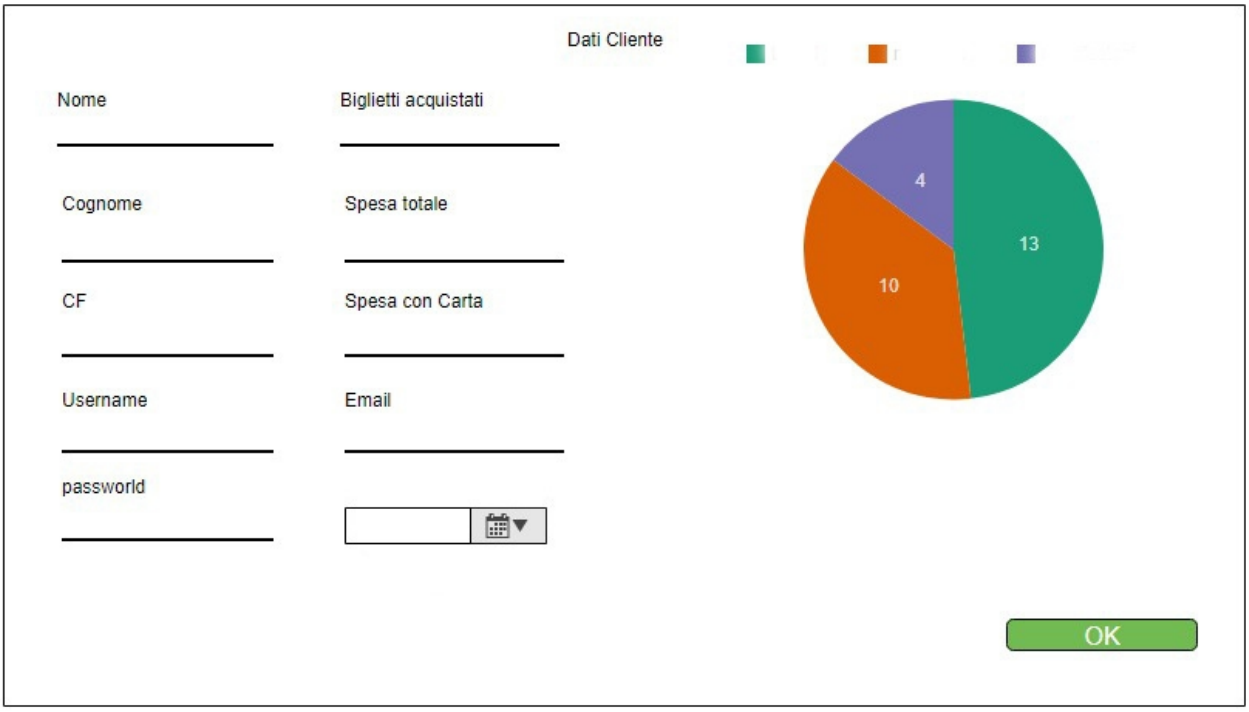
Verranno visualizzati i Clienti precedentemente inseriti, possiamo scegliere di inserire un username per rendere la ricerca più specifica. Come si vede, i tre pulsanti sotto permettono di effettuare le tre funzionalità descritte precedentemente.

CockBurn per l'Use Case : Ricerca Cliente

USE CASE *#1*	*Ricerca cliente*		
Goal in Context	L'amministratore vuole visionare i dati di un cliente		
Scope & Level			
Preconditions	L'amministratore deve essere loggato nel sistema		
Success End Condition	La lista viene visualizzata correttamente		
Failed End Condition			
Primary Actor	Amministratore BackOffice		
Trigger	Amministratore preme Gestione Clienti dal menu principale		
DESCRIPTION	**Step n°**	**Attore 1**	**Sistema**
	1	Preme il bottone Ricerca Clienti	
	2		*Mostra schermata gestione_clienti*
	3	Preme il bottone Cerca	
	4		*Compila la tabella con la lista*

SUBVARIATIONS	*Step*	**Attore 1**	**Sistema**
	3.1	Inserisce 'Username' nell'apposito campo	
	4.1		Mostra la lista dei clienti che rientrano nei critieri di ricerca per username

MockUp: Mock_up_Visualizza_dati_Cliente



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

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

USE CASE #2	Visualizza dati cliente
Goal in Context	L'amministratore vuole visionare i dati di un cliente
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 #1)
Success End Condition	L'amministratore prende visione dei dati di un cliente
Failed End Condition	Il cliente non esiste
Primary Actor	Amministratore back office

Trigger	Accedere al sistema		
DESCRIPTION	**Step n°**	**Attore 1**	**Sistema**
	1	Seleziona il cliente	
	2	Preme 'Visualizza Dati' (MockUP_RicercaCliente)	
	3		Mostra *Mock_up_Visualizza_dati_Cliente*
	4		Compila le caselle e crea il grafico con i dati del database riguardanti il cliente
	5	Preme il bottone ok	

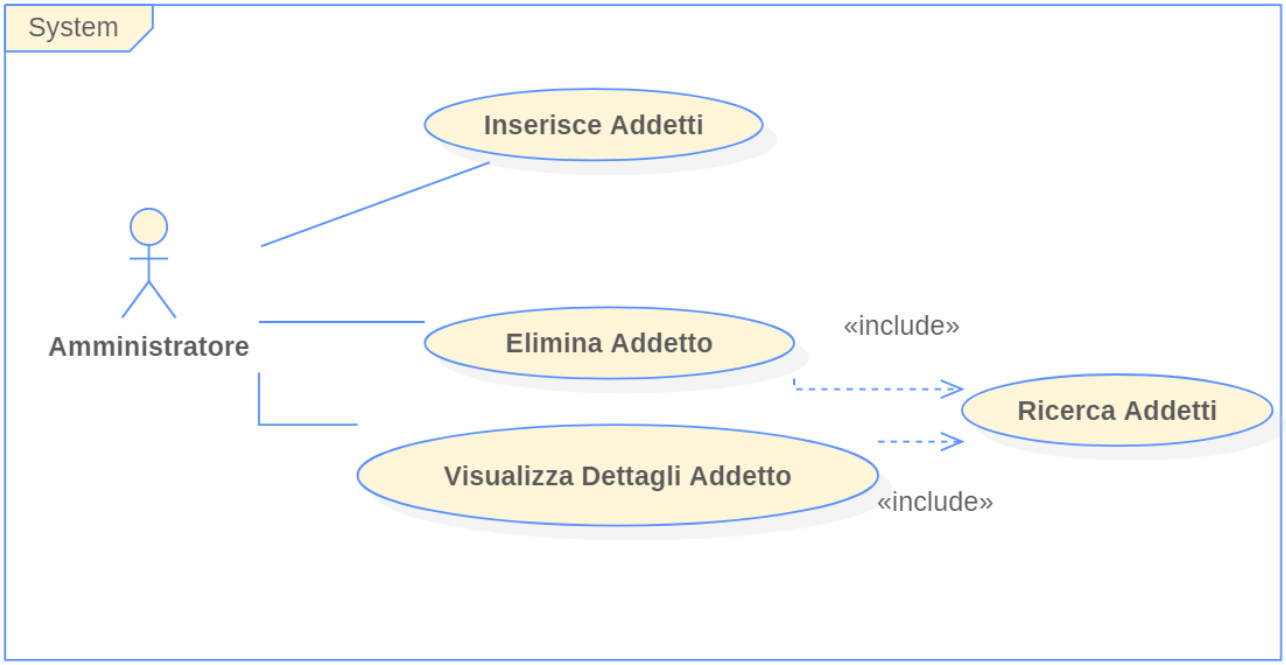
CockBurn per l'Use Case: ' Elimina Cliente'

MockUp: Mock_up_Elimina_Cliente

USE CASE *#x*	Elimina Cliente		
Goal in Context	L'impiegato vuole eliminare un Cliente selezionato vedere USE CASE #1		
Scope & Level			
Preconditions	L'impiegato deve essere loggato e deve essere stata effettuata la ricerca di un Cliente (Vedere USE CASE #1)		
Success End Condition	Il Cliente è stato eliminato		
Failed End Condition	Il cliente non esiste		
Primary Actor	Amministratore back office		
Trigger	L'impiegato preme il pulsate 'Elimina' nel *MockUp_RicercaCliente*		
DESCRIPTION	**Step n°**	**Attore 1**	**Sistema**
	1	Seleziona il cliente	
	2	Preme il Pulsante ‘Elimina’	
	3		Elimina il cliente
	4		Mostra *MockUP_SucessoEliminazione*
	5	Preme 'OK' nel MockUP_SuccessoEliminazione	

Gestione Addetti alla Sicurezza

Illustriamo ora la parte relativa alla gestione degli **Addetti alla Sicurezza** (da ora in poi **Addetti),** da parte di un **Amministratore** (Vedere Glossario), tramite il seguente Use Case Diagram:



Le funzionalità principali accessibili dall'amministratore sono le seguenti:

- **Inserimento** di un nuovo addetto: l'**Amministratore** può accedere a questa funzionalità cliccando il bottone **Nuovo** nella finestra di **Gestione Addetti**. L'interfaccia mostrerà un form in cui è possibile inserire i dati necessari alla memorizzazione nel Database del nuovo addetto.

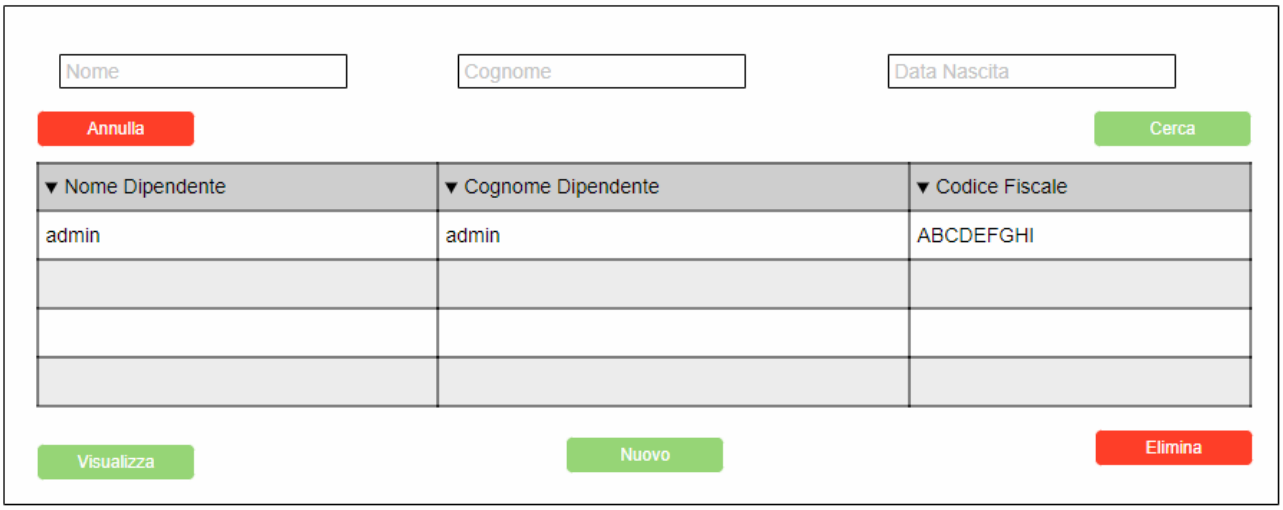
- **Ricerca** degli addetti: la schermata di **Ricerca Addetti** è la prima che viene visualizzata quando viene selezionata la funzionalità di **Gestione Addetti**. La prima volta che viene avviata essa visualizza i dati principali di tutti gli addetti presenti nel Database. L'amministratore può filtrare questi dati utilizzando una qualsiasi combinazione di **Nome**, **Cognome**, o **Data di nascita** in modo da facilitare la ricerca dell'addetto desiderato.

- **Visualizzazione dettagli** di un addetto: è possibile visualizzare tutti i dettagli di un addetto selezionando l'addetto desiderato dalla schermata di ricerca, e cliccando il bottone **Visualizza Dati**. Verrà mostrata una nuova finestra con informazioni aggiuntive sull'addetto selezionato.

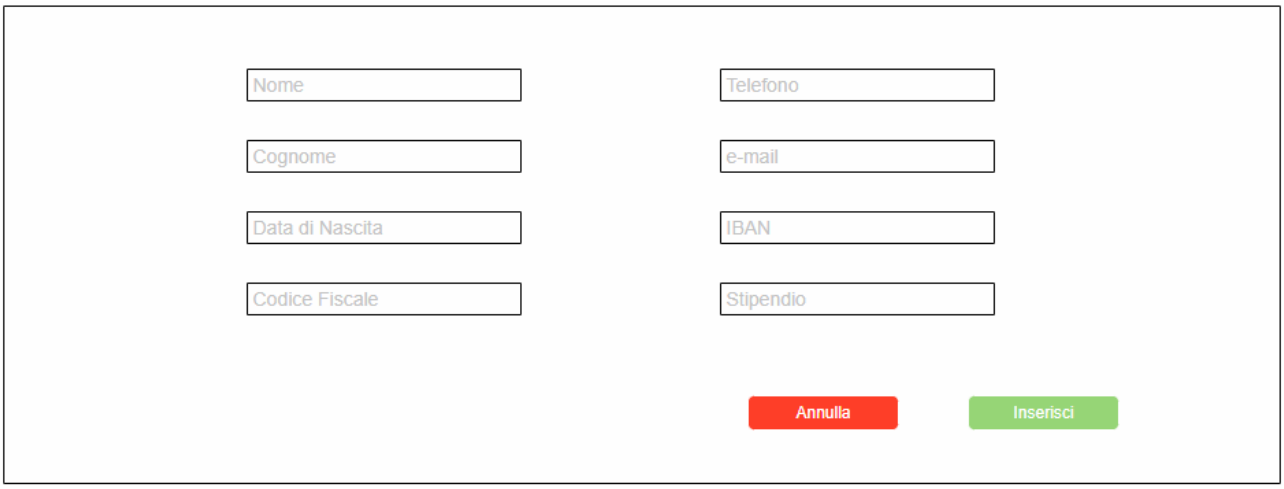
- **Eliminazione** di un addetto: sempre a partire dalla schermata di ricerca, si possono eliminare dal Database le informazioni di un addetto selezionandolo e cliccando su **Elimina**. N.B: in questo modo verranno eliminate dal Database solo le informazioni relative all'impiego dell'addetto; le informazioni relative alla **Persona** verranno conservate.

