



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Corso di Laurea in Informatica

Progetto di “Ingegneria del Software I” EM ‘17

Customer:

Sergio Di Martino

Team:

INGSW2017_41 composto da:

Alessandro Barra N86/1806

Carlo Carfagna N86/1770

Fabrizio De Sanctis N86/1708

Daniele Di Finizio N86/1801

Sommario

1	Introduzione	5
1.1	Tema.....	5
1.2	Storico Revisioni	7
2	Team Organization	8
2.1	Pianificazione delle attività	8
2.1.1	Gantt Diagram	8
2.1.2	Resources Sharing.....	9
3	Documento dei Requisiti Software	10
3.1.1	Requisiti funzionali	10
3.1.2	Requisiti non funzionali	10
3.1.3	Requisiti di dominio	11
3.2	Use Case Diagrams	12
3.2.1	Use Case Addetto	12
3.2.2	Use Case Amministratore	12
3.2.3	Use Case Utente	13
3.3	Tabelle di Cockburn per Use Cases	14
3.3.1	Addetto.....	14
3.3.2	Amministratore	16
3.3.3	Utente.....	25
3.4	User Interface Mockups	34
3.4.1	Mobile Application	34
3.4.2	Desktop Application.....	36
3.4.3	Web Application	40
3.5	Entity – Boundary – Control Class Diagrams.....	47
3.5.1	EBC Class Diagram – Mobile Application	47
3.5.2	EBC Class Diagram – Desktop Application	48
3.5.3	EBC Class Diagram – Web Application	49
3.6	Sequence Diagrams	50
3.6.1	Inserisci Evento (Desktop Application)	50
3.6.2	Modifica Evento (Desktop Application)	51
3.6.3	Modifica Carrello (Web Application)	52
3.6.4	Aggiungi Evento Carrello (Web Application)	53
3.6.5	Effettua Acquisto (Web Application)	54

3.7	Statechart Diagrams	55
3.7.1	Inserisci Evento (Desktop Application)	55
3.7.2	Aggiungi Evento Carrello (Web Application)	55
3.8	Glossario	56
4	Progettazione Software	57
4.1	System Design	57
4.1.1	Architettura Software	57
4.2	Tecnologie e Software Utilizzati	59
4.2.1	Cloud Services.....	60
4.2.2	Frameworks e Librerie	60
4.3	Object Design	61
4.3.1	System	61
4.3.2	Class Diagrams	62
4.3.3	Sequence Diagrams	65
4.3.4	CRC Cards.....	67
5	Testing.....	78
5.1	System Testing.....	78
5.1.1	Web Application	78
5.1.2	Mobile Application	81
5.1.3	Desktop Application.....	82
5.2	Unit Testing	85
5.2.1	Primo metodo: convertPasswd.....	85
5.2.2	Secondo metodo: getOrdinebyId.....	87

1 Introduzione

1.1 Tema

Questo documento rappresenta la documentazione con raccolta dei requisiti, progettazione e test del prodotto software EM'17. L'analisi è basata su un sottoinsieme di funzionalità selezionato dal cliente, il prof. Sergio Di Martino. In particolare, abbiamo sviluppato questi punti: 1,2,3,4,5 e 6. Per maggiori dettagli sono riportate in seguito le specifiche del prodotto come descritti dal documento "Progetto-17-18_V02B", pubblicato dal cliente.

L'*EM-17 (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.

I principali servizi offerti dal sistema sono 6:

1. Visualizzazione, su un sito web, degli eventi disponibili.
2. Acquisto Biglietto per un evento, dopo la visualizzazione del punto 1.
3. Controllo Accesso all'evento da parte di un addetto alla Security.
4. Gestione degli Eventi
 - a. Inserimento nuovo Evento
 - b. Modifica Evento esistente
 - c. Cancellazione Evento
5. Gestione dei Clienti
 - a. Visualizzazione dati relativi ad un Cliente
 - b. Cancellazione di un Cliente
6. Generazione Statistiche relative ad uno o più eventi
7. Gestione Addetti

Il sistema offre un sito web in cui sono elencati tutti i prossimi eventi per cui è possibile acquistare un biglietto.

Un Cliente, al fine di procedere con l'acquisto, deve preventivamente registrarsi presso il sistema, per ottenere un account. In fase di registrazione l'utente specifica i propri dati anagrafici, una login ed una password.

Dopo la fase di autenticazione, l'utente ha la possibilità di selezionare un evento per cui sono disponibili posti, specificando il numero di biglietti desiderati. Dopo aver inserito i dati della carta di credito, il sistema ne controlla la validità, appoggiandosi ad un servizio esterno offerto da una banca. Qualora il pagamento vada a buon fine, il sistema restituisce al cliente un codice QR univoco che ne permetterà il riconoscimento all'ingresso dell'evento.

Gli addetti al controllo dei biglietti dispongono di terminali mobili per validare un accesso. Tali dispositivi, collegati con un server centrale, permettono la lettura del codice QR di un

avventore, e qualora questo risulti corretto, mostreranno il numero di accessi associati.

Lato Back-Office, un gestore del sistema ha possibilità di effettuare le tipiche operazioni CRUD sulla base di dati, specificando un nuovo evento con data, luogo, prezzo e numero massimo di spettatori. Inoltre ha facoltà di visionare tutti gli eventi già inseriti, con la possibilità di visualizzare informazioni statistiche sotto forma di tabelle e/o grafici (biglietti venduti, incasso, etc...), modificarne i dati salienti, o di cancellare l'evento, qualora non vi siano già biglietti venduti.

Infine, l'amministratore ha possibilità di visionare i dati dei clienti, ancora a fini statistici (numero biglietti acquistati, spesa totale effettuata, etc...), o di cancellarlo.

Le funzionalità 1 e 2 sono disponibili attraverso una web application.

La funzionalità 3 è disponibile in forma di App sul client mobile dato in dotazione agli Operatori sul Campo.

Le funzionalità da 4 a 6 sono disponibili sul Back-Office ad un operatore opportunamente identificato dal sistema.

1.2 Storico Revisioni

Data	Versione	Descrizione
09/06/2018	0.1	Editati il primo e secondo capitolo del documento. Inserito il sommario ed il paragrafo introduttivo per il primo e la descrizione dell'organizzazione del team per il secondo.
20/06/2018	0.1.1	Aggiunto, nel terzo capitolo, l'elenco dei requisiti funzionali, non funzionali e di dominio.
20/06/2018	0.2	Aggiunti i diagrammi dei casi d'uso.
26/06/2018	0.2.1	Aggiunte descrizioni testuali dei casi d'uso (Tabelle di Cockburn).
27/06/2018	0.2.2	Inseriti tutti i mockups.
30/06/2018	0.2.3	Aggiunti al documento i class diagrams EBC.
03/07/2018	0.2.4	Inseriti i sequence diagrams e statechart diagram.
07/07/2018	0.3	Creata sezione "Progettazione software"
30/07/2018	0.3.1	Modifica di alcuni mockups e tabelle di Cockburn.
05/09/2018	0.4	Inseriti diagrammi di design.
10/09/2018	0.4.1	Inseriti class diagram.
16/09/2018	0.4.2	Inseriti sequence diagram.
20/09/2018	0.4.3	Inserite le CRC Cards.
01/10/2018	0.5	Fix di varie sezioni del documento.
10/10/2018	0.5.1	Conclusa sezione di Design.
16/10/2018	0.6	Inserita sezione dedicata al Testing
19/10/2018	1.0	Revisione finale

2 Team Organization

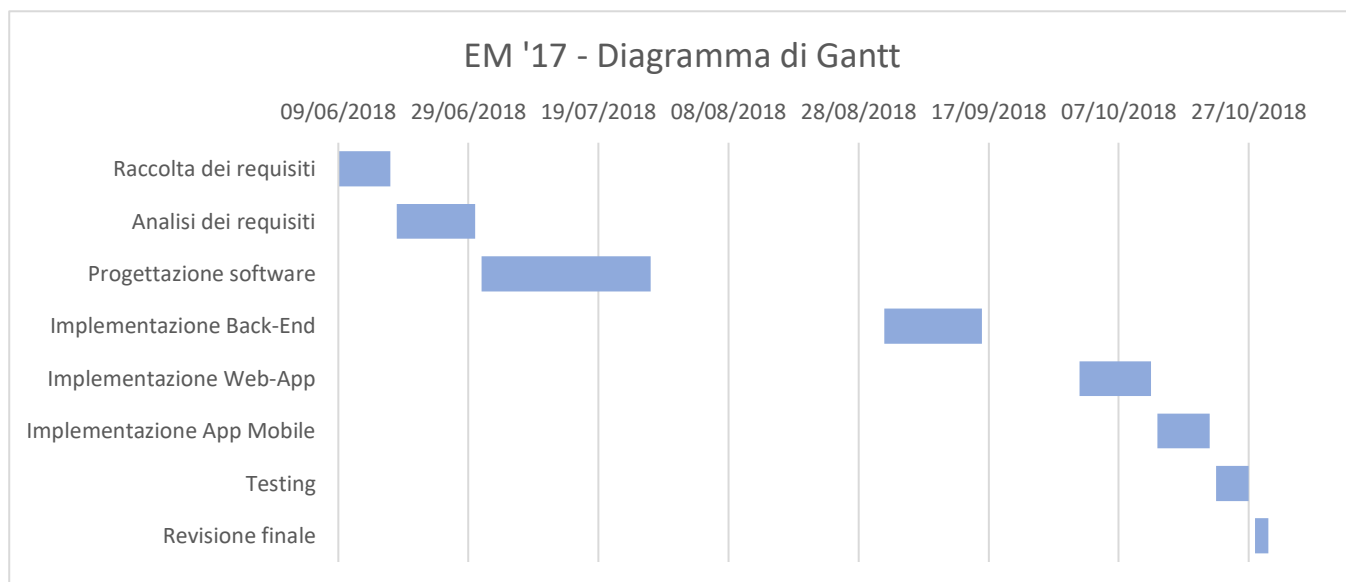
2.1 Pianificazione delle attività

Il diagramma seguente mostra come il team ha deciso di organizzare il lavoro di squadra. Quest'ultimo è stato sviluppato nella fase iniziale del progetto e le scadenze che erano state decise inizialmente non sempre sono state rispettate. IL diagramma è stato creato usando **Microsoft Office Excel**.

Per tutta la durata del progetto, il team si è diviso i compiti da svolgere in piccole task quotidiane. In particolare, ogni giorno ad ogni componente del team venivano assegnate 2/3 task (della stessa difficoltà) da portare a termine entro la fine della giornata. Una volta completate, ogni componente del team si impegnava a verificare, correggere e/o validare il lavoro di un suo collega. Questo ciclo di operazioni ricominciava il giorno successivo. In questo modo abbiamo garantito un carico di lavoro equo per ogni componente del team e allo stesso tempo la correttezza e l'omogeneità delle cose scritte in questo documento.

2.1.1 Gantt Diagram

TASK	DATA INIZIO	DURATA (GIORNI)	DATA FINE
Raccolta dei requisiti	09/06/2018	8	17/06/2018
Analisi dei requisiti	18/06/2018	12	30/06/2018
Progettazione software	01/07/2018	26	27/07/2018
Implementazione Back-End	01/09/2018	15	16/09/2018
Implementazione Web-App	01/10/2018	11	12/10/2018
Implementazione App Mobile	13/10/2018	8	21/10/2018
Testing	22/10/2018	5	27/10/2018
Revisione finale	28/10/2018	2	30/10/2018



2.1.2 Resources Sharing

Le risorse, intese come documenti, immagini e file generali sono stati condivisi utilizzando la piattaforma [Google Drive](#) . La directory è accessibile al seguente indirizzo:

https://drive.google.com/drive/folders/1HU0JvZc_59mM_AGkeARZLz5axAfJrfvf?usp=sharing

Per quanto riguarda il codice, invece, è stato ritenuto più opportuno condividerlo utilizzando il sistema del controllo delle versioni [Git](#) , offerto dalla piattaforma [GitHub](#).

La directory è accessibile al seguente indirizzo: https://github.com/fabrizio-desanctis/INGSW2017_41

3 Documento dei Requisiti Software

3.1.1 Requisiti funzionali

- Il Sistema deve mostrare all'Utente l'elenco degli eventi disponibili.
- Il Sistema deve offrire all'Utente la possibilità di poter effettuare delle ricerche sugli eventi disponibili in modo da filtrarne i risultati in maniera adeguata.
- Il Sistema deve fornire all'Utente un meccanismo di login e/o registrazione.
- Il Sistema deve permettere ad un Utente Autenticato di selezionare un evento per cui sono disponibili posti e di specificare il numero di biglietti desiderati.
- Il Sistema deve offrire la possibilità ad un Utente Autenticato di aggiungere/eliminare gli eventi nel proprio Carrello.
- Un Utente Autenticato può acquistare i biglietti per gli eventi inseriti nel proprio carrello. Il sistema dovrà offrire un servizio per gestire i pagamenti.
- Un Utente Autenticato può accedere ai propri dati personali e modificarli.
- Un Utente Autenticato può accedere alla sua cronologia acquisti e al .PDF relativo ad ogni acquisto.
- Quando un Utente Autenticato porta correttamente a termine la procedura di acquisto, il Sistema deve generare e salvare un documento .PDF con tutte le informazioni sull'acquisto appena effettuato, un QR-Code (uno per ogni biglietto acquistato) valido come tagliando d'accesso e svuotarne il carrello.
- Il Sistema deve permettere ad un Utente Autenticato di poter salvare in locale una copia di ogni proprio .PDF generato e/o di stamparlo.
- Quando un Utente Autenticato non porta a termine correttamente la procedura d'acquisto, il Sistema deve visualizzare una pagina di errore, senza svuotare il carrello.
- Quando un Utente Autenticato non porta a termine o annulla la procedura d'acquisto, il Sistema deve tornare al carrello, senza svuotarlo.
- Un Operatore sul Campo deve avere la possibilità di leggere i QR-Code.
- Il Sistema deve visualizzare, sull'App mobile in dotazione all'Operatore sul Campo, un messaggio di errore nel caso in cui il QR-Code non sia valido oppure sia già stato usato.
- Il Sistema deve visualizzare, sull'App mobile in dotazione all'Operatore sul Campo, una finestra di dialogo con le info relative all'evento del QR-Code, con la possibilità di annullare l'operazione oppure validare l'accesso.
- Il Sistema deve visualizzare, sull'App mobile in dotazione all'Operatore sul Campo, un messaggio di conferma nel caso in cui il QR-Code venga correttamente validato e impostare il biglietto come usato.
- Un Amministratore deve poter accedere al Sistema di Back-End tramite un login.
- Un Amministratore deve avere la possibilità di inserire un nuovo evento.
- Un Amministratore deve avere la possibilità di modificare le informazioni relative ad un evento.
- Un Amministratore deve avere la possibilità di cancellare un evento.
- Un Amministratore deve avere la possibilità di eliminare un utente.
- Un Amministratore deve avere la possibilità di visualizzare i dati di un utente.
- Il Sistema deve generare, per ogni evento disponibile, determinate statistiche che potranno essere visualizzate dall'Amministratore.