CockBurn:

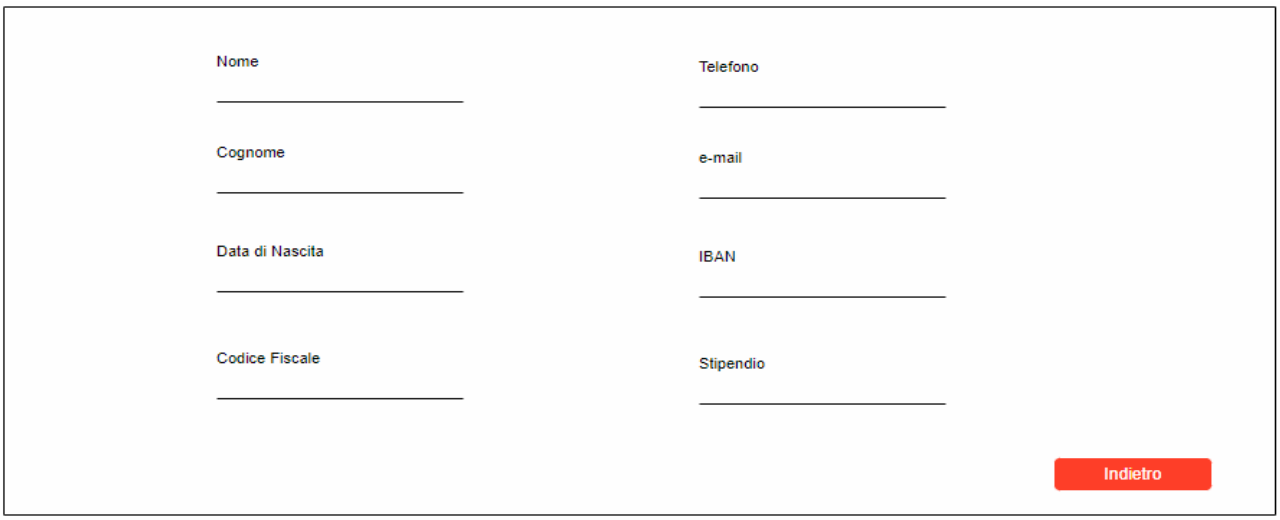
Ora dettagliamo l'interazione tra **Amministratore** e **Sistema** tramite le seguenti tabelle di Cockburn, aiutandoci con l'utilizzo di alcuni mockup dell'interfaccia.



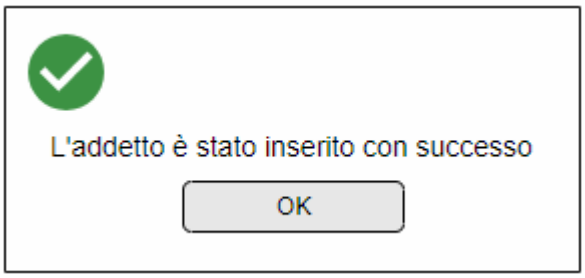
Mockup_RicercaAddetti



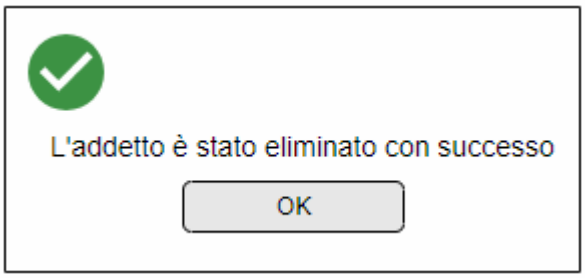
Mockup_NuovoAddetto



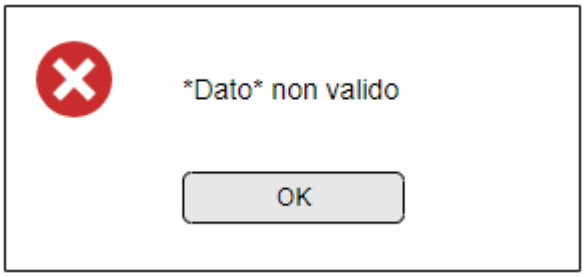
Mockup_DettagliAddetto



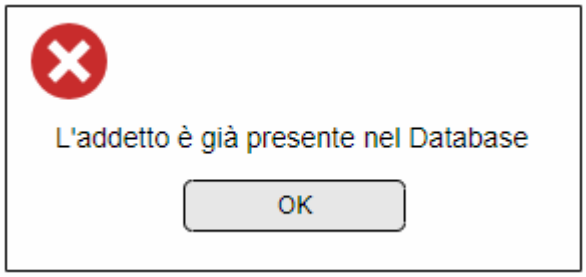
Mockup_AddettoInserito



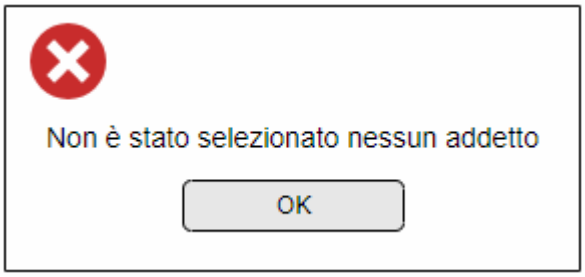
Mockup_AddettoEliminato



Mockup_AlertDatiAddetto



Mockup_AlertAddettoEsistente



Mockup_AlertSelezioneAddetto

CockBurn per lo Use Case : Ricerca Addetti

USE CASE *#1*	*Ricerca Addetti*		
Goal in Context	L'amministratore vuole ricercare uno o più addetti alla sicurezza presenti nel Database.		
Preconditions	L'amministratore deve essere loggato nel sistema.		
Success End Condition	Gli addetti corrispondenti vengono visualizzati in una tabella.		
Failed End Condition	I dati inseriti non corrispondono a nessun utente. La tabella viene visualizzata vuota.		
Primary Actor	Amministratore		
Trigger	Amministratore preme il bottone *Gestione Addetti* dal menu principale		
DESCRIPTION	**Step**	**Attore 1**	**Sistema**
	1	Preme il bottone Gestione Addetti (*MockUp_MenuPrincipale)*	
	2		Mostra schermata *MockUp_RicercaAddetti*
	3		Compila la tabella con la lista di tutti gli addetti presenti nel Database.
SUBVARIATION #1	***Step***	**Attore 1**	**Sistema**
	3.1	Inserisce 'Nome' nell'apposito campo *(opzionale)*	
	4.1	Inserisce 'Cognome' nell'apposito campo *(opzionale)*	
	5.1	Inserisce 'Data Nascita' nell'apposito campo *(opzionale)*	
	6.1	Preme il bottone *Cerca*	
	7.1		Mostra la tabella con gli addetti corrispondenti ai criteri di ricerca inseriti.
SUBVARIATION #2	***Step***	**Attore 1**	**Sistema**
	6.1.1	Preme il bottone *Annulla*	
	7.1.1		Svuota i campi *Nome*, *Cognome*, *Data Nascita*

CockBurn per lo Use Case : ' Elimina Addetto'

USE CASE #2	*Elimina Addetto*		
Goal in Context	L'amministratore vuole eliminare un addetto presente nel database.		
Scope & Level			
Preconditions	• L'amministratore deve essere loggato. • Deve essere stato effettuato lo Use Case *Ricerca Addetti.*		
Success End Condition	L'addetto selezionato viene eliminato dal Database.		
Failed End Condition	L'amministratore non seleziona nessun addetto.		
Primary Actor	Amministratore		
Trigger	L'amministratore preme il bottone *Elimina* da *Mockup_RicercaAddetti*		
DESCRIPTION	**Step**	**Attore 1**	**Sistema**
	1	Seleziona un addetto dalla tabella *(Mockup_RicercaAddetti)*	
	2	Preme il bottone 'Elimina' *(Mockup_RicercaAddetti)*	
	3		Mostra *Mockup_AddettoEliminato*
	4	Preme il bottone 'OK' *(Mockup_AddettoEliminato)*	
	5		Mostra *Mockup_RicercaAddetti*
SUBVARIATION #1	1.1	Preme il bottone 'Elimina' senza aver selezionato un addetto *(Mockup_RicercaAddetti)*	
	2.1		Mostra *Mockup_AlertSelezioneAddetto*

CockBurn per lo Use Case: 'Visualizza Dettagli Addetto'

USE CASE *#3*	*Visualizza Dettagli Addetto*		
Goal in Context	L'amministratore vuole visualizzare tutti i dettagli memorizzati nel Database relativi ad un addetto.		
Scope & Level			
Preconditions	• L'amministratore deve essere loggato. • *Deve essere stato effettuato lo Use Case Ricerca Addetti.*		
Success End Condition	Vengono visualizzati i dati relativi all'addetto selezionato.		
Failed End Condition	L'amministratore non seleziona nessun addetto.		
Primary Actor	Amministratore		
Trigger	L'amministratore preme il bottone *Visualizza* in Mockup_RicercaAddetti		
DESCRIPTION	**Step**	**Attore 1**	**Sistema**
	1	Seleziona un addetto dalla tabella *(Mockup_RicercaAddetti)*	
	2	Preme il bottone 'Visualizza' *(Mockup_RicercaAddetti)*	
	3		Mostra *Mockup_DettagliAddetto*
	4	Preme il bottone 'Indietro' *(Mockup_DettagliAddetto)*	
	5		Mostra *Mockup_RicercaAddetto*
SUBVARIATION #1	**Step**	**Attore 1**	**Sistema**
	1.1	Preme il bottone 'Visualizza' senza aver selezionato un addetto *(Mockup_RicercaAddetto)*	
	2.1		Mostra *Mockup_AlertSelezioneAddetto*

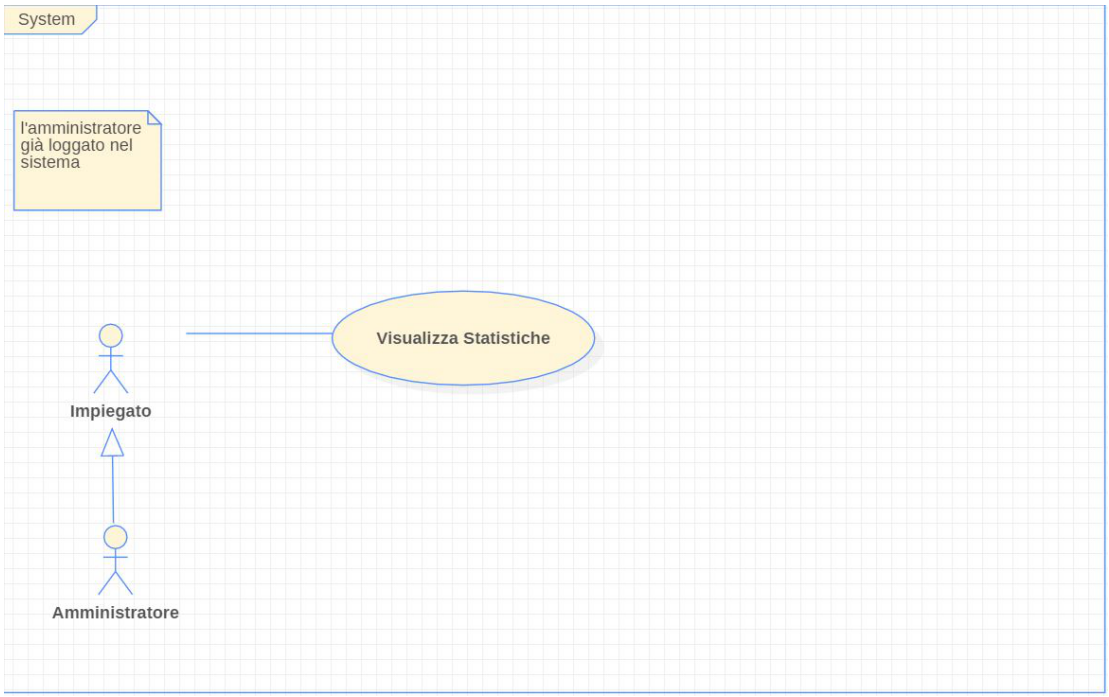
CockBurn per lo Use Case: 'Inserisce Addetto'

USE CASE *#4*	*Inserisce Addetto*		
Goal in Context	L'amministratore vuole inserire un nuovo addetto nel sistema.		
Scope & Level			
Preconditions	• L'amministratore deve essere loggato. • Il sistema deve mostrare la schermata Ricerca Addetti.		
Success End Condition	L'addetto viene correttamente inserito nel Database.		
Failed End Condition	• L'amministratore inserisce dati errati • Un addetto con lo stesso Codice Fiscale è già presente nel Database		
Primary Actor	Amministratore		
Trigger	L'amministratore preme il bottone *Nuovo* in Mockup_RicercaAddetti		
DESCRIPTION	**Step**	**Attore 1**	**Sistema**
	1	Preme il bottone 'Nuovo' *(Mockup_RicercaAddetti)*	
	2		Mostra *Mockup_NuovoAddetto*
	3	Inserisce tutti i dati nel form	
	4	Preme il bottone 'Inserisci'	
	5		Mostra *Mockup_AddettoInserito*
SUBVARIATION #1	**Step**	**Attore 1**	**Sistema**
	3.1	Inserisce tutti i dati nel form, ma un campo è vuoto o errato	
	4.1	Preme il bottone 'Inserisci'	
			Mostra *Mockup_AlertDatiAddetto*
SUBVARIATION #2	**Step**	**Attore 1**	**Sistema**
	3.2	Inserisce correttamente i dati nel form, ma nel Database è già presente un addetto con lo stesso Codice Fiscale	
	4.2	Preme il bottone 'Inserisci'	
	5.2		Mostra *Mockup_AlertAddettoEsistente*