3.1.2 Requisiti non funzionali

- Per poter acquistare uno o più biglietti, bisogna essere preventivamente registrati presso il sistema per ottenere un account ed infine effettuare il login.

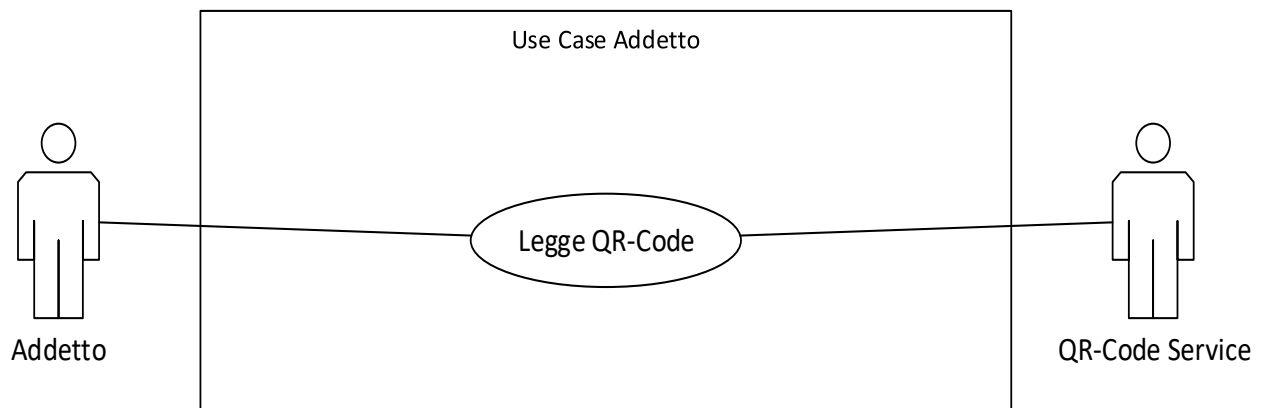
- Il sistema dovrà essere compatibile con i principali browser (Google Chrome, Mozilla Firefox, Internet Explorer).
- Il Sistema deve permettere di essere utilizzato da più utenti in maniera concorrente.
- Il sistema dovrà fornire automaticamente ad ogni nuovo ordine emesso, un identificato numerico in maniera incrementale.
- Il sistema deve permettere la ricerca degli eventi per: tipologia, località, nome e descrizione.
- Per accedere al Sistema di Back-End, l'Amministratore deve effettuare un login.
- I nuovi eventi non possono essere inseriti in date precedenti a quella odierna del Sistema.
- Possono essere cancellati/modificati solo gli eventi con nessun biglietto ancora venduto.
- Il sistema deve fornire per ogni evento le seguenti statistiche: nr. biglietti venduti, incasso, ...

3.1.3 Requisiti di dominio

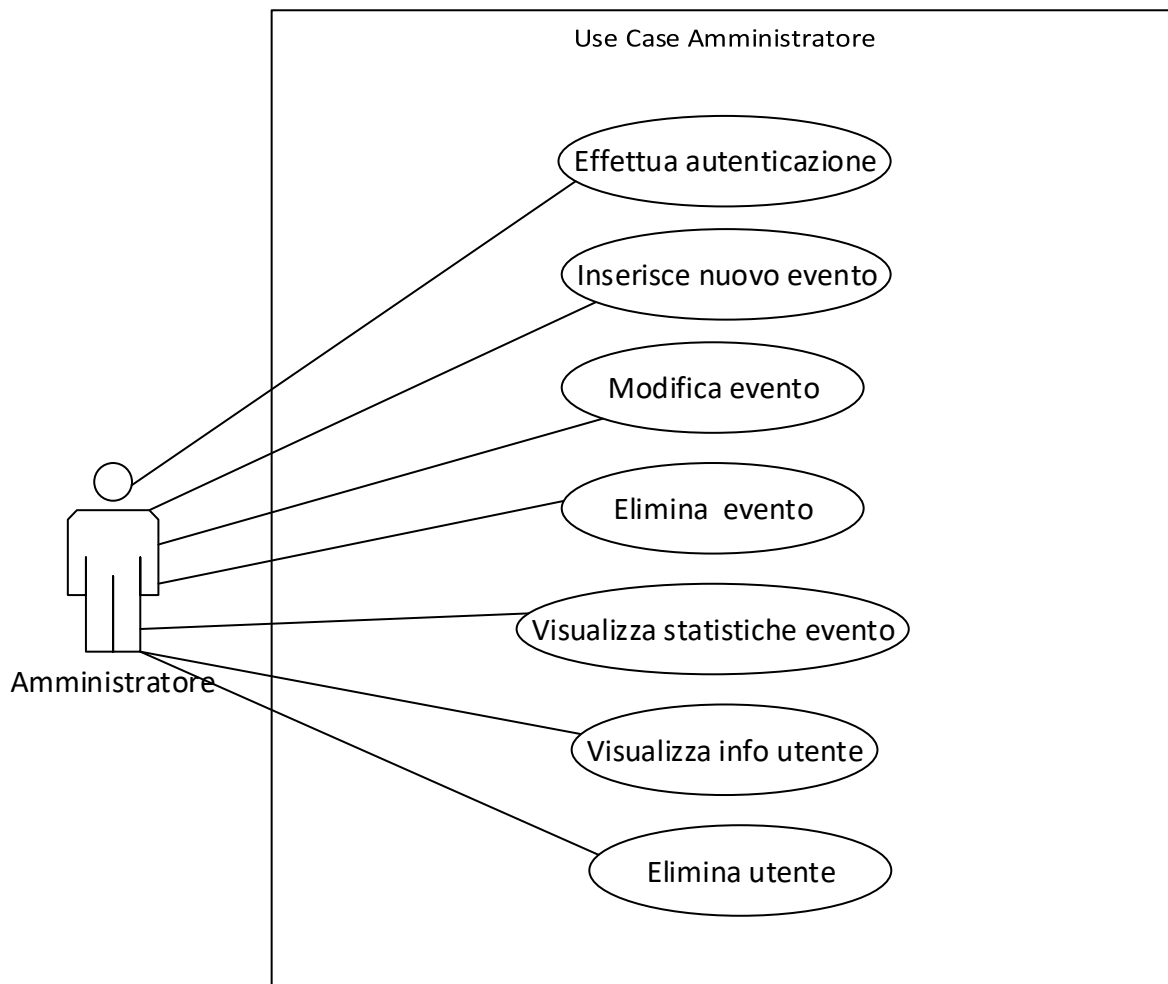
- I biglietti emessi non possono essere cancellati e/o rimborsati.
- Non è possibile vendere più biglietti della capienza massima del luogo che ospita un evento.

3.2 Use Case Diagrams

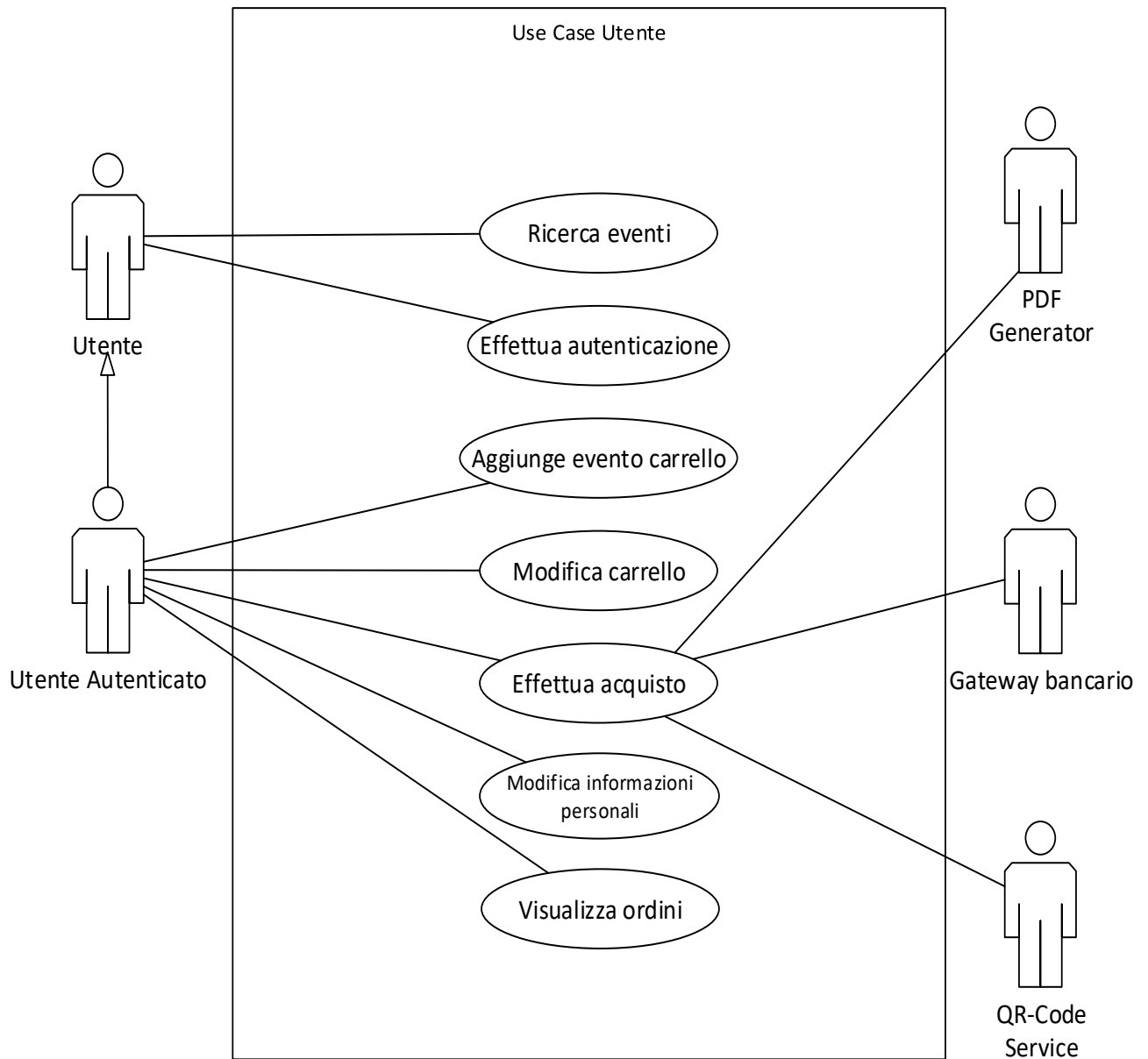
3.2.1 Use Case Addetto



3.2.2 Use Case Amministratore



3.2.3 Use Case Utente



3.3 Tabelle di Cockburn per Use Cases

3.3.1 Addetto

3.3.1.1 Seleziona evento

USE CASE #1	SELEZIONA EVENTO		
Goal in Context	L'Addetto seleziona un evento dalla lista degli eventi disponibili.		
Preconditions	Nessuna.		
Success End Condition	Il Sistema visualizza la pagina specifica relativa all'evento selezionato.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Addetto.		
Trigger	Nessuno.		
DESCRIPTION	Step n°	Addetto	Sistema
	1		Mostra mockup "Mobile_Home"
	2	Clicca su un evento disponibile e su "Avanti"	
	3		Mostra mockup "Mobile_Scanner"
EXTENSIONS	Step	Addetto	Sistema
SUBVARIATIONS	Step	Addetto	Sistema

3.3.1.2 Legge QR-Code

USE CASE #2	<i>LEGGE QR-CODE</i>			
Goal in Context	L'Addetto legge un QR-Code relativo all'evento selezionato.			
Preconditions	Nessuna.			
Success End Condition	Il Sistema accetta il QR-Code.			
Failed End Condition	Il Sistema rimane invariato.			
Primary Actor	Addetto.			
Trigger	L'Addetto seleziona un evento da mockup "Mobile_Elenco_Ordini".			
DESCRIPTION	Step n°	Addetto	Sistema	QR-Code Service
	1		Mostra mockup "Mobile_Scanner"	
	2	Scannerizza il QR-Code.		
	3		Acquisisce il QR-Code	
	4			Verifica QR-Code
	5		Mostra mockup messaggio di conferma e torna allo step #1	
EXTENSIONS	<i>Step</i>	Addetto	Sistema	
	2.1	<i>Clicca su "Indietro"</i>		
	3.1		<i>Mostra mockup "Mobile_Home"</i>	
SUBVARIATIONS	<i>Step</i>	Addetto	Sistema	
	5.1		<i>Mostra messaggio di errore e torna allo step #1</i>	

3.3.2 Amministratore

3.3.2.1 Effettua autenticazione

USE CASE #3	<i>EFFETTUA AUTENTICAZIONE</i>		
Goal in Context	L'Amministratore accede all'area di back-end.		
Preconditions	Nessuna.		
Success End Condition	Il Sistema riconosce l'username e la password dell'Amministratore, permettendogli l'accesso all'area di Back-End.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Amministratore.		
Trigger	Nessuno.		
DESCRIPTION	Step n°	Amministratore	Sistema
	1		Mostra mockup "Back_End_Login"
	2	Compila il form, inserendo username e password e infine clicca su "Accedi"	
	3		Verifica i dati inseriti, accetta la richiesta e mostra mockup "Back_End_Home"
EXTENSIONS	<i>Step</i>	Amministratore	Sistema
SUBVARIATIONS	<i>Step</i>	Amministratore	Sistema
	3.1		<i>Mostra messaggio errore e torna allo step #1</i>