SUBVARIATION #3	**Step**	**Attore 1**	**Sistema**
	3.3	Preme il bottone 'Annulla'	
	4.3		Mostra *Mockup_RicercaAddetto*

Statistiche

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

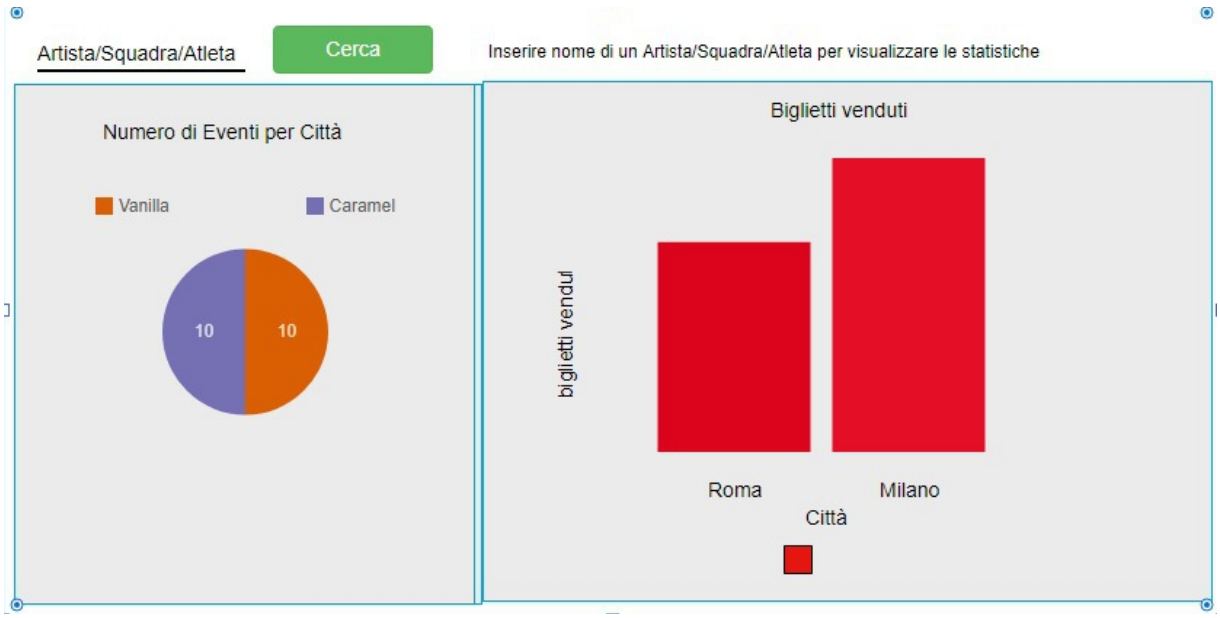


L'Impiegato del Back-Office può cercare tramite il nome di un artista le statistiche relative alle partecipazione agli eventi in diverse città

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: Mock_up_Statistiche:



USE CASE #x	Gestione_Cliente		
Goal in Context	L’amministratore vuole visionare le statistiche		
Scope & Level			
Preconditions	L’utente deve essere un Impiegato BackOffice o Amministratore		
Success End Condition	L’amministratore visualizza le statistiche		
Failed End Condition	La ricerca non va a buon fine		
Primary Actor	**Impiegato BackOffice**		
Trigger	L'utente preme il pulsante 'Statistiche' nel menu		
DESCRIPTION	**Step n°**	**Attore 1**	**Sistema**
	1	Inserisce un artista	
	2	Preme il bottone ‘Cerca’	
	3		Mostra il grafico a torta e l’istogramma
SUBVARIATIONS	Step	**Attore 1**	**Sistema**
	1.1	Lascia il campo vuoto/dati errati	
	2.1	Preme il bottone ‘Cerca’	
	3.1		Mostra 'MockUp_ErroreDati'

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 useremo i **Class Diagram**, **Sequence Diagram**, **Activity Diagram**.

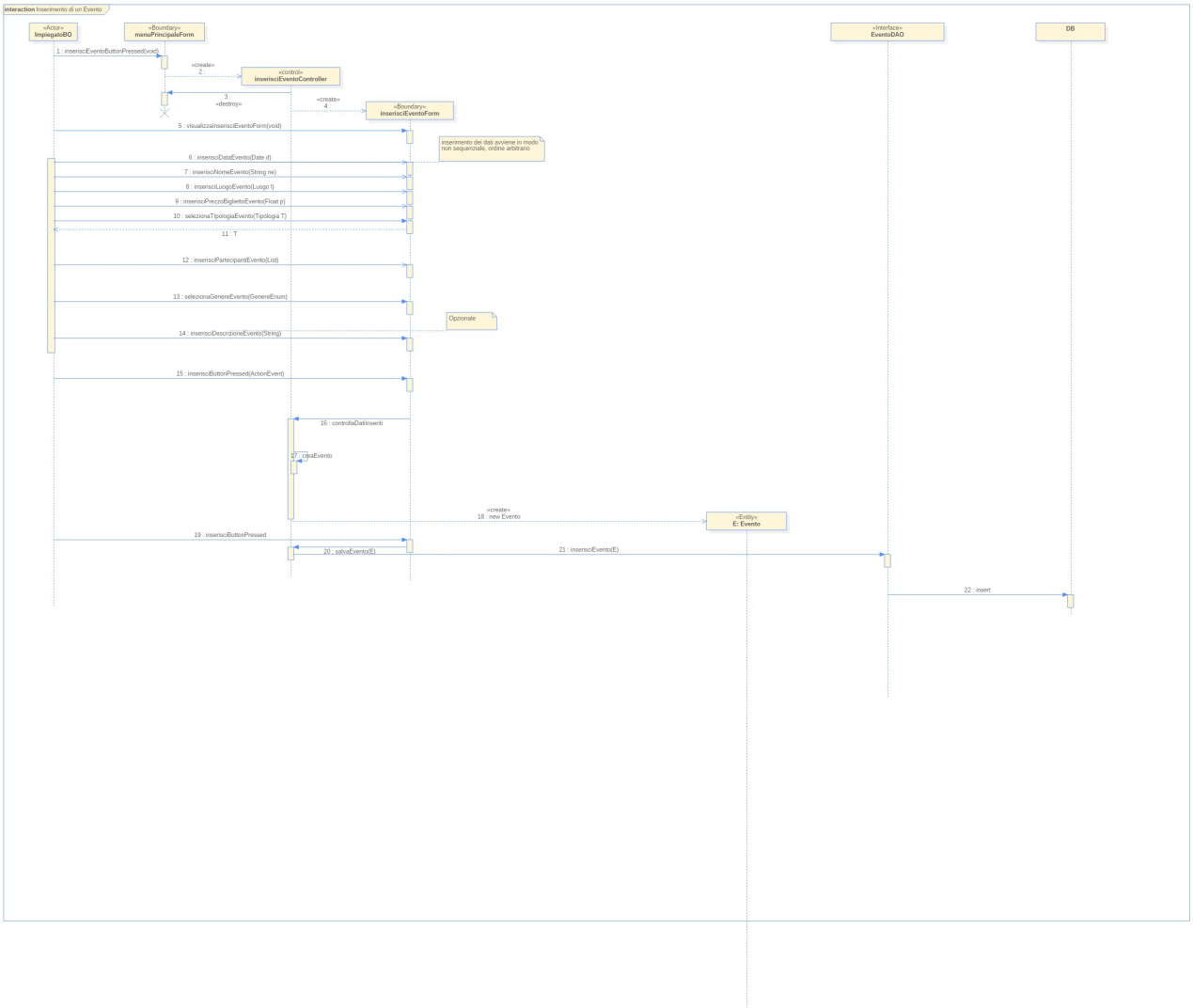
In questa fase si cercherà di individuare gli oggetti software necessari per modellare con più dettaglio quello descritto nella sezione di Analisi, per fare questo abbiamo usato l'Euristica Three-Object (Entiry,Control,Buondary), seguendo le seguenti regole:

1. Gli oggetti *Boundary* 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 **Boundary**, 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 **Boundary**, 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 **Boundary,** rappresenta il *MockUp_MenuPrincipale,* dovrà creare i vari form che serviranno a compiere le varie funzionalità del sistema.
- *InserisciEventoForm:* Classe di tipo **Boundary**, 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.

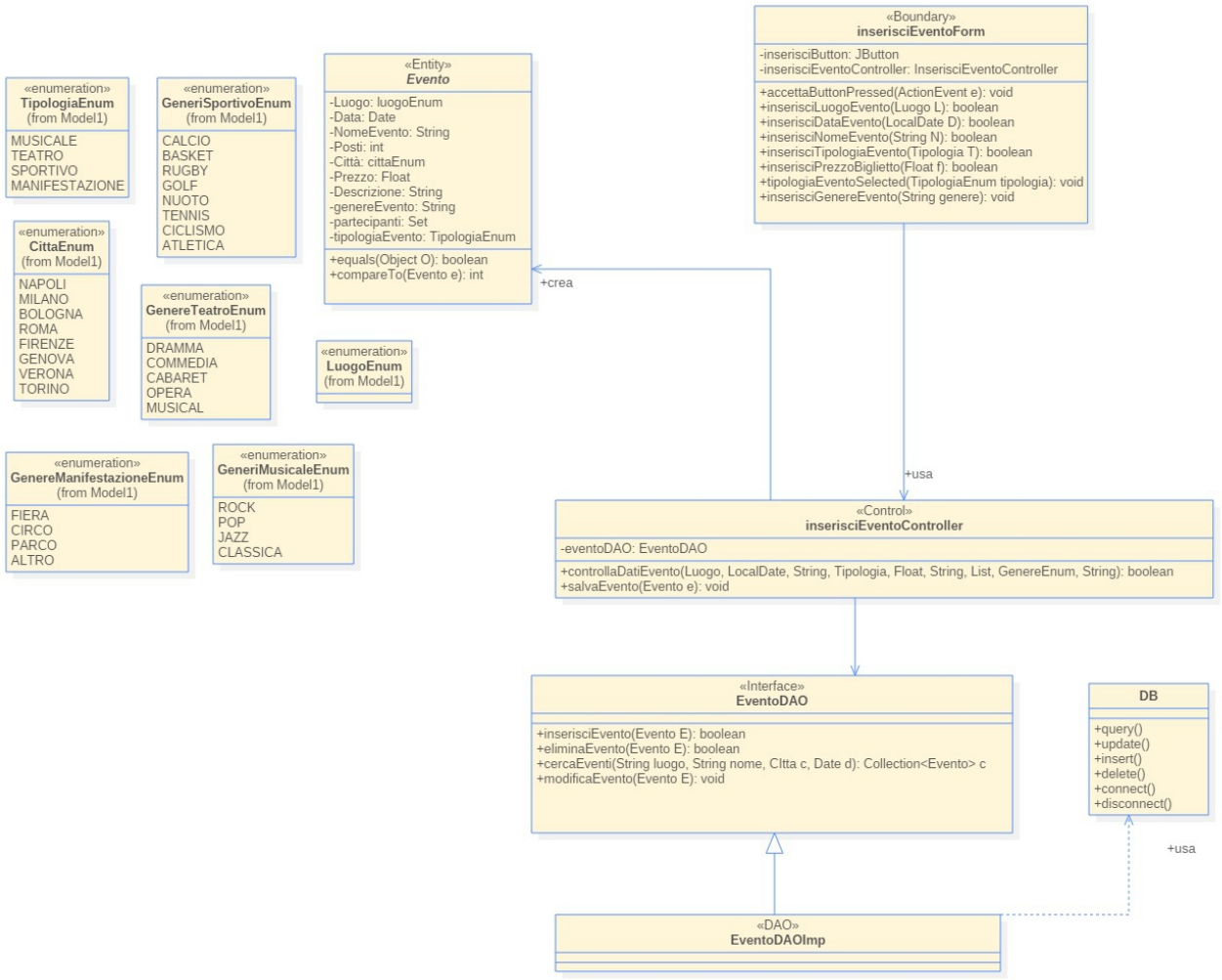
Abbiamo aggiunto delle enumerazioni per i tipo di dato con i dominio limitato:

- *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.

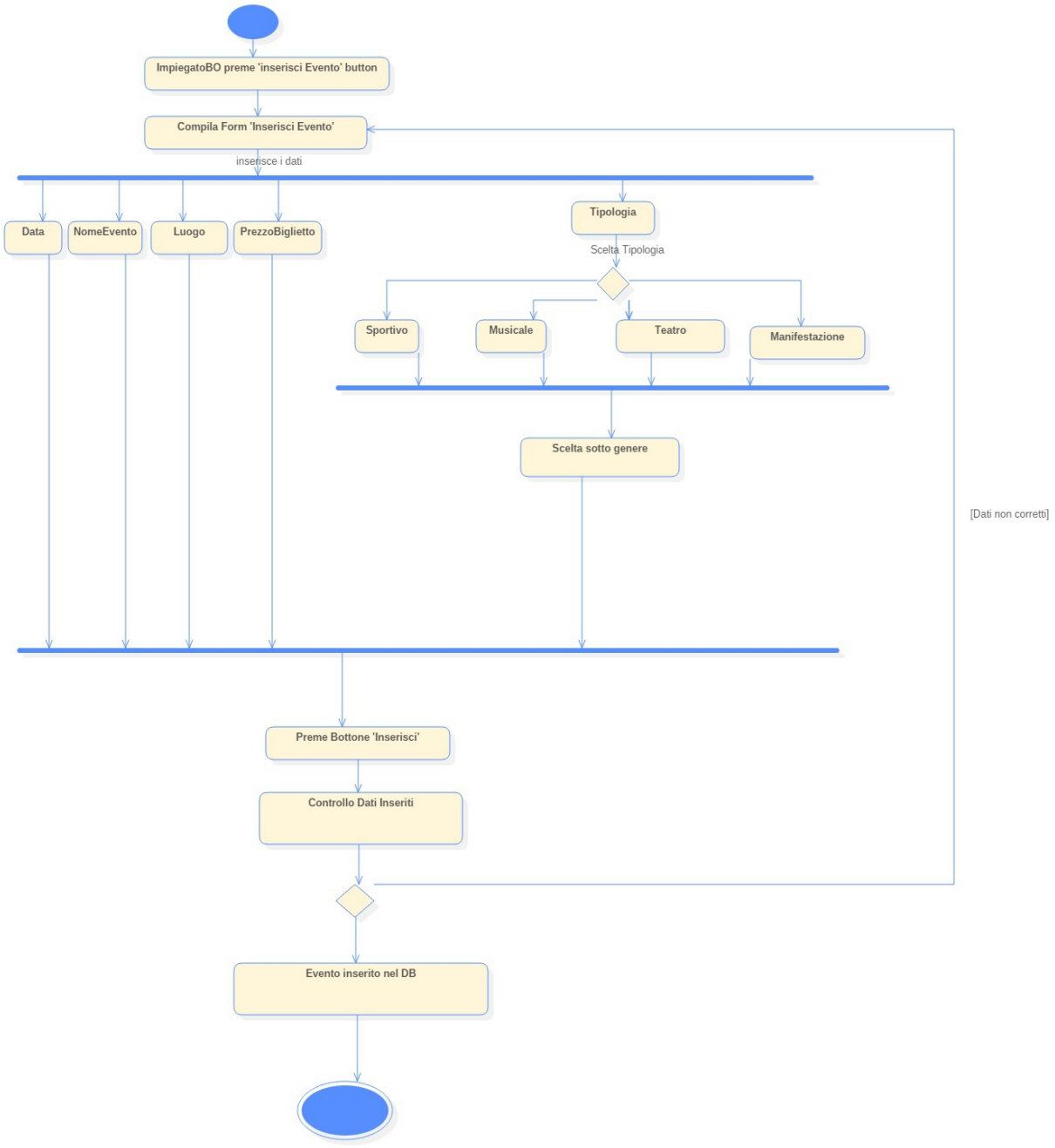
Vincoli OCL :

- 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).

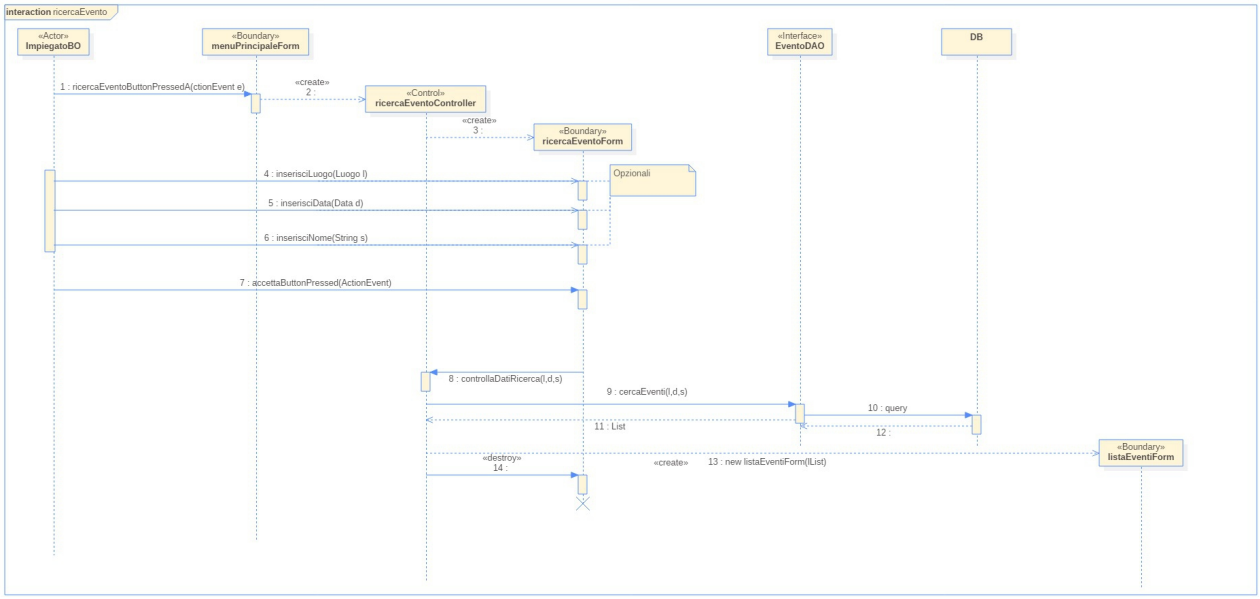
Oltre questi vincoli, tutti gli attributi dell'evento devono essere diversi da NULL, fatta eccezione per la descrizione, la quale è opzionale, e la DataEvento e LuogoEvento non devono esistere nella lista degli eventi già presenti.
Si può notare che nonostante siano presenti 4 enumerazioni per i sottogeneri, uno per ogni tipologia, abbiamo come tipo dell'attributo 'GenereEvento : String', questa scelta è stata fatta per facilitare l'implementazione.

Activity Diagram



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.



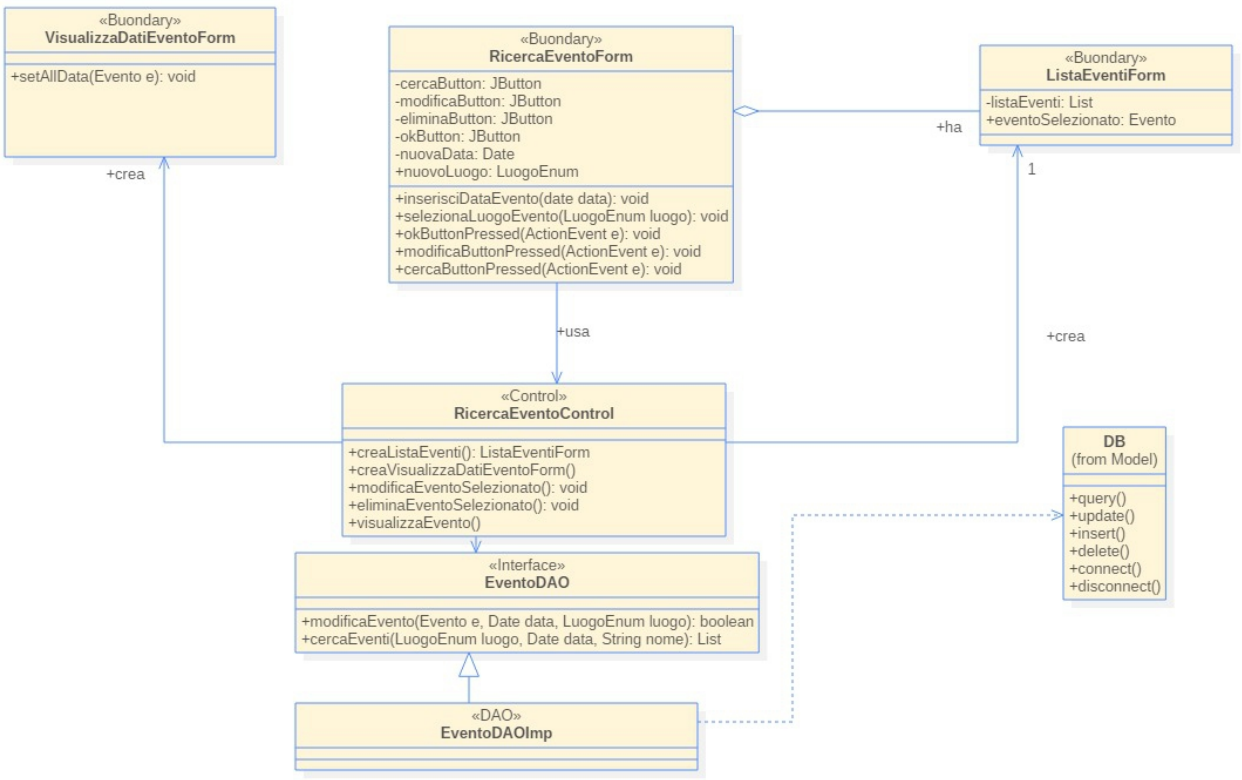
Sommario classi usate:

- *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 **Boundary**, 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 selezionato utile per le funzionalità su un evento,prima nominate.

Vincoli:

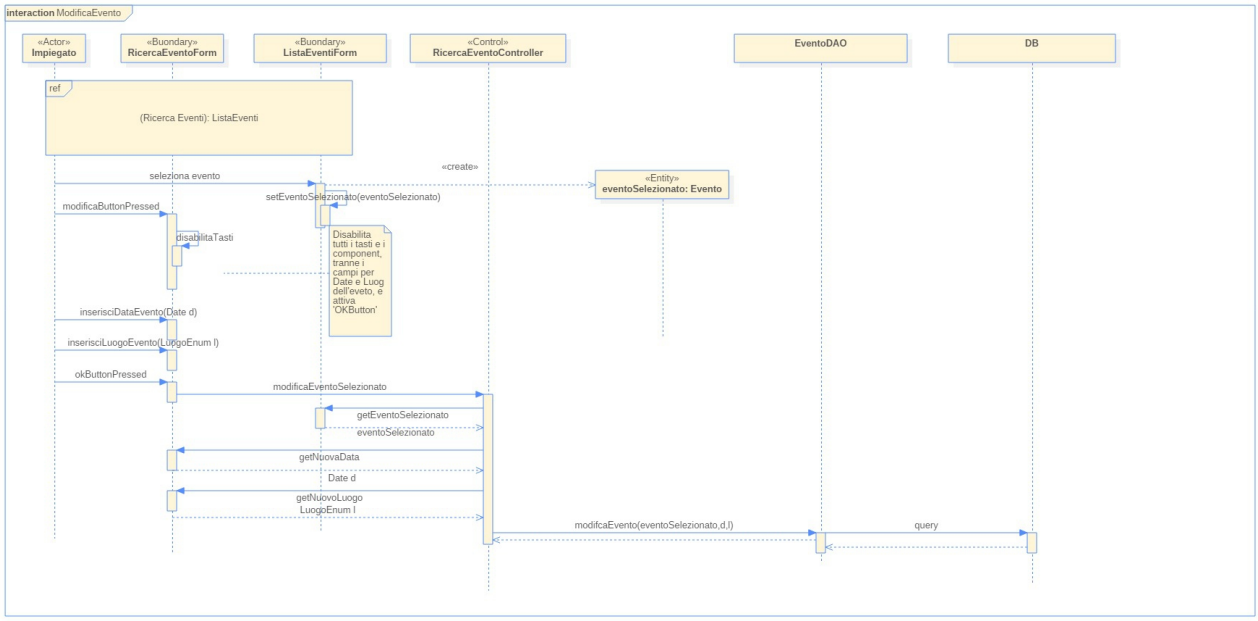
- Prima di poter modificare, eliminare o visualizzare un evento, l'Attore deve averlo selezionato dall'apposita lista di eventi.
- Per quanto riguarda la modifica, i dati seguendo i vincoli dell'inserimento, per i campi DataEvento e LuogoEvento.

Rappresentiamo ora le classi in un Class Diagram

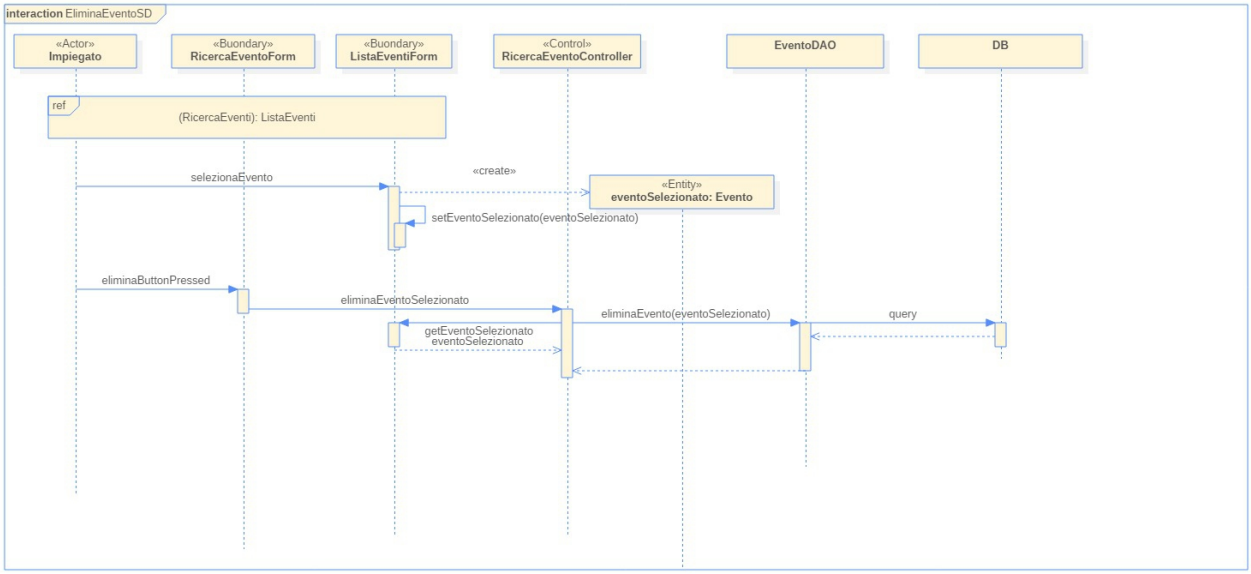


Proseguiamo con i Sequence Diagram per le altre tre funzionalità:

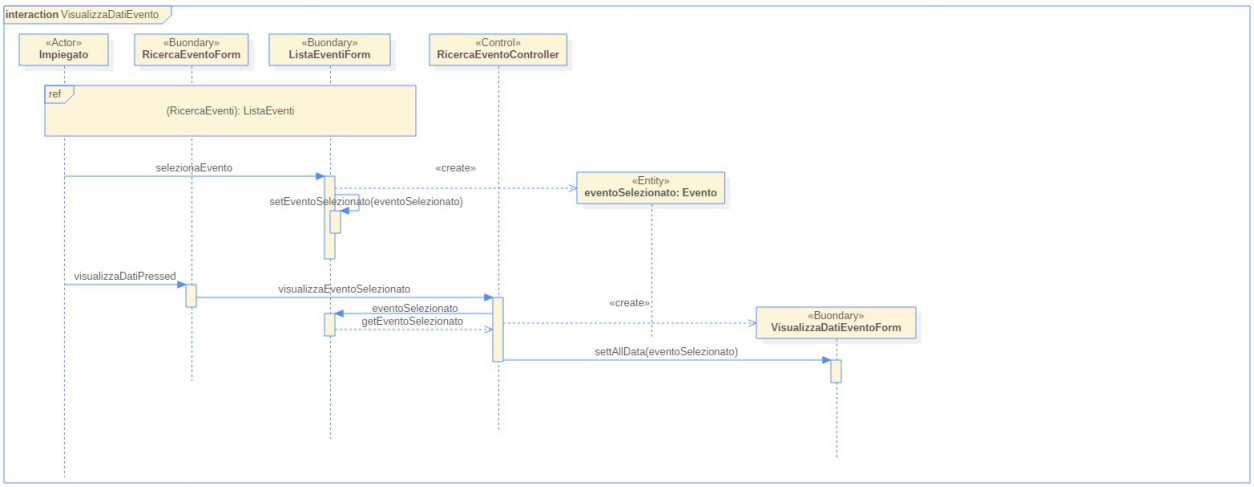
Modifica Evento:



Elimina Evento

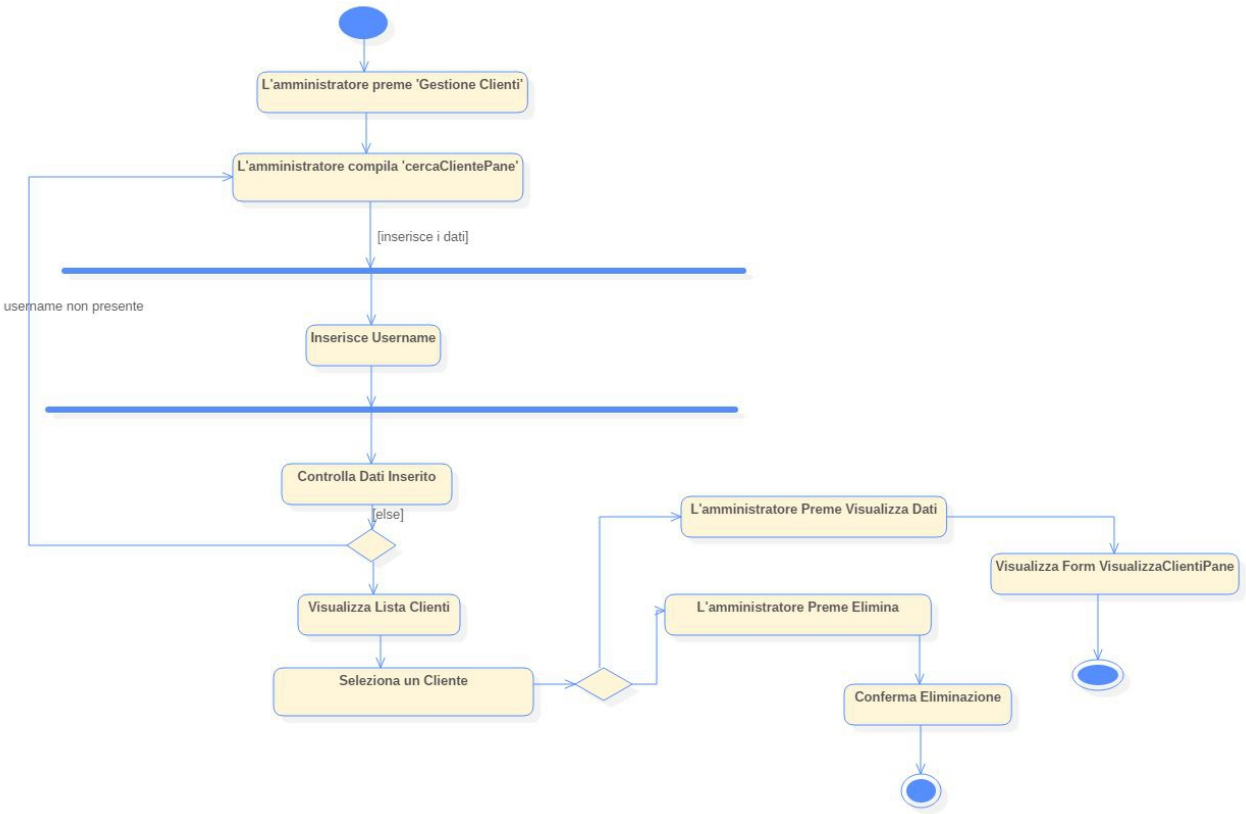


Visualizza dati Evento


interaction VisualizzaDatiEvento
«Actor»
Impiegato
«Buondary»
RicercaEventoForm
«Buondary»
ListaEventiForm
«Control»
RicercaEventoController
ref
(RicercaEventi): ListaEventi
selezionaEvento
«create»
«Entity»
eventoSelezionato: Evento
setEventoSelezionato(eventoSelezionato)
visualizzaDatiPressed
visualizzaEventoSelezionato
«create»
«Buondary»
VisualizzaDatiEventoForm
eventoSelezionato
getEventoSelezionato
settAllData(eventoSelezionato)

Gestione Clienti:

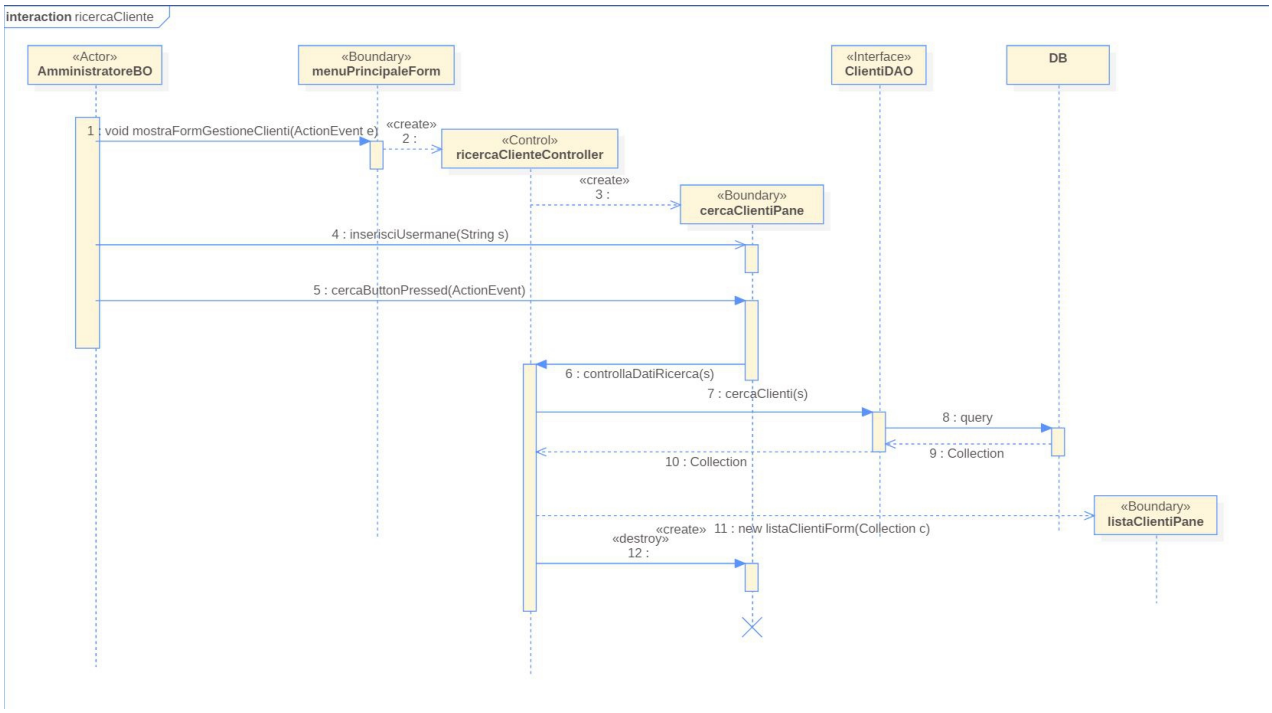
Per quanto riguarda i clienti come detto in precedenza abbiamo la ricerca di un cliente che porta a due scelte eliminazione o visualizzazione dei dati come si vede nell'Activity Diagram.



Ricerca Cliente:

Per questa funzionalità abbiamo usato la stessa tecnica del Three-Object-Type usato negli Eventi, questa funzionalità comprenderà la: Visualizzazione dei dati ed Eliminazione di un cliente

Fa riferimento quindi al *MockUp_RicercaClienti.*
Il Sequence Diagram, ci sarà utile per identificare una prima versione degli oggetti necessari



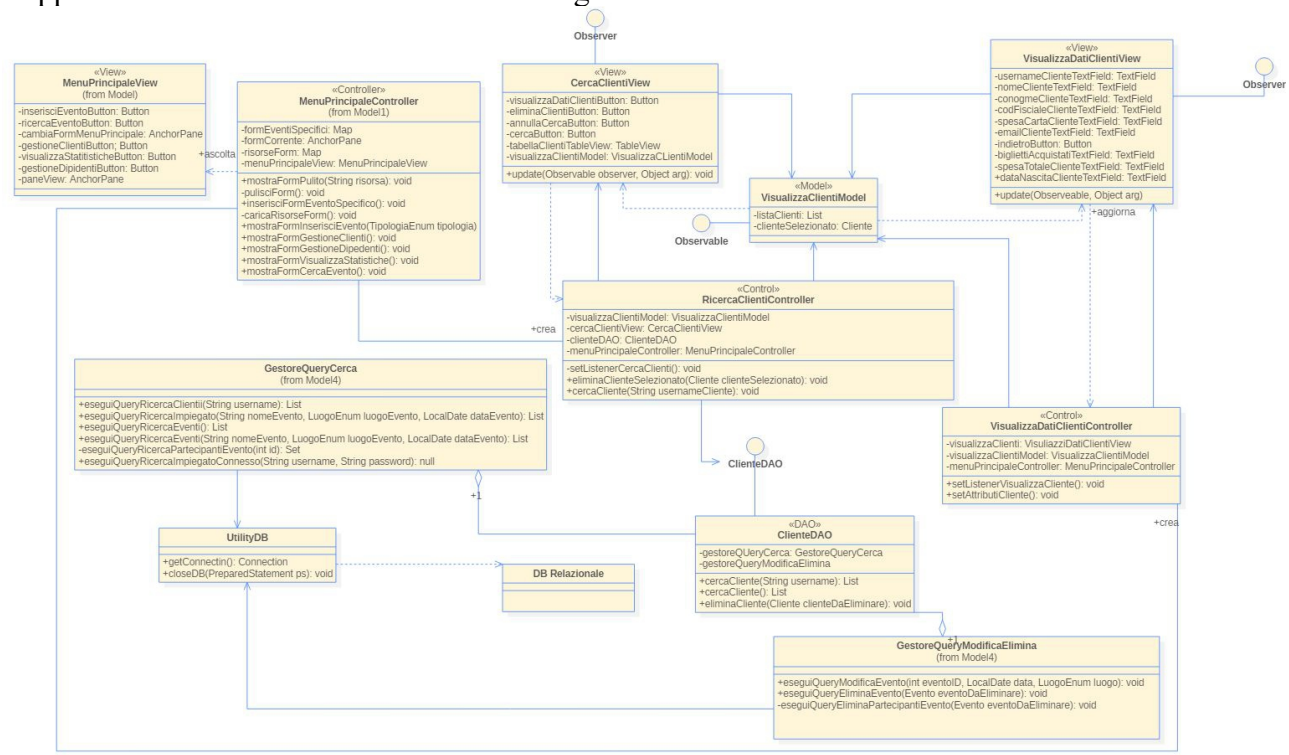
Sommario classi usate:

- *ricercaClienteController*: Questo oggetti di tipo Control, creare il form per la ricerca evento, e comunicherà con le classi DAO per raccogliere i dati della ricerca.
- *cercaClientiPane*: Ci permetterà di 'leggere' le azioni dell'Attore, questa classe Boundary, ci permetterà di effettuare le tre funzionalità legate alla ricerca di un cliente (Visualizzazione, ed Eliminazione di un Cliente).
- *listaClientiPane*: visualizzerà la lista dei clienti ricercati, e conterrà il riferimento al cliente selezionato utile per le funzionalità prima nominate.