3.3.2.2 Inserisce nuovo evento

USE CASE #4	<i>INSERISCE NUOVO EVENTO</i>		
Goal in Context	L'Amministratore inserisce un nuovo evento disponibile.		
Preconditions	Nessuna.		
Success End Condition	Il Sistema aggiunge un nuovo all'elenco degli eventi disponibili.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Amministratore.		
Trigger	L'Amministratore clicca su "Inserire nuovo evento" dal mockup "Back_End_Home"		
DESCRIPTION	Step n°	Amministratore	Sistema
	1		Mostra mockup "Back_End_Nuovo_Evento"
	2	Compila il form e clicca su "Conferma"	
	3		Verifica i dati inseriti, accetta la richiesta, mostra messaggio di conferma e torna a mockup "Back_End_Home"
EXTENSIONS	<i>Step</i>	Amministratore	Sistema
	2.1	Clicca su "Indietro"	
	3.1		Chiede conferma e mostra mockup "Back_End_Home"
SUBVARIATIONS	<i>Step</i>	Amministratore	Sistema
	3.1		<i>Verifica i dati inseriti, rifiuta la richiesta, mostra un messaggio relativo all'errore e torna al #1.</i>

3.3.2.3 Modifica evento

USE CASE #5	<i>MODIFICA EVENTO</i>		
Goal in Context	L'Amministratore modifica le informazioni relative ad un evento già esistente.		
Preconditions	Non ci sono ancora biglietti venduti per il relativo evento.		
Success End Condition	Il Sistema aggiorna permanentemente le informazioni richieste.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Amministratore.		
Trigger	L'Amministratore clicca su "Modifica evento" dal mockup "Back_End_Home"		
DESCRIPTION	Step n°	Amministratore	Sistema
	1		Mostra mockup "Back_End_Seleziona_Evento"
	2	Seleziona un evento dall'elenco	
	3		Abilita tasto "Avanti"
	4	Clicca su "Avanti"	
	5		Mostra mockup "Back_End_Modifica_Evento"
	6	Modifica il form e clicca su "Conferma"	
	7		<i>Verifica i dati inseriti, accetta la richiesta e mostra un messaggio di conferma, tornando al mockup "Back_End_Home"</i>
EXTENSIONS	<i>Step</i>	Amministratore	Sistema
	2.1 4.1	Clicca su "Indietro"	

	3.1 5.1		Mostra mockup "Back_End_Home"
SUBVARIATIONS	<i>Step</i>	Amministratore	Sistema
	6.1	Clicca su "Indietro"	
	7.1		<i>Torna a #1.</i>
	7.2		<i>Verifica i dati inseriti, rifiuta la richiesta, mostra un messaggio relativo all'errore e torna al #1.</i>

3.3.2.4 Elimina evento

USE CASE #6	<i>ELIMINA EVENTO</i>		
Goal in Context	L'Amministratore elimina un evento esistente.		
Preconditions	Non ci sono ancora biglietti venduti per il relativo evento.		
Success End Condition	Il Sistema rimuove l'evento e tutte le informazioni ad esso associate in maniera permanente.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Amministratore.		
Trigger	L'Amministratore clicca su "Elimina evento" dal mockup "Back_End_Home"		
DESCRIPTION	Step n°	Amministratore	Sistema
	1		Mostra mockup "Back_End_Seleziona_Evento"
	2	Seleziona un evento dall'elenco	
	3		Abilita tasto "Avanti"
	4	Clicca su "Avanti"	
	5		Accetta la richiesta, mostra messaggio di conferma e torna a mockup "Back_End_Home"
EXTENSIONS	<i>Step</i>	Amministratore	Sistema
	2.1 4.1	Clicca su "Indietro"	
	3.1 5.1		Mostra mockup "Back_End_Home"

SUBVARIATIONS	Step	Amministratore	Sistema
	5.2		<i>Rifiuta la richiesta, mostra un messaggio relativo all'errore e torna al #1.</i>

3.3.2.5 Visualizza statistiche

USE CASE #7	VISUALIZZA STATISTICHE		
Goal in Context	L'Amministratore visualizza le statistiche relative ad un evento.		
Preconditions	Nessuna,		
Success End Condition	Il Sistema genera le statistiche relative ad un evento selezionato.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Amministratore.		
Trigger	L'Amministratore clicca su "Visualizza statistiche" dal mockup "Back_End_Home"		
DESCRIPTION	Step n°	Amministratore	Sistema
	1		Mostra mockup "Back_End_Seleziona_Evento"
	2	Seleziona un evento dall'elenco	
	3		Abilita tasto "Avanti"
	4	Clicca su "Avanti"	
	5		Mostra mockup "Back_End_Statistiche"
EXTENSIONS	Step	Amministratore	Sistema
	2.1 4.1	Clicca su "Indietro"	
	3.1 5.1		Mostra mockup "Back_End_Home_Page"
SUBVARIATIONS	Step	Amministratore	Sistema

3.3.2.6 Visualizza dati / Elimina utente

USE CASE #8	VISUALIZZA DATI/ELIMINA UTENTE		
Goal in Context	L'amministratore elimina un utente.		
Preconditions	Nessuna.		
Success End Condition	Il Sistema rimuove l'utente e tutte le informazioni ad esso associate in maniera permanente.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Amministratore.		
Trigger	L'Amministratore clicca su "Visualizza/Elimina utente" dal mockup "Back_End_Home"		
DESCRIPTION	Step n°	Amministratore	Sistema
	1		Mostra lista di utenti registrati.
	2	Seleziona un utente dall'elenco	
	3		Abilita tasto "Elimina"
	4	Clicca su "Elimina"	
	5		Accetta la richiesta, mostra messaggio di conferma e torna a mockup "Back_End_Home"
EXTENSIONS	<i>Step</i>	Amministratore	Sistema
	2.1 4.1	Clicca su "Indietro"	
	3.1 5.1		Mostra mockup "Back_End_Home"

SUBVARIATIONS	Step	Amministratore	Sistema
	5.2		<i>Rifiuta la richiesta, mostra un messaggio relativo all'errore e torna al #1.</i>

3.3.3 Utente

3.3.3.1 Ricerca Eventi

USE CASE #9	RICERCA EVENTI		
Goal in Context	L'utente visualizza gli eventi relativi alla query effettuata		
Preconditions	Nessuna.		
Success End Condition	Il Sistema rimane invariato.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Utente		
Trigger	Nessuno.		
DESCRIPTION	Step n°	Utente	Sistema
	1		Mostra mockup "Web_Home_Page"
	2	Inserisce delle keywords nella barra di ricerca	
	3	Clicca su "Cerca"	
	4		Mostra pagina con l'elenco degli eventi relativi alle keywords inserite.
	5		
SUBVARIATIONS	Step n°	Utente	
	4		Mostra mockup "Web_Query_Error"

3.3.3.2 Effettua autenticazione

USE CASE #10	<i>EFFETTUA AUTENTICAZIONE</i>		
Goal in Context	L'utente effettua l'accesso con le proprie credenziali.		
Preconditions	L'utente deve essere già registrato.		
Success End Condition	Il sistema rimane invariato.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Utente		
Trigger	L'utente clicca su "Login"		
DESCRIPTION	Step n°	Addetto	Sistema
	1		Mostra mockup "Web_Login"
	2	Compila il form di autenticazione con email e password e clicca su "Accedi"	
	3		Controlla la presenza e la correttezza nel database dei dati inseriti dall'utente. Mostra mockup "Web_Home_Page" se va a buon fine
EXTENSIONS	<i>Step</i>	Addetto	Sistema
	2.1	Clicca su "Registrati"	
	3.1		Mostra mockup "Web_Registrazione"
SUBVARIATIONS	<i>Step</i>	Addetto	Sistema
	3.3		<i>Mostra un messaggio di errore se l'autenticazione non va a buon fine e ritorna allo step #1</i>

3.3.3.3 Aggiunge evento carrello

USE CASE #10	AGGIUNGE EVENTO CARRELLO		
Goal in Context	L'Utente Autenticato seleziona un evento e lo aggiunge al carrello.		
Preconditions	L'Utente deve autenticarsi.		
Success End Condition	Il Sistema aggiorna il Carrello relativo all'Utente Autenticato, aggiungendo l'evento selezionato.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Utente Autenticato.		
Trigger	L'Utente Autenticato seleziona un evento dalla Home Page oppure dopo aver effettuato una ricerca.		
DESCRIPTION	Step n°	Utente	Sistema
	1	Seleziona un evento	
	2		Mostra mockup "Web_Pagina_Evento"
	3	Setta il numero di biglietti e clicca "Aggiungi al carrello"	
	4		Se l'utente è autenticato, aggiunge evento al Carrello e mostra "Web_Home_Page"
EXTENSIONS	<i>Step</i>	Utente	Sistema
	3.2	<i>Clicca su Carrello</i>	
	4.2		Mostra mockup "Web_Carrello"
SUBVARIATIONS	<i>Step</i>	Utente	Sistema

3.3.3.4 Modifica carrello

USE CASE #11	MODIFICA CARRELLO		
Goal in Context	L'Utente Autenticato modifica il proprio Carrello.		
Preconditions	L'Utente deve essere autenticato.		
Success End Condition	Il Sistema aggiorna le quantità e il prezzo totale relativi agli eventi presenti nel carrello.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Utente Autenticato.		
Trigger	L'Utente seleziona l'icona Carrello.		
DESCRIPTION	Step n°	Utente	Sistema
	1		Mostra mockup "Web_Carrello" se l'utente è autenticato
	2	Rimuove evento	
	3		Accoglie la richiesta, eliminando l'evento dal carrello e mostra "Web_Home_Page"
EXTENSIONS	<i>Step</i>	Utente	Sistema
	2.2	<i>Clicca su Carrello</i>	
	3.2		Mostra mockup "Web_Carrello"
SUBVARIATIONS	<i>Step</i>	Utente	Sistema
	2.3	Cerca un nuovo evento	
	3.3		Mostra elenco eventi

3.3.3.5 Effettua acquisto

USE CASE #12	EFFETTUA ACQUISTO					
Goal in Context	L'Utente Autenticato acquista n-biglietti.					
Preconditions	L'Utente deve aver aggiunto elementi nel proprio Carrello.					
Success End Condition	Il Sistema genera un nuovo ordine ed emette i biglietti in formato .pdf					
Failed End Condition	Il Sistema rimane invariato.					
Primary Actor	Utente Autenticato.					
Trigger	L'Utente Autenticato clicca su "Procedi al pagamento"					
DESCRIPTION	Step n°	Utente	Sistema	Gateway bancario	PDF Generator	QR-Code Service
	1	Clicca su PayPal oppure paga con carta				
	2		Avvia servizio pagamento esterno			
	3			Inizializza pagamento		

	4	Inserisce dati per il pagamento e clicca su "Conferma"				
	5			Effettua verifiche sulla transazione		
	6		Se il pagamento è andato a buon fine, avvia servizi PDF-Generator.			
	7				Richiede QR-Code	
	8					Genera e fornisce QR-Code
	9				Acquisisce QR-Code e genera documento con tutte le informazioni	
	10		Mostra mockup "Web_Pagamento_effettuato"			
EXTENSIONS	Step	Utente	Sistema			
	1.1	Clicca su logo "EM'17" oppure su "Indietro"				
	2.1		Mostra mockup "Web_Home_Page"			
SUBVARIATIONS	Step	Amministratore	Sistema			

	6.2		Se il pagamento non va a buon fine, visualizza mockup "Web_Errore_Pagamento"			
--	-----	--	------------------------------------------------------------------------------	--	--	--

3.3.3.6 Modifica informazioni personali

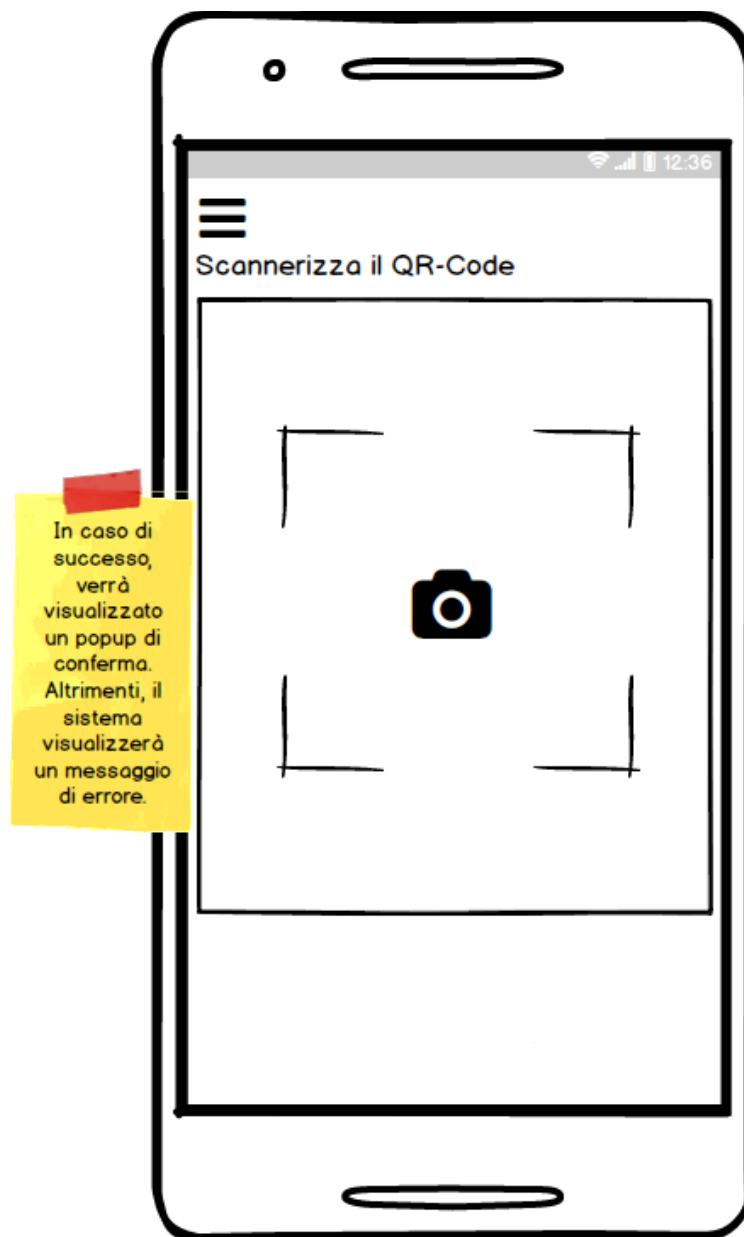
USE CASE #13	<i>MODIFICA INFORMAZIONI PERSONALI</i>		
Goal in Context	L'Utente Autenticato visualizza i suoi dati inseriti al momento della registrazione e può eventualmente modificarli		
Preconditions	L'utente deve essere registrato		
Success End Condition	L'utente riesce a visualizzare/modificare i suoi dati		
Failed End Condition	Nessuna.		
Primary Actor	Utente Autenticato		
Trigger	Nessuno.		
DESCRIPTION	Step n°	Utente Autenticato	Sistema
	1		Mostra mockup "Web_Change_Data"
	2	Clicca su "Modifica informazioni personali"	
	3		Abilita la modifica
	4	Modifica i dati interessati attraverso i campi di testo	
	5		Accetta la richiesta.
SUBVARIATIONS	Step n°	Utente	
	5.1		Mostra mockup "Web_Change_Edited" se si presenta un errore e torna al punto 3.

3.3.3.7 Visualizza ordini

USE CASE #14	<i>VISUALIZZA ORDINI</i>		
Goal in Context	L'utente visualizza lo storico degli ordini effettuati.		
Preconditions	L'utente deve essere autenticato .		
Success End Condition	Il sistema rimane invariato.		
Failed End Condition	Il Sistema rimane invariato.		
Primary Actor	Utente Autenticato.		
Trigger	L'Utente clicca su "I miei ordini" dal mockup "Web_Profilo"		
DESCRIPTION	Step n°	Utente Autenticato	Sistema
	1		Esegue query per reperire tutti gli ordini effettuati dall'utente.
	2		Mostra mockup "Web_Riepilogo_ordini"
EXTENSIONS	<i>Step</i>	Utente Autenticato	Sistema
	3	<i>Clicca su Carrello</i>	
	4		Mostra mockup "Web_Carrello"
	3.2	Cerca un nuovo evento	
	4.2		Mostra elenco eventi.
SUBVARIATIONS	<i>Step</i>	Utente Autenticato	Sistema

3.4 User Interface Mockups

3.4.1 Mobile Application

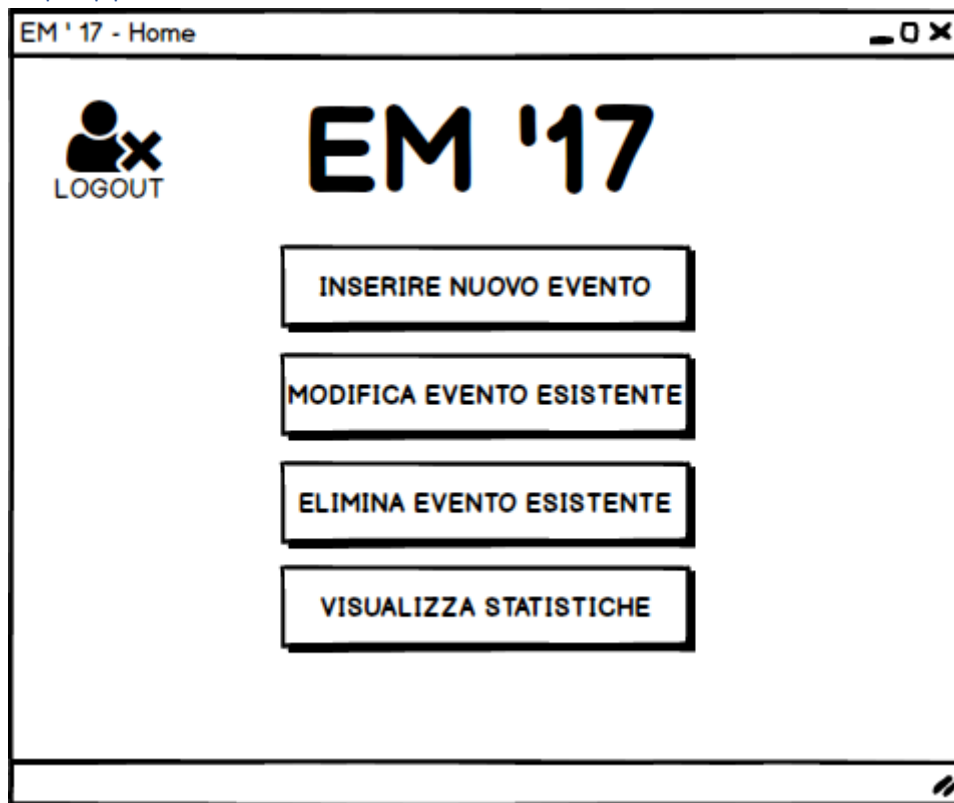


Mockup 1 - Mobile_Home

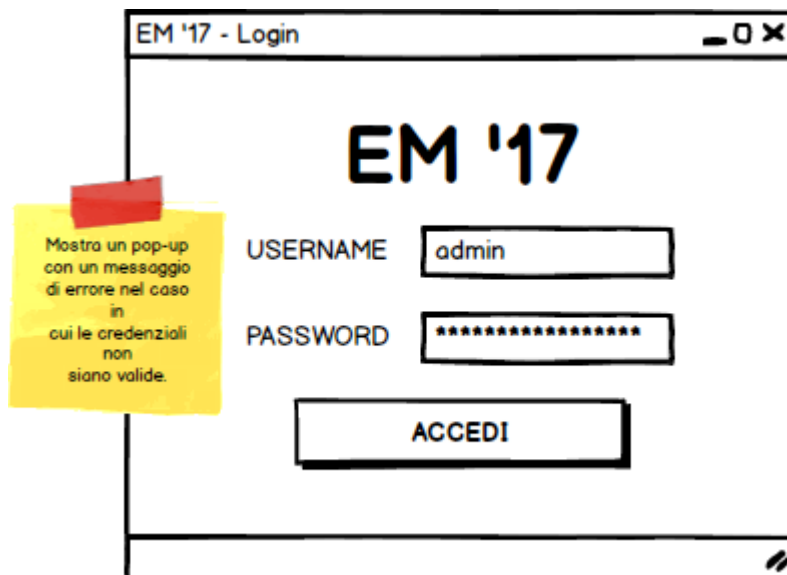


Mockup 2- Mobile_Scanner

3.4.2 Desktop Application



Mockup 3 - Back_End_Home



Mockup 4 - Back_End_Login

EM '17 - Nuovo evento

"EM '17"

AGGIUNGI NUOVO EVENTO

Tipologia:

Concerti

Nome evento:

Data:

/ /

Orario:

Ora

Minuti

Descrizione:

Link:

Luogo:

Località:

Località

Prezzo:

Nr. biglietti:

ANNULLA

CONFERMA

EM '17 - Nuovo evento

"EM '17"

MODIFICA EVENTO

Tipologia:

Sport ▼

Nome evento:

Napoli - Juventus

Data:

◀ JUNE 2018 ▶

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Orario:

20 ▼

45 ▼

Luogo:

Stadio San Paolo

Località:

Napoli ▼

Prezzo:

30€

Nr. biglietti:

60000 ▴▾

Descrizione:

15a Giornata del campionato di Serie A TIM 2018/2019

INDIETRO

CONFERMA

EM '17 - Seleziona evento

"EM '17"

SELEZIONA EVENTO

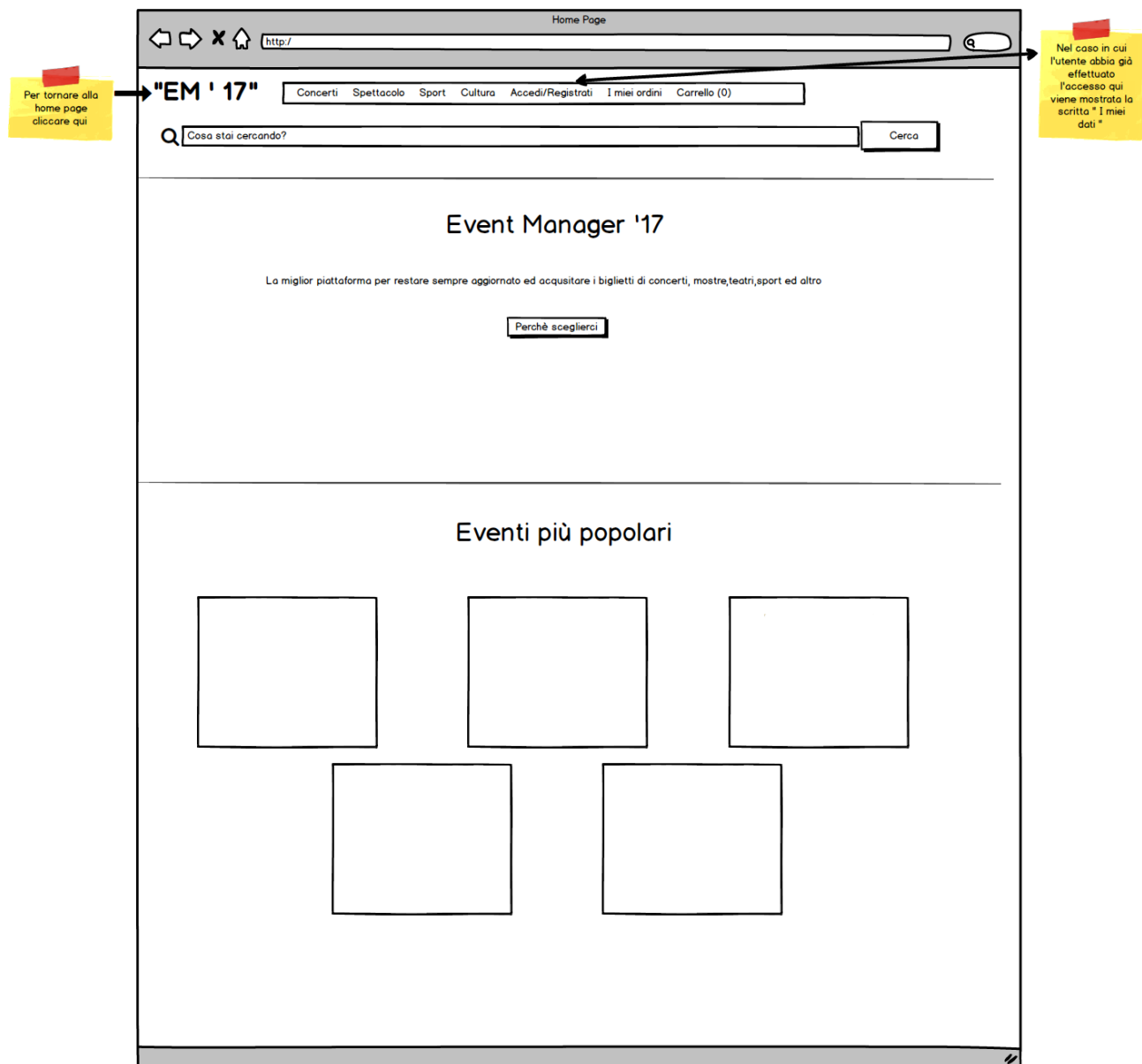
Nome evento	Data
Evento 1	23/04/2019
Evento 2	05/09/2020
Evento 3	25/12/2019
Evento 4	23/04/2019
Evento 5	05/09/2020
Evento 6	25/12/2019