Vincoli:

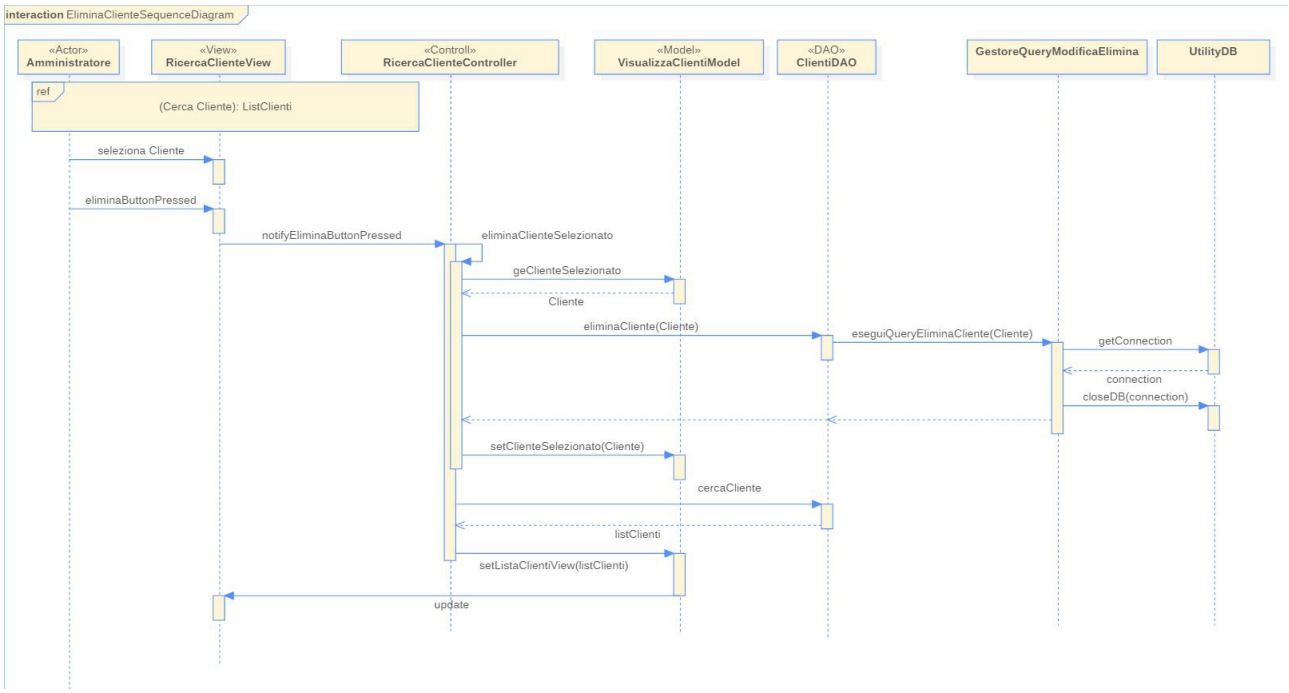
- Prima di poter eliminare o visualizzare un cliente, l'Amministratore deve averlo selezionato dall'apposita lista.

Rappresentiamo ora le classi in un Class Diagram:

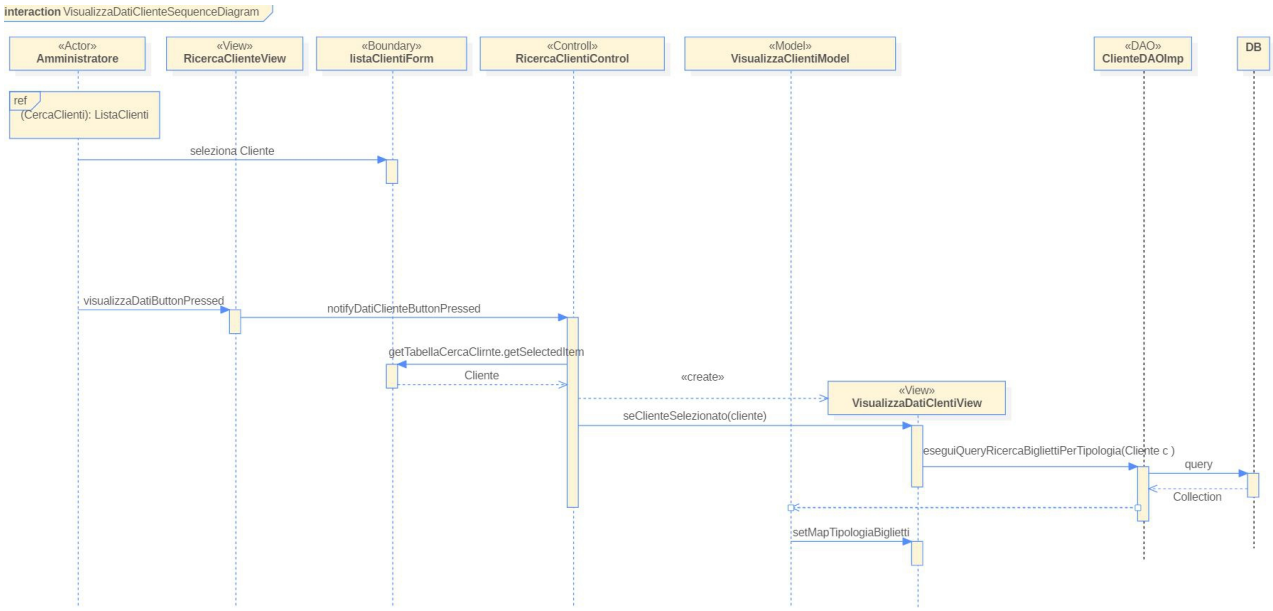


Proseguiamo con i Sequence Diagram per le altre due funzionalità:

Elimina Cliente:



Visualizza Clienti:



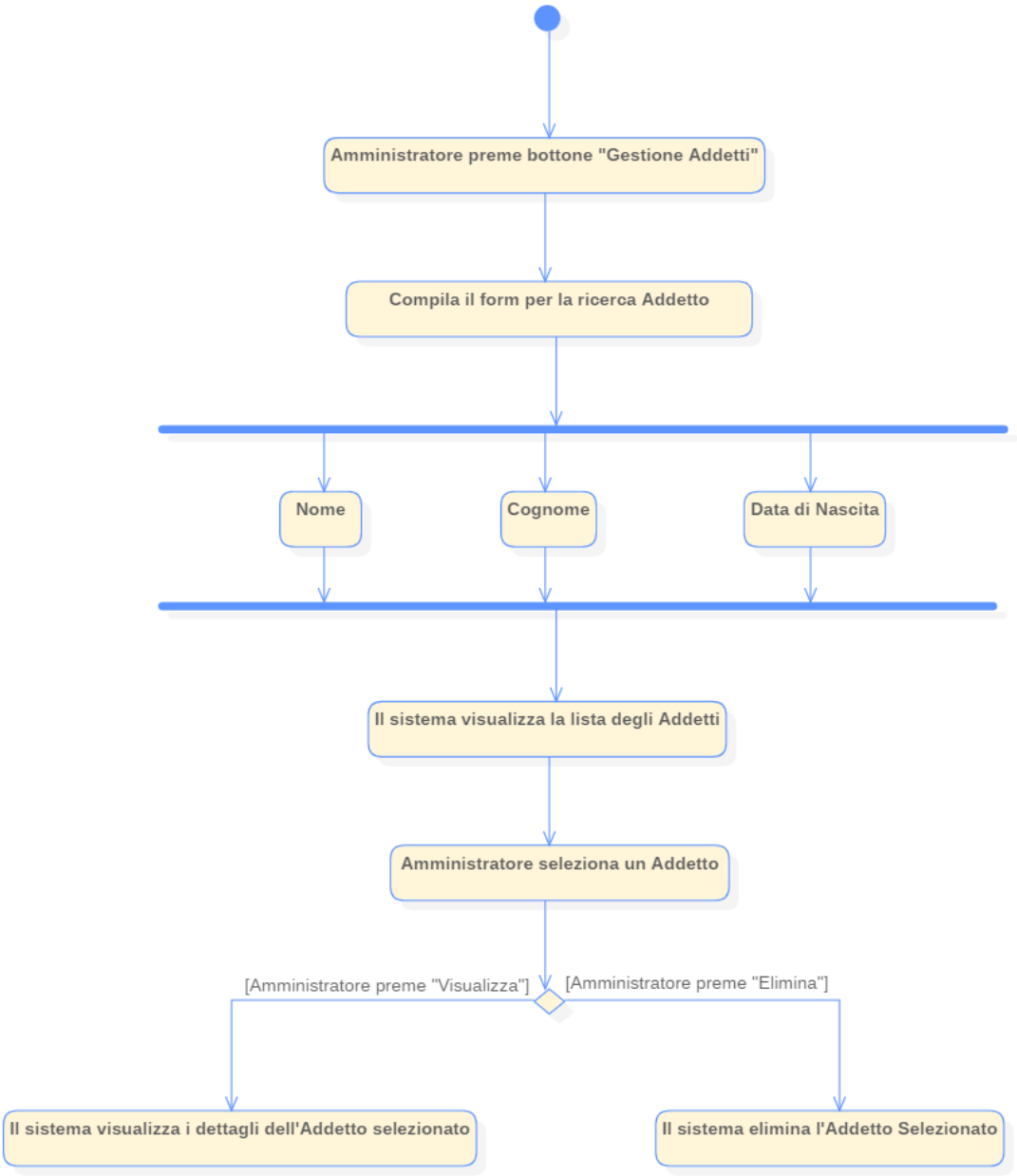
Gestione Addetti

L'approccio usato per questa funzionalità è sempre quello Three-Object-Type. La principale classe **Entity** in questo caso è *Addetto*. Le classi di tipo **Boundary** e **Control** sono diverse a seconda della sottofunzionalità specifica.
Questa funzionalità, infatti, è stata suddivisa in:

- **Inserimento Addetto**
- **Ricerca Addetti**
- **Eliminazione Addetto**
- **Visualizzazione Dettagli Addetto**

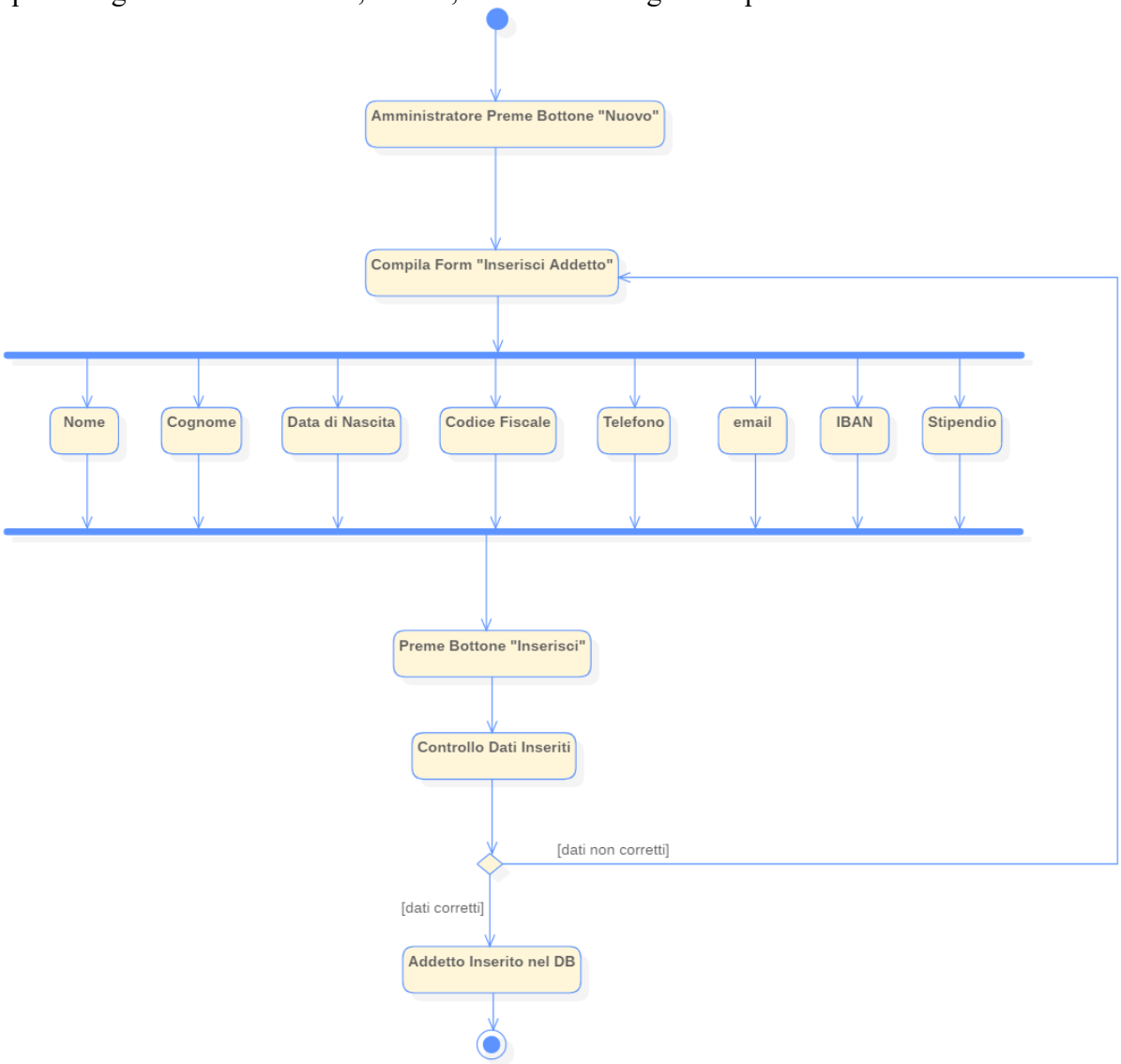
Le ultime 3 funzionalità elencate verranno gestite dalla stessa interfaccia *(Ricerca Cliente)*, come evidenziato nelle tabelle di Cockburn precedenti.

Il flusso delle attività per queste 3 funzioni, verrà gestito come mostrato nel seguente Activity Diagram:



Nel quale la ricerca viene completata nel passo *"Il sistema visualizza la lista degli Addetti"*, mentre gli ultimi 2 passi corrispondono alle funzionalità di Visualizzazione e Eliminazione. L'inserimento dei dati *Nome*, *Cognome*, e *Data di Nascita* è opzionale. Nel caso in cui non vengano inseriti, verranno visualizzati tutti gli addetti presenti nel sistema.