INDIETRO

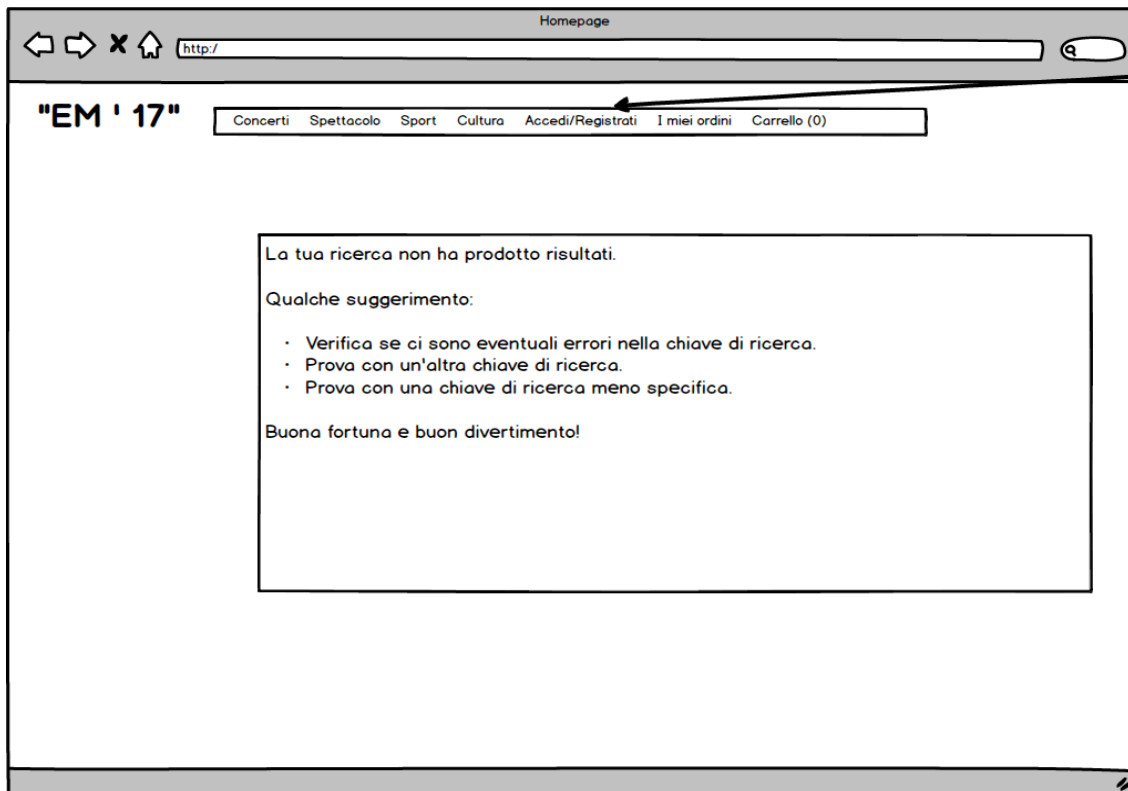
AVANTI

Mockup 7 - Back_End_Seleziona_Evento

3.4.3 Web Application

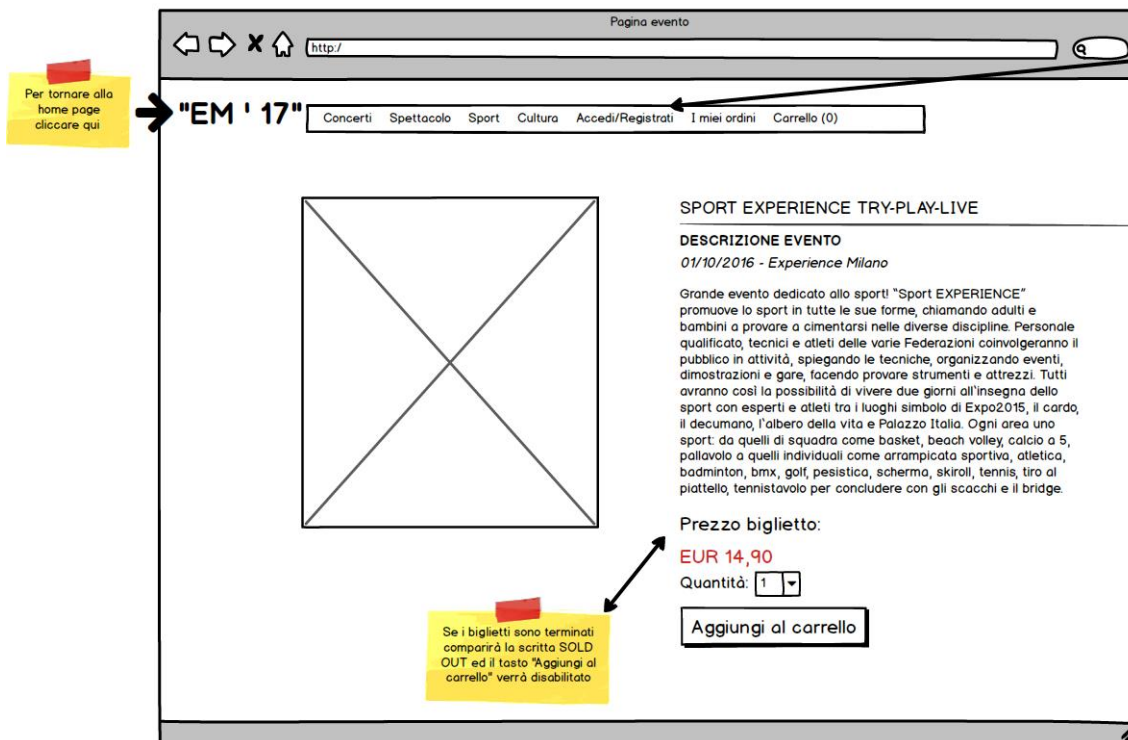


Mockup 8 - Web_Home_Page



Nel caso in cui l'utente abbia già effettuato l'accesso qui viene mostrata la scritta "I miei dati"

Mockup 9- Web_Query_Error



Nel caso in cui l'utente abbia già effettuato l'accesso qui viene mostrata la scritta "I miei dati"

Mockup 10- Web_Pagina_Evento

The mockup shows a web browser window with the title "Home Page" and a URL bar containing "http://". The page content is titled "Registrazione" and is divided into two main sections: "Dati anagrafici" and "Dati accesso".

Dati anagrafici:

- Nome:
- Cognome:
- Telefono:
- Sesso:
- Data di nascita:
- Nazionalità:
- Indirizzo:
- Città:
- Provincia:
- CAP:
- Indirizzo:
- Numero civico:

Dati accesso:

- Indirizzo email:
- Password:
- Conferma Password:


At the bottom of the form is a button labeled "Conferma registrazione".


Annotations on the mockup:

- Left side: A yellow box with a red arrow pointing to the "EM '17" logo, containing the text "Per tornare alla home page cliccare qui".
- Right side: A yellow box with a red arrow pointing to the search bar, containing the text "Nel caso in cui l'utente abbia già effettuato l'accesso qui viene mostrata la scritta 'I miei dati'".


Mockup 11 - Web_Registrati

EM '17 - Accedi





"EM '17"


Home Page

NOME UTENTE/E-MAIL

PASSWORD

Accedi

Registrati

Non riesci ad accedere?

Mockup 12 - Web_Login

Cambia Dati

http://

In questa pagina l'utente può sia visualizzare i suoi dati che modificarli

Tipologia cliente ☒ Signor ☐ Signora

Nome

Cognome

Via e n° civico


Paese

Provincia

Cap, città

Indirizzo email

Ripeti Indirizzo email

Data di nascita 

Telefono

Codice fiscale

Paese di nascita

Salva e procedi ►

Mockup 13 - Web_Change_Data

Cambia Dati

http://

In questa pagina l'utente può sia visualizzare i suoi dati che modificarli

Tipologia cliente ☒ Signor ☐ Signora

Nome

Cognome

Via e n° civico


Paese

Provincia

Cap, città

Indirizzo email

Ripeti Indirizzo email

Data di nascita 

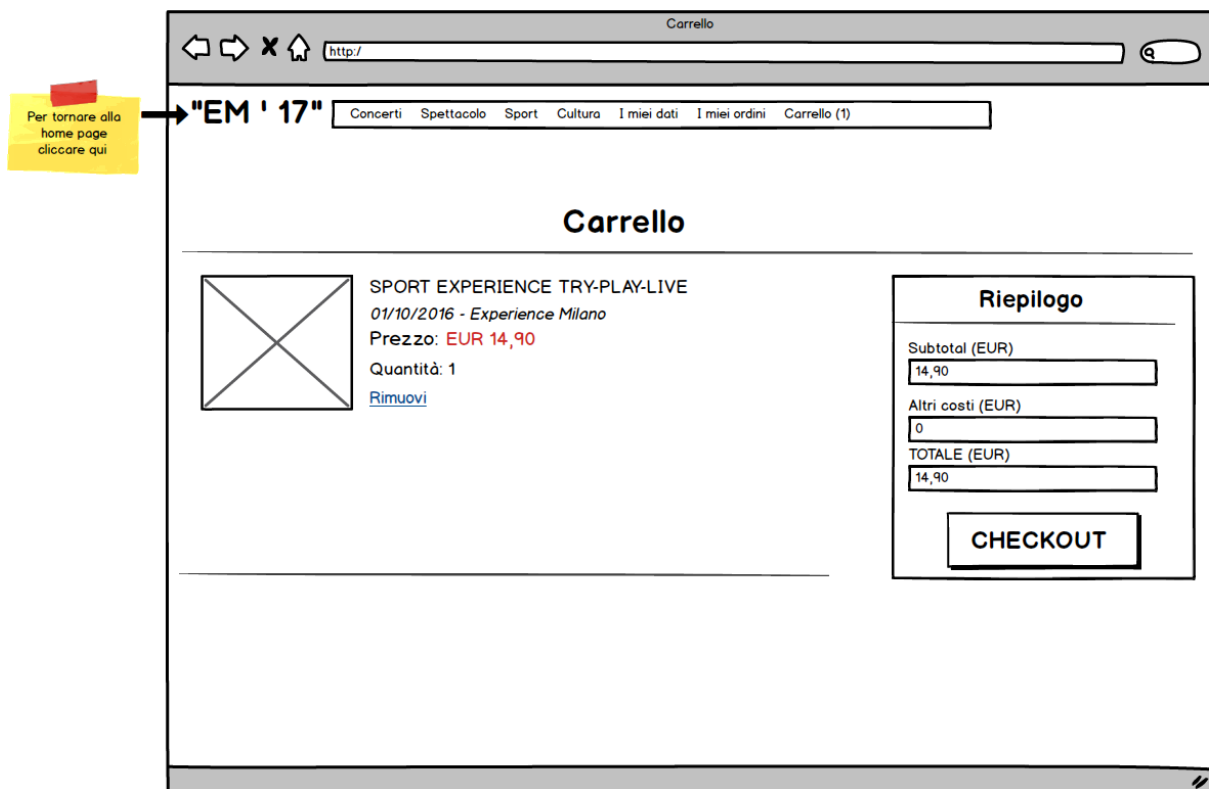
Telefono

Codice fiscale

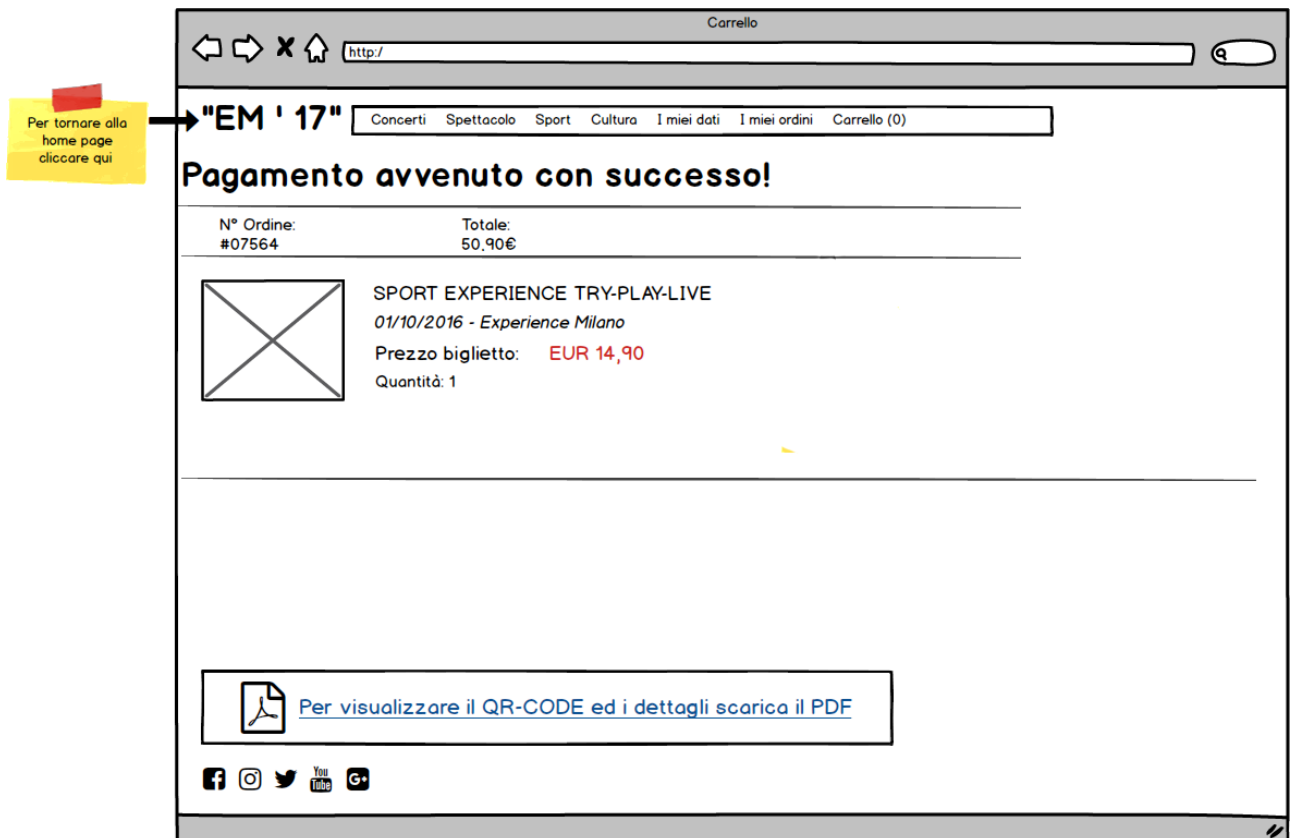
Paese di nascita

Salva e procedi ►

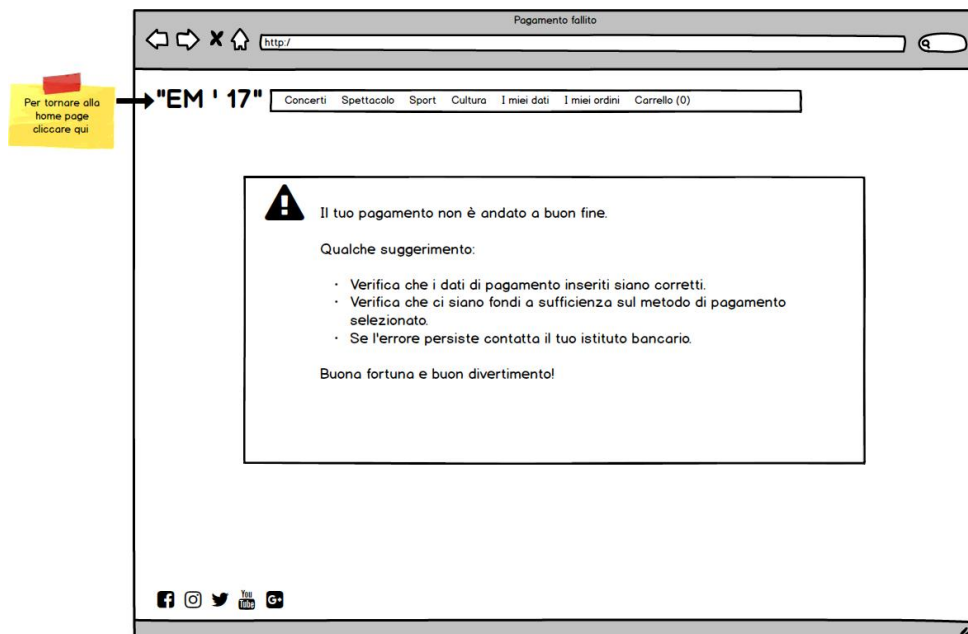
Mockup 14 - Web_Change_Data_Edited



Mockup 15 - Web_Carrello



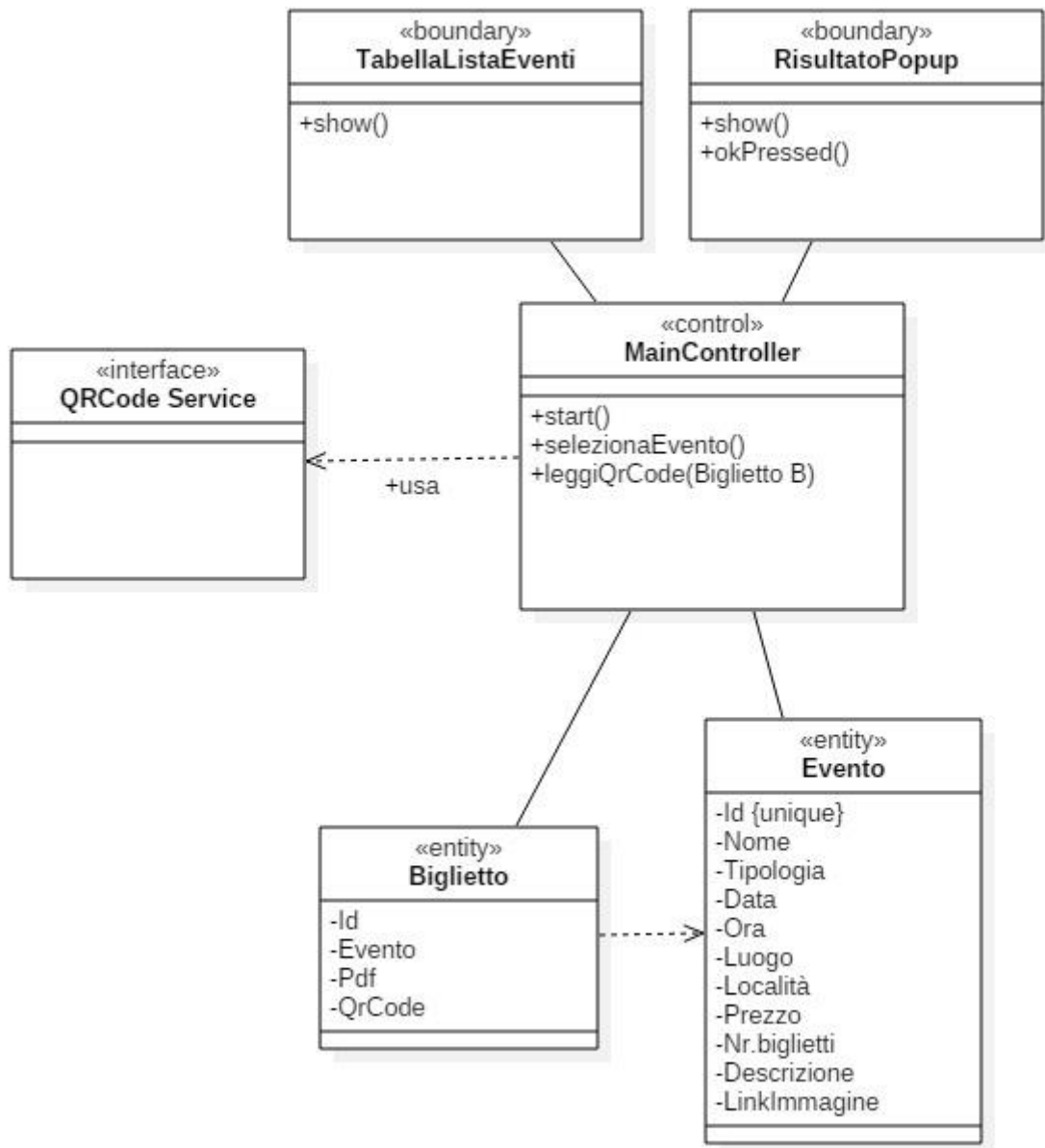
Mockup 16 - Web_Pagamento_effettuato



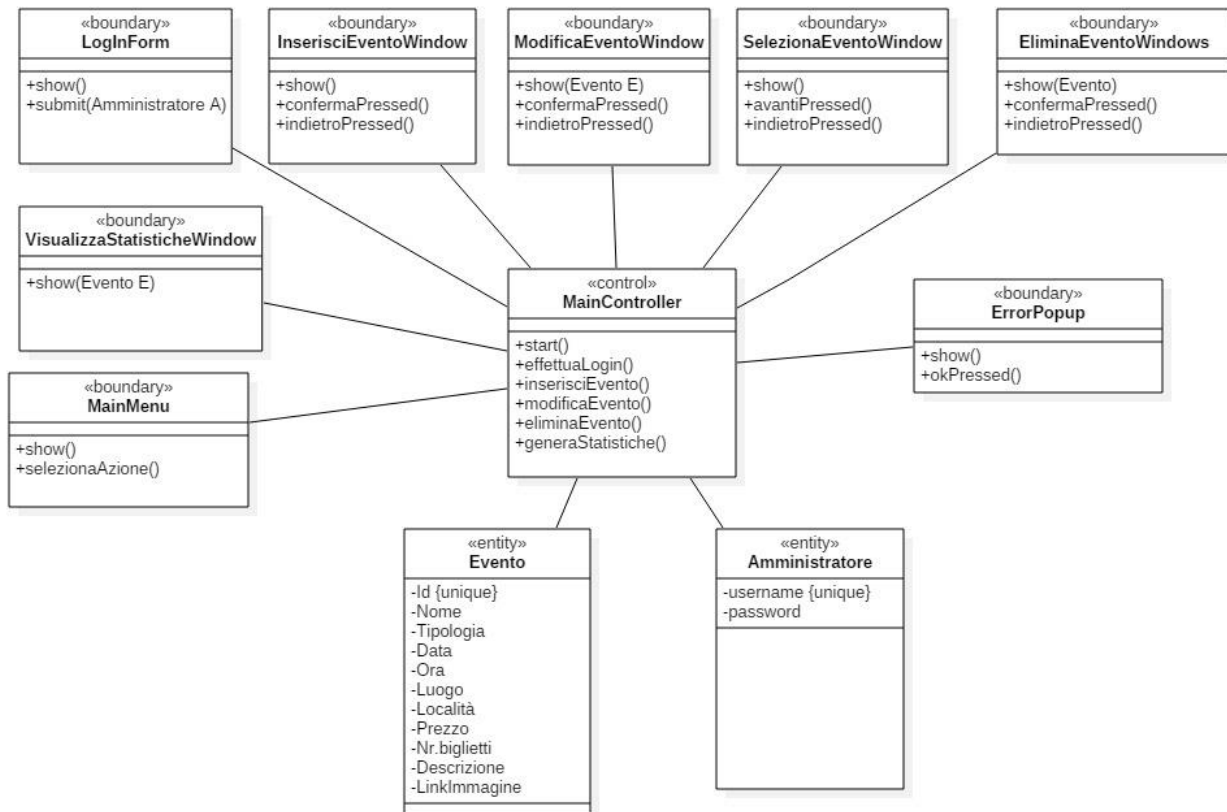
Mockup 17 - Web_Pagamento_fallito

3.5 Entity – Boundary – Control Class Diagrams

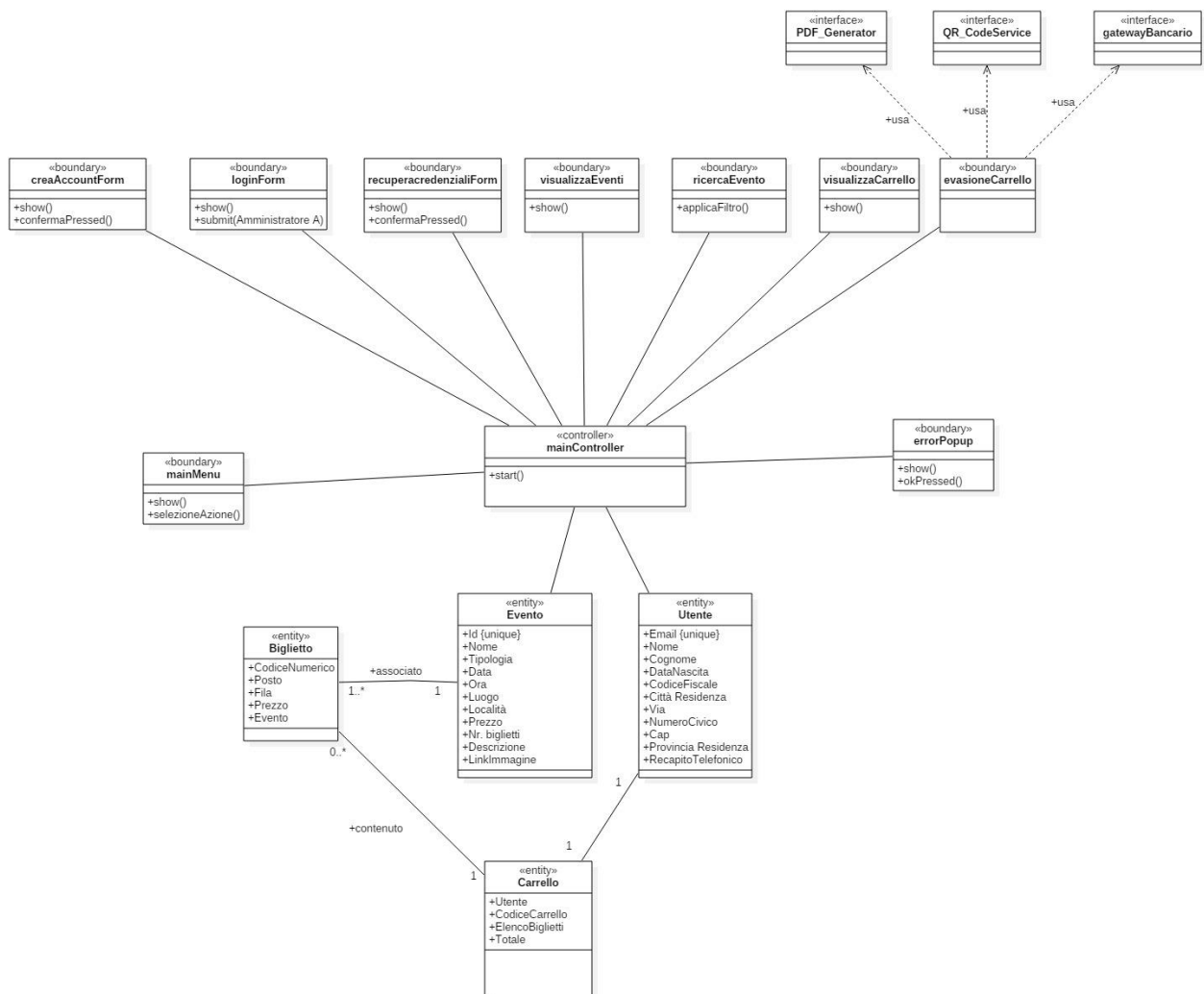
3.5.1 EBC Class Diagram – Mobile Application