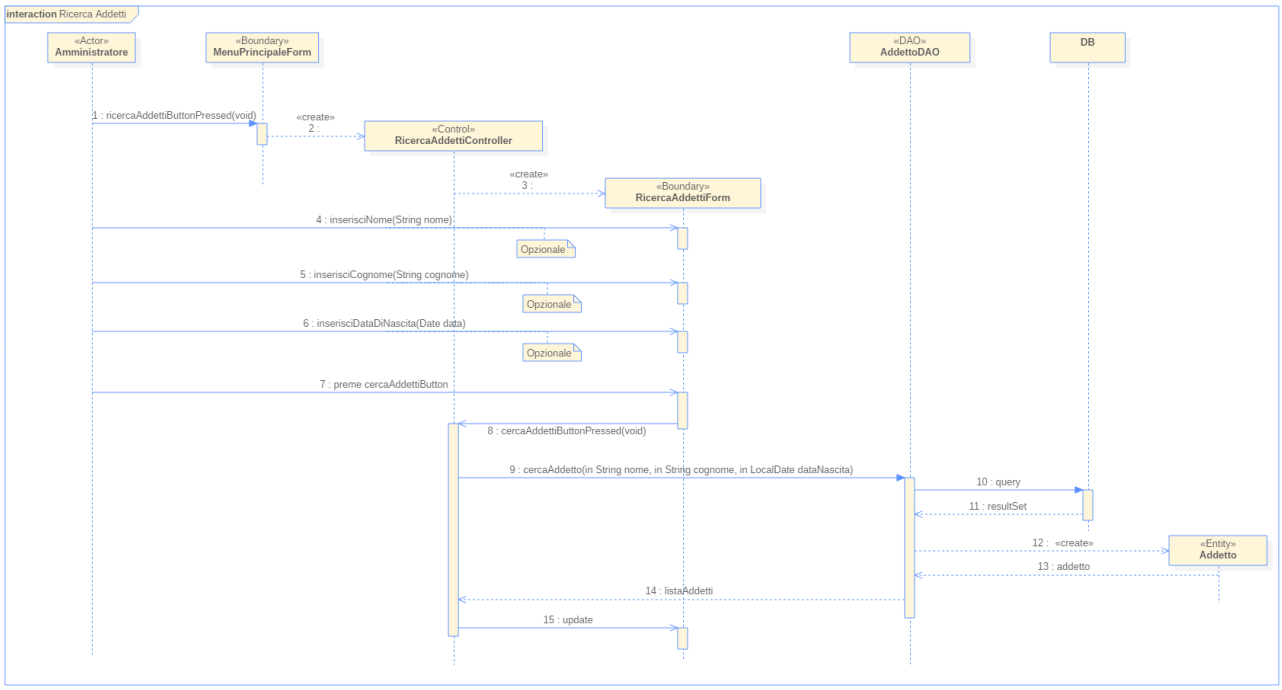
Per quanto riguarda l'Inserimento, invece, il flusso si svolgerà in questo modo:



Passiamo all'analisi delle interazioni per ogni sottofunzionalità.

Ricerca Addetto

Presentiamo per primo il Sequence Diagram derivato dal flusso mostrato in precedenza della funzione di Ricerca Addetto, in quanto come detto in precedenza, da questa interfaccia è poi possibile accedere alle altre funzionalità.



Sommario delle classe usate:

- *MenuPrincipaleForm:* Già presentato in precedenza.
- *RicercaAddettiForm:* Classe di tipo **Boundary**, che rappresenta l'interfaccia con cui interagirà l'**Attore**. In questa interfaccia sarà possibile inserire i dati (nome, cognome, data di nascita) per eseguire la ricerca degli addetti corrispondenti nel Database. I risultati della ricerca saranno mostrati in una tabella. Da questa interfaccia, inoltre, sarà possibile accedere alle funzionalità di inserimento, visualizzazione e eliminazione tramite appositi bottoni. Questa classe comunica al **Controller** quali azioni vengono eseguite dall'utente.
- *RicercaAddettiController:* Classe **Control**, si occupa di chiamare le funzioni richieste dall'utente alle classi corrispondenti. Nel diagramma mostrato, comunica alla classe **DAO** i dati inseriti per la ricerca, ma nei diagrammi che seguiranno vedremo che la classe si occuperà anche di creare le interfacce per le altre funzionalità presenti, oltre a comunicare sempre con la classe **DAO** per inserimento ed eliminazione.
- *AddettoDAO:* Classe **DAO**, comunica solo con il **Control** e l'implementazione della basi di dati scelta.
- *Addetto:* Classe **Entity**, modella l'addetto.

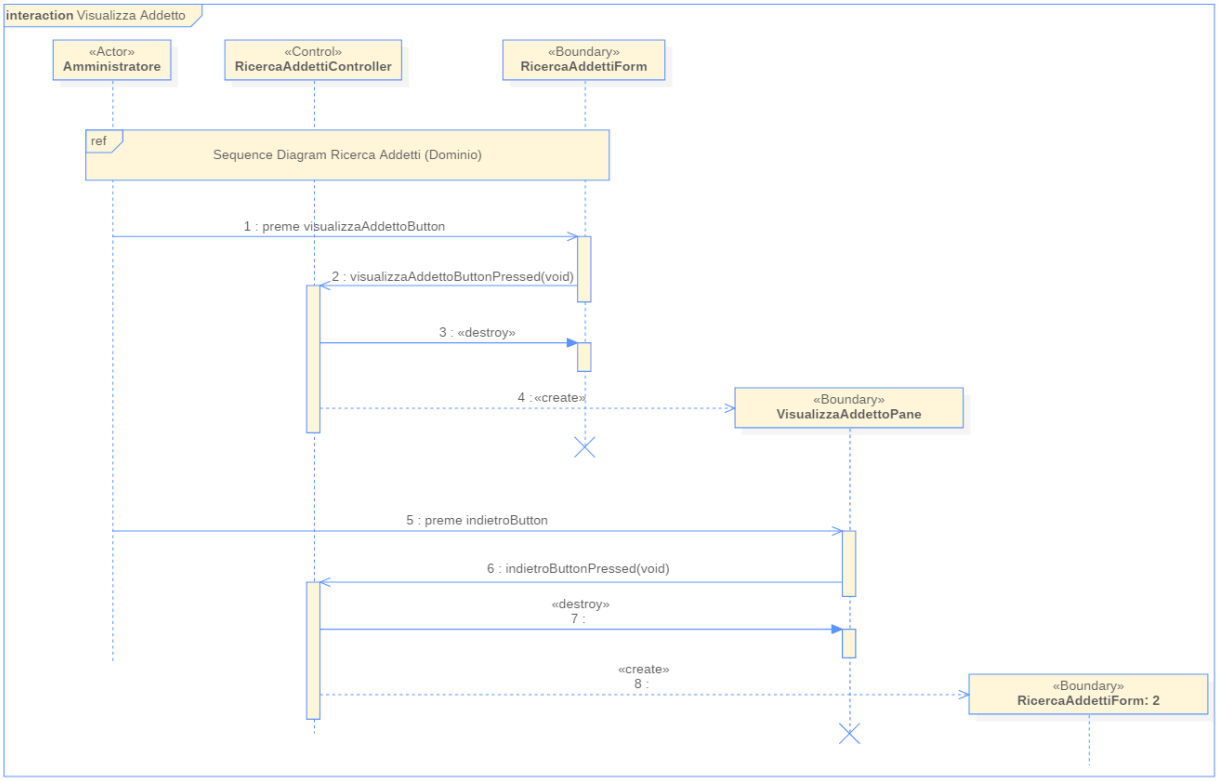
Vincoli:

- Prima di poter modificare, eliminare o visualizzare un addetto, l'Attore deve averlo selezionato dall'apposita lista di addetti.

Mostreremo il Class Diagram subito dopo aver presentato la funzionalità di **Visualizzazione Dettagli**, in quanto le classi per entrambi i casi d'uso possono essere rappresentate in uno stesso diagramma.

Visualizzazione Dettagli Addetto

Si potrà accedere alla funzionalità di **Visualizzazione Dettagli** dall'interfaccia di **Ricerca**, cliccando il tasto corrispondente dopo aver selezionato l'**Addetto** di cui si vogliono visualizzare i dati. Il controller mostrerà una nuova finestra in cui saranno visualizzati nel dettaglio i dati dell'addetto presenti nel Database. Il Sequence Diagram che segue, dunque, richiede che venga eseguito prima quello mostrato nel punto precedente, come evidenziato dal riferimento inserito:

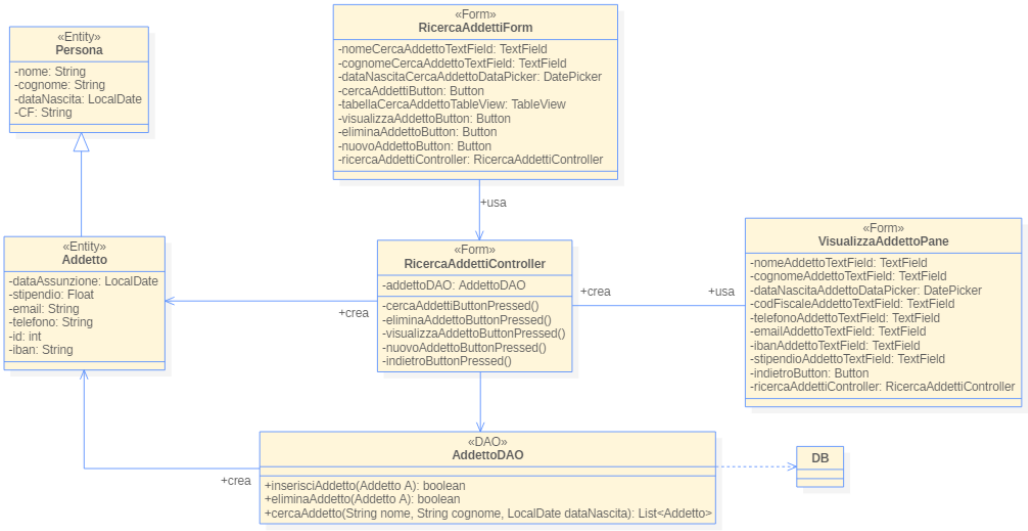


Sommario delle classe usate:

- *VisualizzaAddettoPane:* Classe di tipo **Boundary**, che mostrerà le informazioni complete dell'addetto selezionato. L'unica altra funzione presente in questa interfaccia sarà quella di poter tornare indietro al menu di Ricerca.

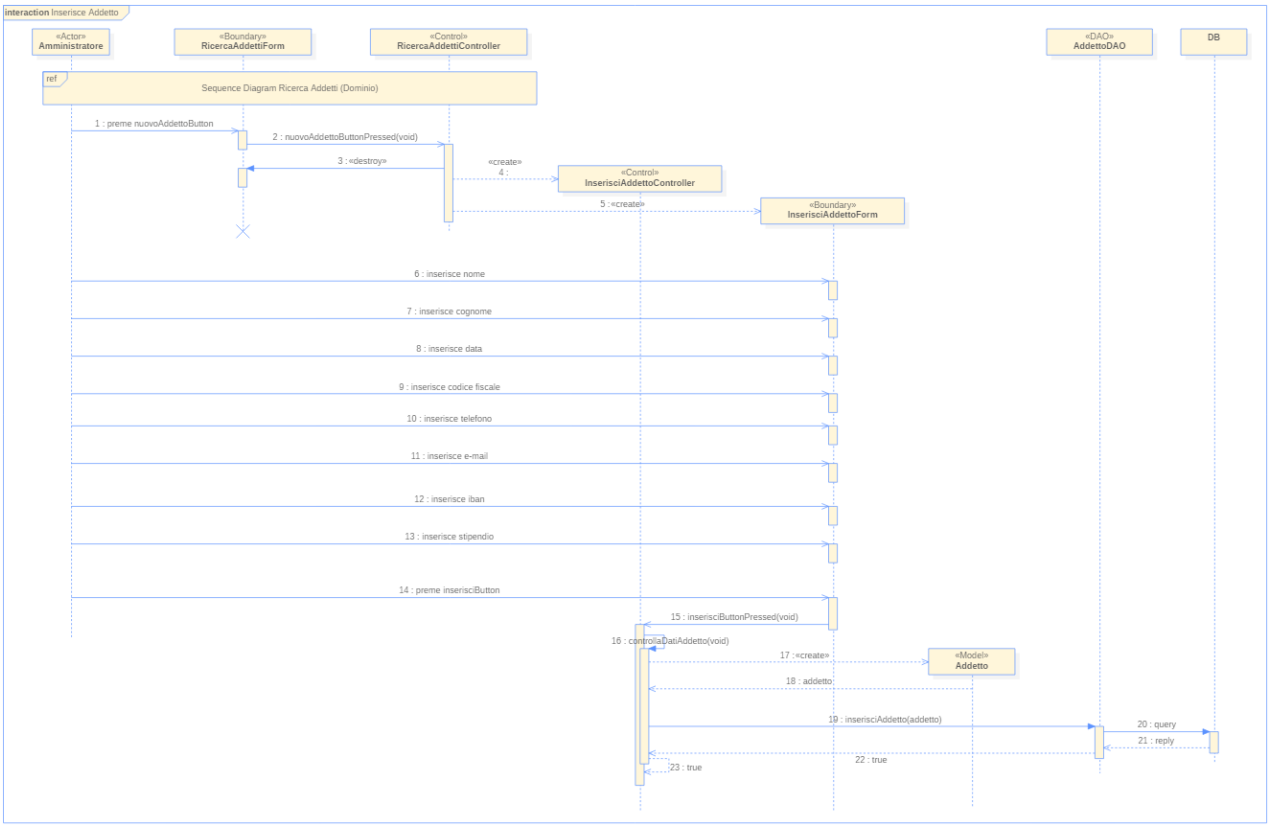
Le rimanenti classi sono già state descritte nel punto precedente.

Possiamo ora passare alla presentazione del Class Diagram relativo alle funzioni di **Ricerca** e **Visualizzazione Dettagli**:



Inserimento Addetto

Si potrà accedere alla funzionalità di I**nserimento** cliccando l'apposito tasto nell'interfaccia di **Ricerca**.