3.5.2 EBC Class Diagram – Desktop Application

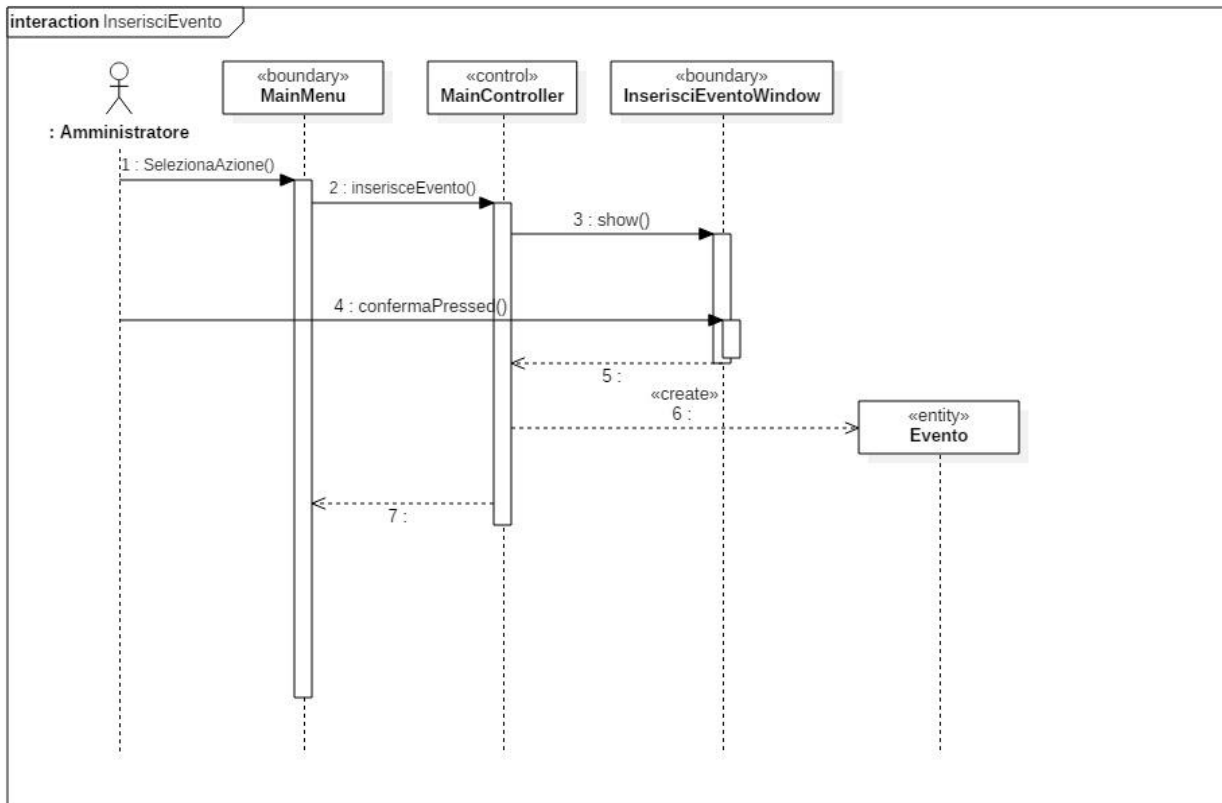


3.5.3 EBC Class Diagram – Web Application

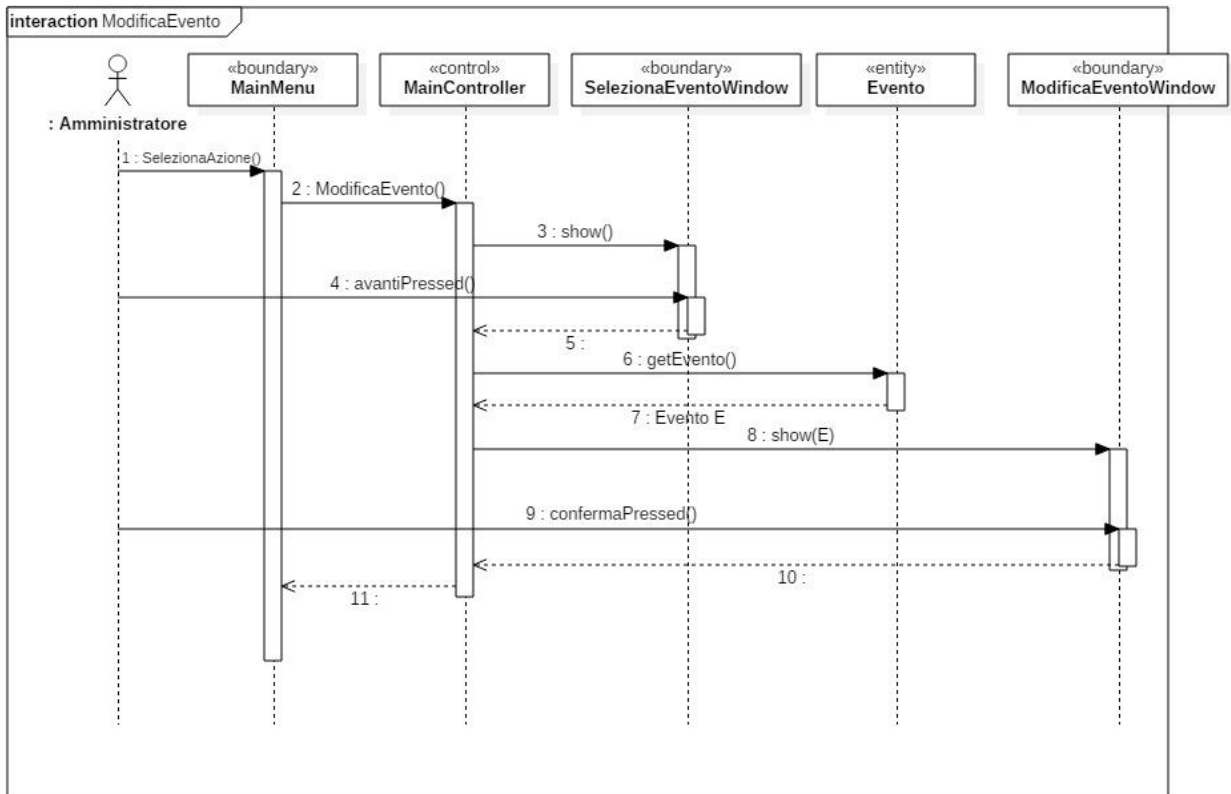


3.6 Sequence Diagrams

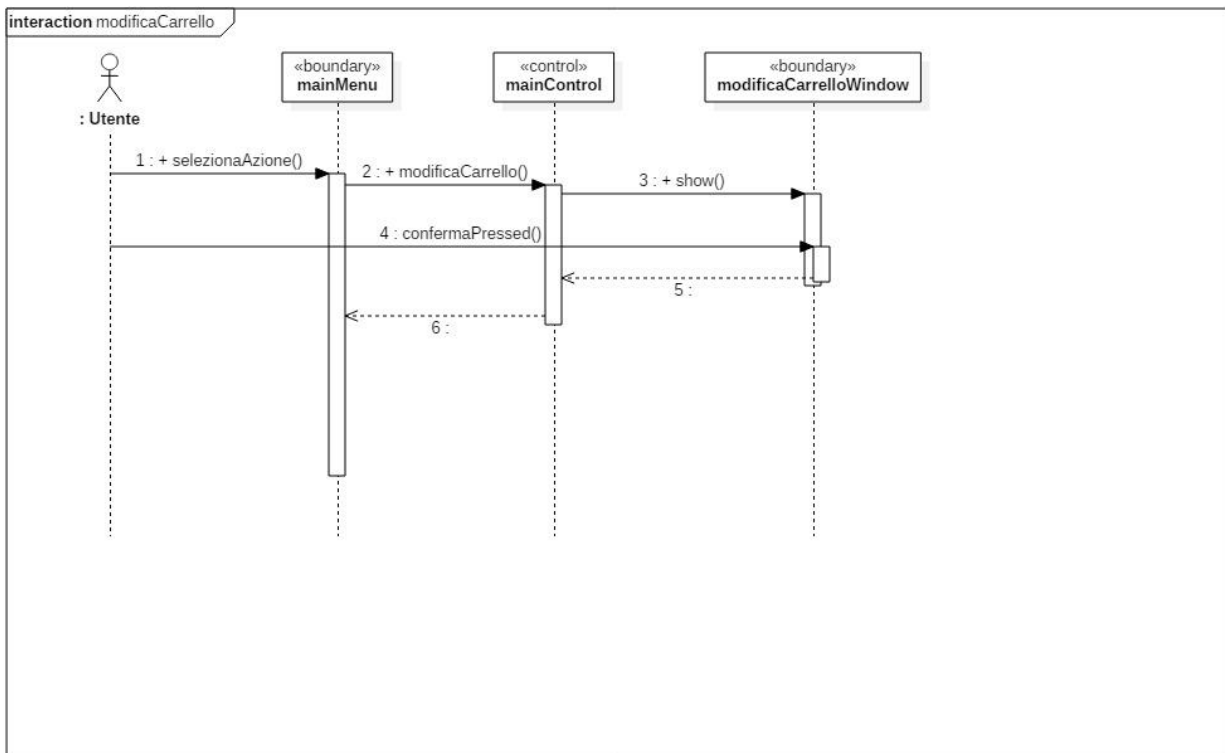
3.6.1 Inserisci Evento (Desktop Application)



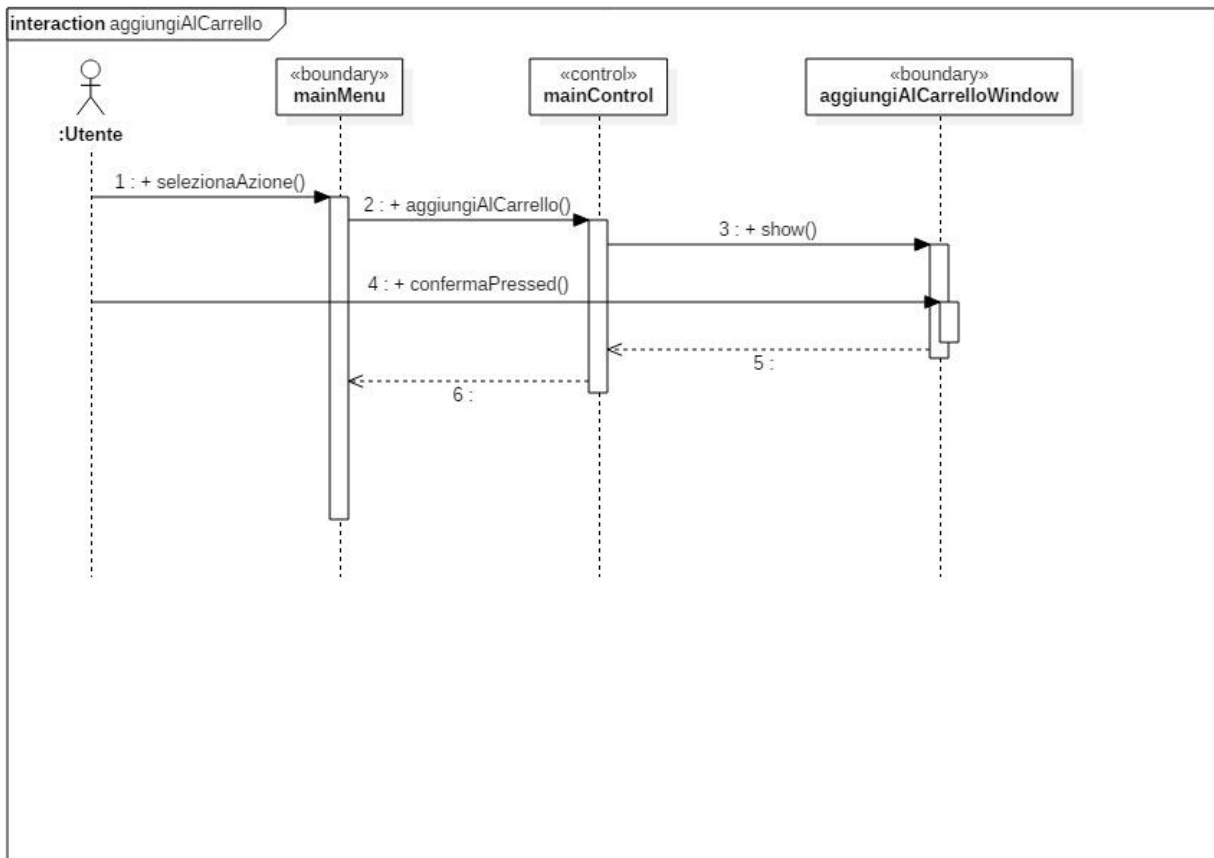
3.6.2 Modifica Evento (Desktop Application)



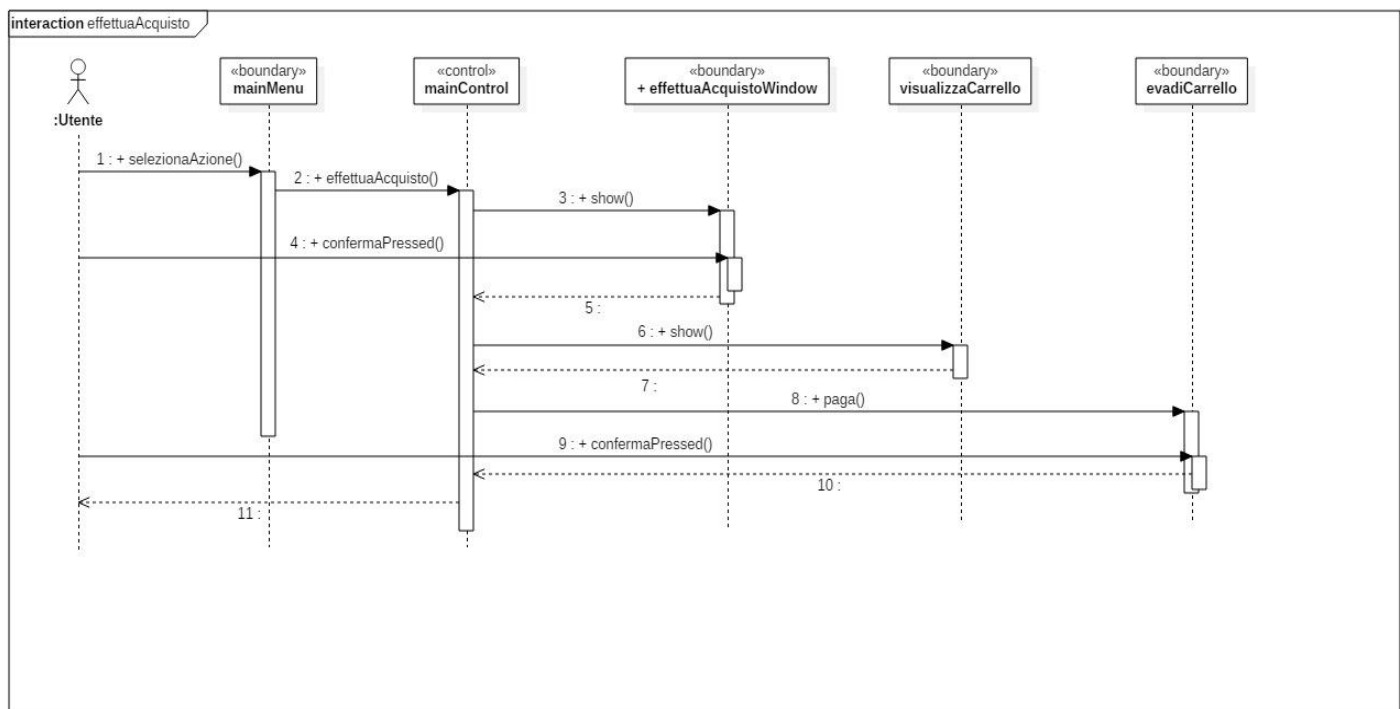
3.6.3 Modifica Carrello (Web Application)



3.6.4 Aggiungi Evento Carrello (Web Application)



3.6.5 Effettua Acquisto (Web Application)

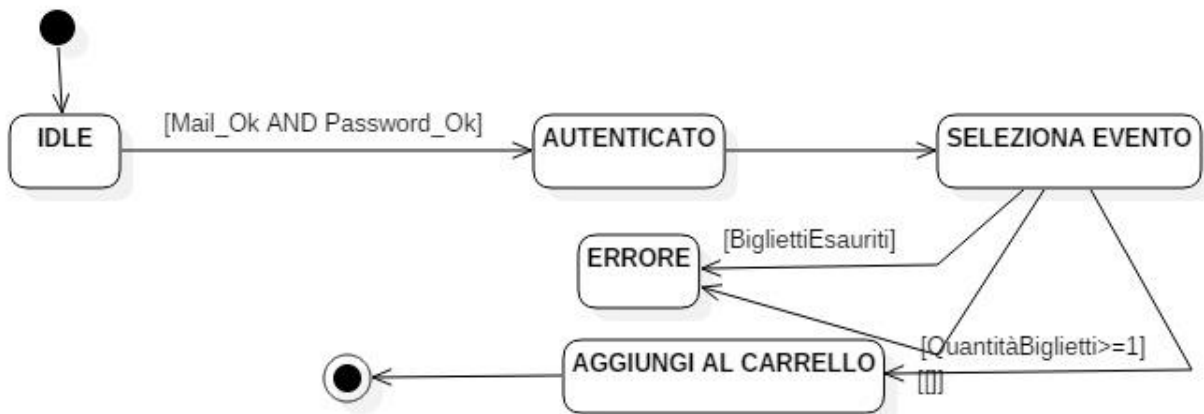


3.7 Statechart Diagrams

3.7.1 Inserisci Evento (Desktop Application)



3.7.2 Aggiungi Evento Carrello (Web Application)



3.8 Glossario

Nome	Descrizione
Amministratore	Utente della Desktop Application, il suo obiettivo è di gestire l'intero Sistema e di visualizzare le statistiche relative agli eventi.
Carrello	Oggetto che può contenere una lista di Eventi selezionati dall'utente, acquistabili in qualsivoglia momento.
Numero di ordine	Ogni ordine di pagamento emesso è etichettato da un indice, un numero di ordine, che è considerato un identificativo univoco.
QR-Code	Codice a barre bidimensionale, composto da moduli neri disposti all'interno di uno schema bianco di forma quadrata.
Utente Autenticato	Utente della Web Application che ha effettuato prima la registrazione e di seguito l'accesso.