Sommario delle classe usate:

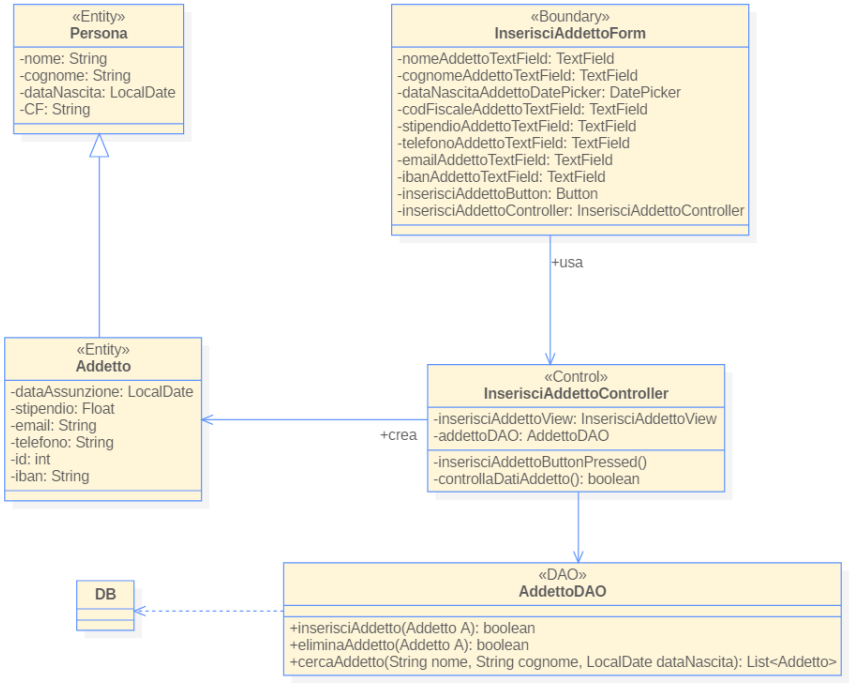
- *InserisciAddettoForm:* Classe di tipo **Boundary**, che rappresenta l'interfaccia con cui interagirà l'**Attore**. In questa interfaccia sarà possibile inserire i dati (nome, cognome, data di nascita, codice fiscale, telefono, e-mail, iban, stipendio) relativi all'addetto da inserire, e di inoltrarli al Database. L'unica altra funzione presente in questa interfaccia sarà quella di poter tornare indietro al menu di Ricerca. Questa classe comunica al **Controller** quali azioni vengono eseguite dall'utente.
- *InserisciAddettoController:* Classe **Control**, controllerà la correttezza dei dati secondo i vincoli che verranno successivamente descritti in dettaglio, e creerà l'addetto con i dati correnti, comunicando con la classe **DAO**.

Le rimanenti classi sono già state descritte nel punto precedente.

Vincoli:

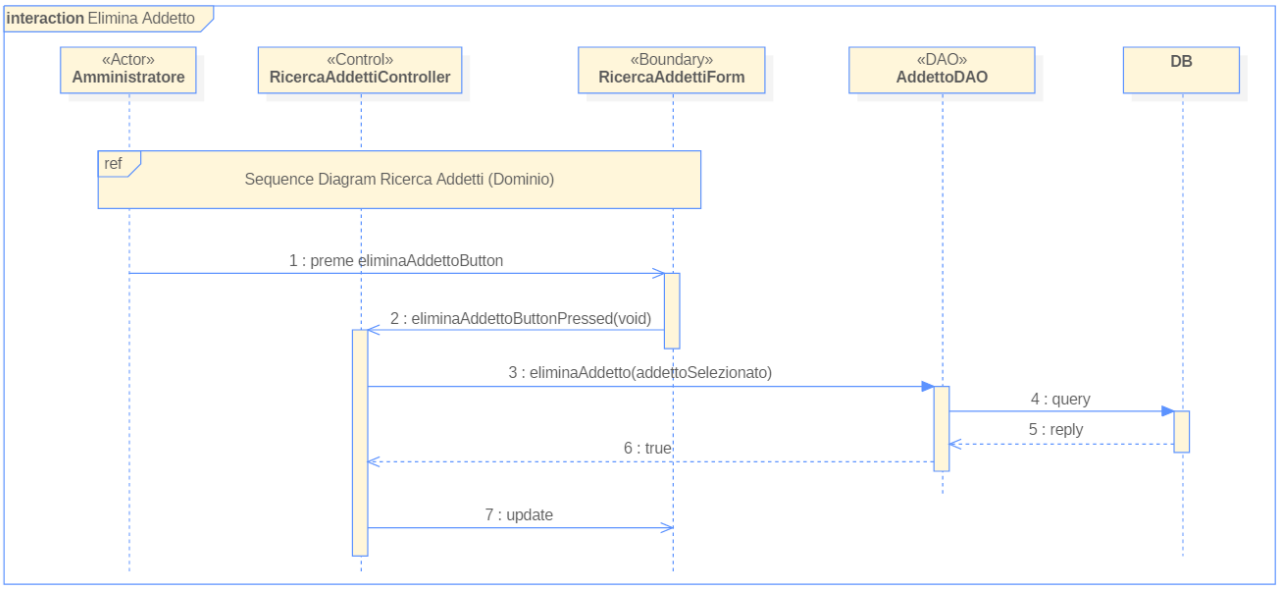
- Prima di poter eliminare o visualizzare un addetto, l'Attore deve averlo selezionato dall'apposita lista di addetti.

Mostriamo le classi trovate in un Class Diagram:



Eliminazione Addetto

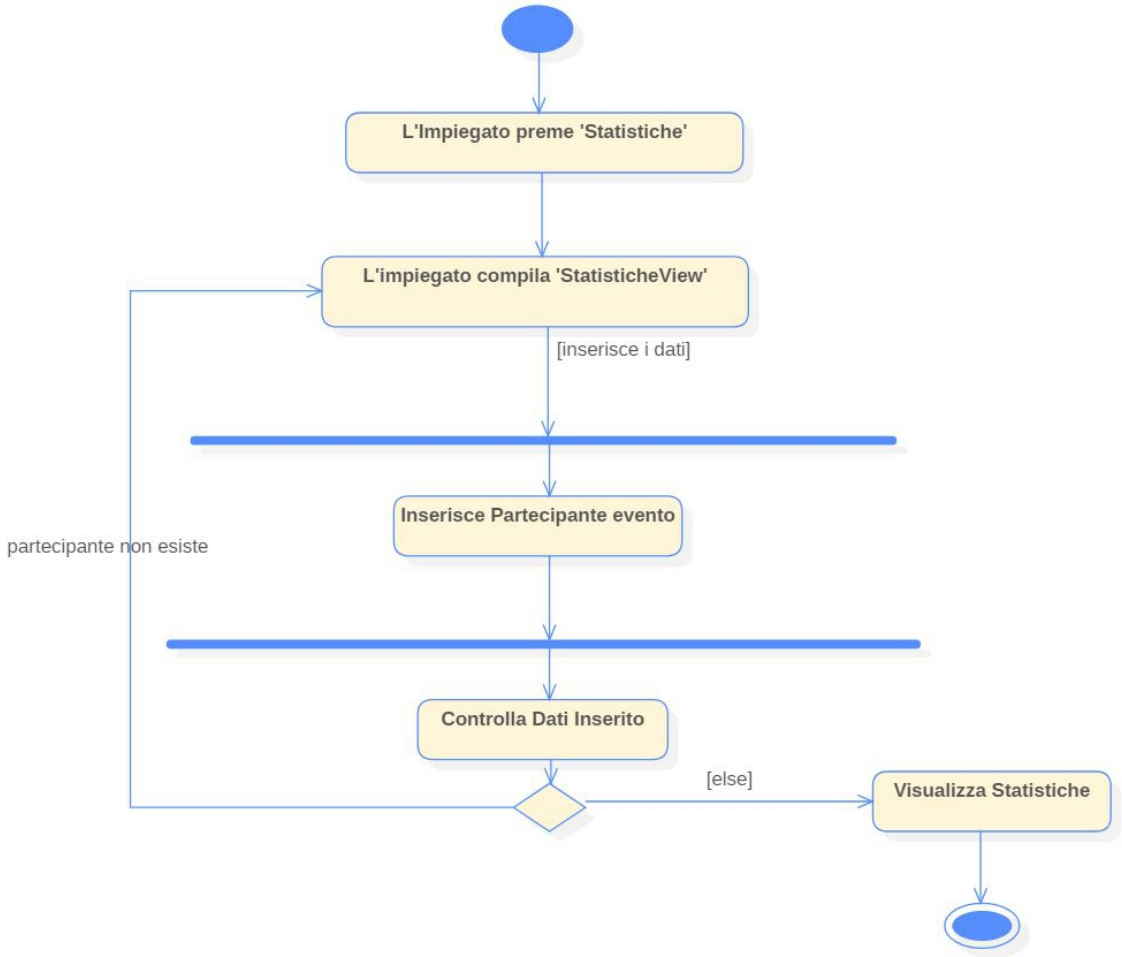
Per questa funzione, ci limitiamo a presentare il relativo Sequence Diagram con le interazioni, in quanto le classi utilizzate sono le stesse dei punti precedenti.
Si potrà accedere alla funzionalità di **Eliminazione** dall'interfaccia di **Ricerca**, cliccando il tasto corrispondente dopo aver selezionato l'**Addetto** da eliminare. Anche in questo caso, quindi, sarà necessario eseguire prima la funzione di ricerca.
Presentiamo il Sequence Diagram:



Statistiche

Come già accennato le statistiche sono state gestite in parte in **Visualizza Clienti** e in parte qui. Per questa funzionalità abbiamo usato la stessa tecnica del Three-Object-Type. Questa funzionalità comprenderà anche la: Visualizzazione delle statistiche di un artista all'evento ed il numero di biglietti venduti nelle città di esibizione.

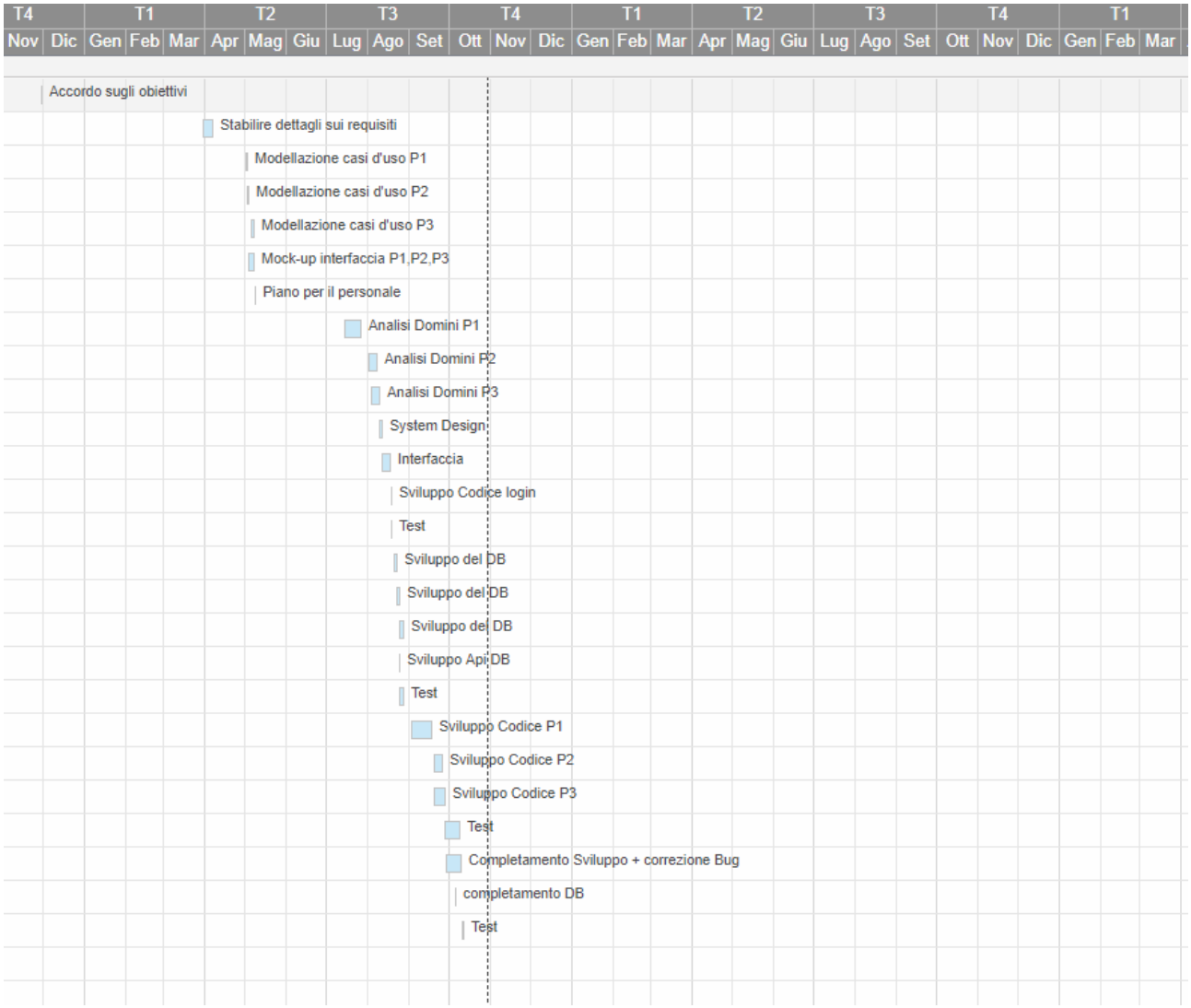
Activity Diagram:



Fa riferimento quindi al *Mock_up_Statistiche*.

Impegno Risorse

Le attività saranno ripartite come mostrato nel seguente diagramma di Gantt:



Il dettaglio con la suddivisione delle attività tra i membri del gruppo, i costi delle attività e le percentuali di completamento è presente in un file Excel all'interno della cartella *Documento dei Requisiti Software*.

Glossario:

- **Impiegato BackOffice**: È 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.