4 Progettazione Software

Questa fase consiste in un insieme di attività che hanno il compito di trasformare il modello di analisi dei requisiti nel modello di design del sistema. A differenza della fase di analisi in cui ci si occupa del dominio di applicazione, nella fase di design ci si concentra sull'implementazione e quindi il livello di granularità è molto alto. Questo documento rappresenta una guida nell'implementazione del sistema descritto nel documento di analisi ed è quindi destinato a degli esperti di informatica che dovranno poi implementare il sistema. Di conseguenza, anche il linguaggio che sarà utilizzato d'ora in avanti sarà più tecnico rispetto a quello utilizzato nelle precedenti sezioni di questo documento.

4.1 System Design

4.1.1 Architettura Software

Per questo progetto , essendo abbastanza ampio ed abbracciando diversi tipi di tecnologie, abbiamo deciso di utilizzare più di un' architettura software, a seconda delle esigenze.

In particolare, il sottosistema relativo all'Applicazione Desktop di Back-End, implementa un'architettura del tipo "REPOSITORY".

Il sottosistema accede ad una singola struttura dati, chiamata *repository*, nel nostro caso una base di dati relazionale fornita da Amazon Cloud Services.

La scelta è ricaduta su questo tipo di architettura poiché permette in modo efficiente di condividere grandi moli di dati. Un sottosistema non si deve preoccupare di come i dati sono prodotti/usati da ogni altro eventuale sottosistema. Questa architettura garantisce, tra le altre cose, una gestione centralizzata di backup, security, access control e recovery da errori.

Inoltre, una volta pubblicato il modello di condivisione dati risulta più facile aggiungere nuovi sottosistemi e funzionalità, garantendo infine un buon riuso del codice.

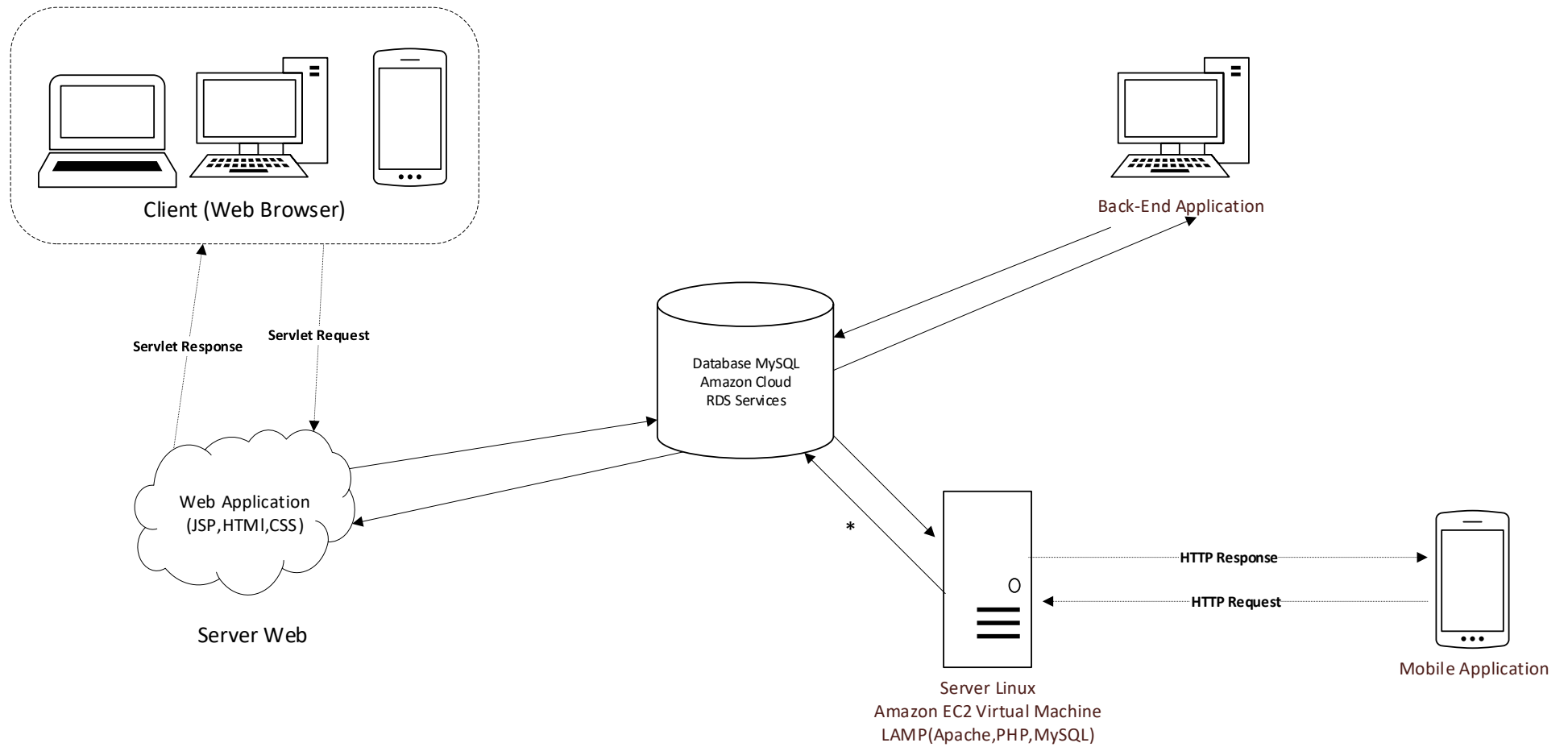
Per quanto riguarda le Applicazioni Web e Mobile , la scelta è ricaduta invece su un'architettura del tipo "CLIENT – SERVER" che possiamo definire quasi uno standard in situazioni come queste.

Nello specifico, nel caso dell' Applicazione Web, il ruolo del client è ricoperto da un Web Browser, mentre quello del server è ricoperto da un Web Server. Client e server scambiano informazioni attraverso delle servlet, mentre è compito del server gestire la connessione con la base di dati.

Infine, nel caso dell' Applicazione Mobile, il ruolo del client è ricoperto da un dispositivo mobile (es. Smartphone o Tablet Android), mentre quello del server è ricoperto da una Macchina Virtuale LAMP (Linux Kernel,Apache,MySQL,PHP) fornita sempre da Amazon Cloud Services. In questo caso, I due attori scambiano messaggi attraverso richieste HTTP, mentre è compito del server gestire la connessione con la base di dati, questa volta attraverso degli script realizzati in PHP.

Nel modello client-server c'è il vantaggio di avere una macchina centrale che è in grado di eseguire compiti complessi e di alleggerisce il carico sui client che possono non godere delle stesse capacità di elaborazione. Questo modello aiuta anche a rendere i dati più sicuri nelle occupazioni sensibili, dal momento che questi ultimi sono sempre fisicamente sul server, e sono semplicemente visualizzati sulle macchine client. Anche in questo caso, l'architettura garantisce un buon riuso del codice , la possibilità di cambiare facilmente base di dati in futuro e di creare nuove tipologie di client(es. app iOS) , modificando soltanto l'implementazione del server, senza modificare nessun client.

Uno degli obiettivi prefissati dal gruppo, per questo particolare progetto, è stato quello di volersi mettere alla prova con diverse tecnologie e diversi modi di gestirle, anche per questo motivo la nostra scelta è ricaduta su due diversi tipi di architettura.



(*) Il Server recupera informazioni dal Database attraverso script PHP.

Back-End Application :

Consiste nella prima delle tre applicazioni che interagiscono direttamente con gli utenti. Esso sarà utilizzato dagli amministratori per gestire l'intero Sistema, permettendo operazioni CRUD sulla base di dati. Tale applicazione, sarà compatibile con i principali sistemi operativi esistenti.

Mobile Application :

La mobile application è la seconda delle tre applicazioni che interagiscono direttamente con gli utenti finali. Essa sarà utilizzata da operatori sul campo per validare l'accesso agli eventi. Tale applicazione sarà compatibile con il Sistema operativo Android.

Web Application:

Ultima delle tre applicazioni ad interagire direttamente con l'utente. Dovrà permettere ad un utente, in questo caso un cliente, di potersi registrare sul Sistema, ricercare gli eventi disponibili, acquistare un certo numero di biglietti e poter accedere al proprio Qr-Code in qualsiasi momento.

Database :

DBMS relazionale MySQL fornito dal servizio Amazon Cloud, usato per la memorizzazione e condivisione dei dati permanenti.

Server Linux:

Macchina virtuale LAMP fornita dal servizio Amazon Cloud, usata come tramite tra l'App Mobile e il Database, per raccogliere e/o salvare informazioni.

Web Server:

Applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un web browser. Nel nostro caso, Apache Tomcat.

4.2 Tecnologie e Software Utilizzati

Tecnologia	Versione	Uso
Java	JDK 1.8	Linguaggio di programmazione usato per tutte le applicazioni.
Apache Tomcat	8.5	Web Server.
Java EE	Java EE 6	Servlet per server web services.

MySQL	Community Edition	Database.
Android	SDK 16 – API 4.1 JDK 1.7	Mobile application.
PHP	5.6.6	Linguaggio di scripting usato per connettere e gestire le interazioni tra Database e App Mobile.
Eclipse IDE	Oxygen 3.A	Ambiente di sviluppo per le applicazioni.
Android Studio	2.3.3	Ambiente di sviluppo per l'applicazione Android.
Bootstrap Studio	4.1.7	Ambiente di sviluppo per la creazione e manipolazione di pagine HTML,CSS e Javascript con tecnologia Bootstrap.
PutTy	0.70	Programma utilizzato per connettersi e gestire il server remoto.

4.2.1 Cloud Services

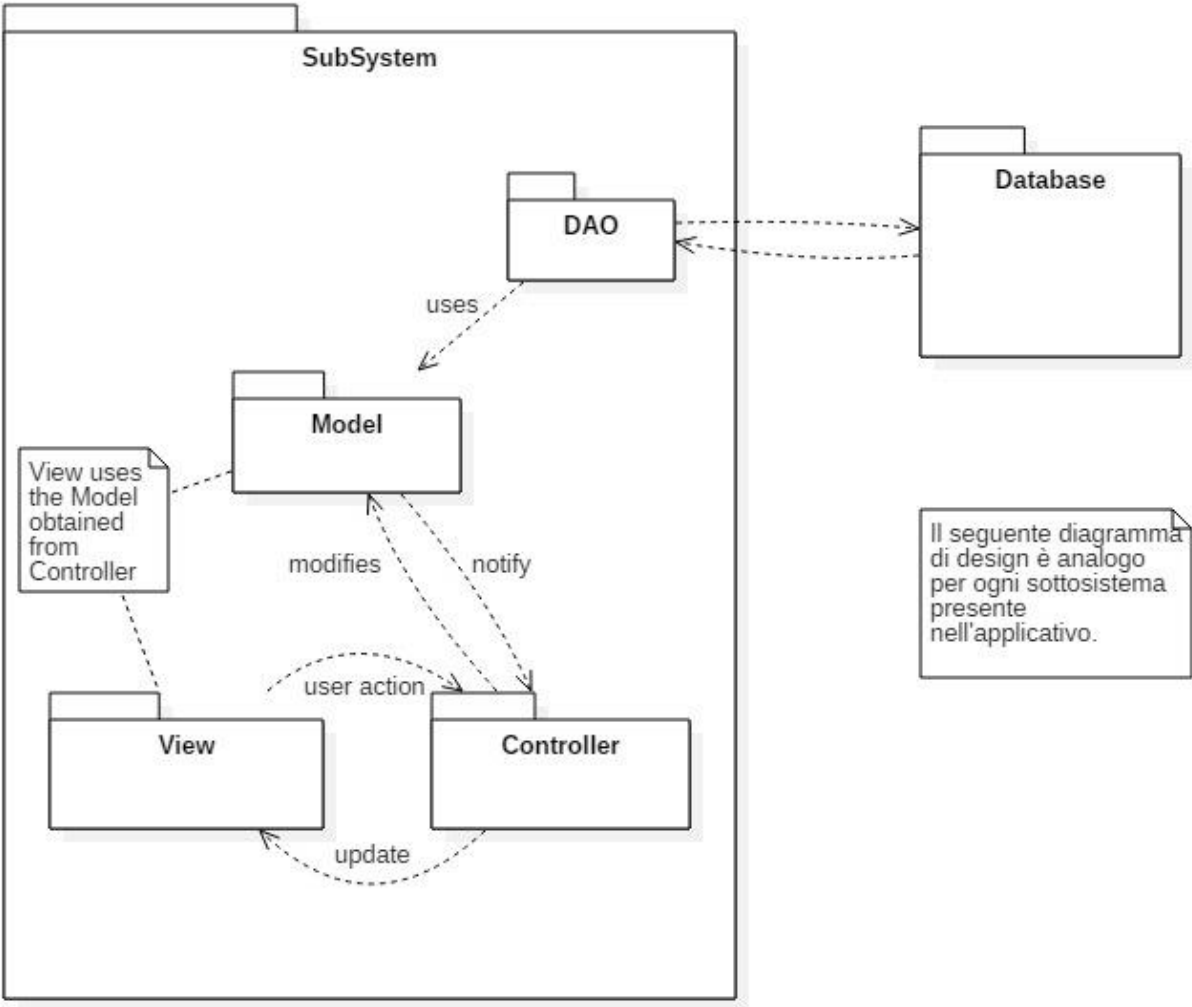
Tecnologia	Descrizione
Amazon Cloud Database RDS Service	Servizio offerto da Amazon Cloud Services per la creazione, gestione e condivisione di un Database MySQL.
Amazon Cloud EC2 Service Virtual Machine	Servizio offerto da Amazon Cloud Service per la creazione, gestione e condivisione di una macchina virtuale funzioni Lamp Server (Linux,Apache,MySQL,PHP).

4.2.2 Frameworks e Librerie

Nome	Uso	Referenze
iText	Creazione e manipolazione file PDF in Java e QR-Code	https://developers.itextpdf.com/apis
MySQL JDBC	Drivers per MySQL DBMS	https://docs.oracle.com/cd/E11882_01/java.112/e16548/toc.htm
jUnit 4.9	Unit testing	http://junit.org/junit4/
API PayPal	Gestione dei metodi di pagamento	https://developer.paypal.com/
Google Volley	Gestione richieste di connessioni App mobile.	https://developer.android.com/training/volley/

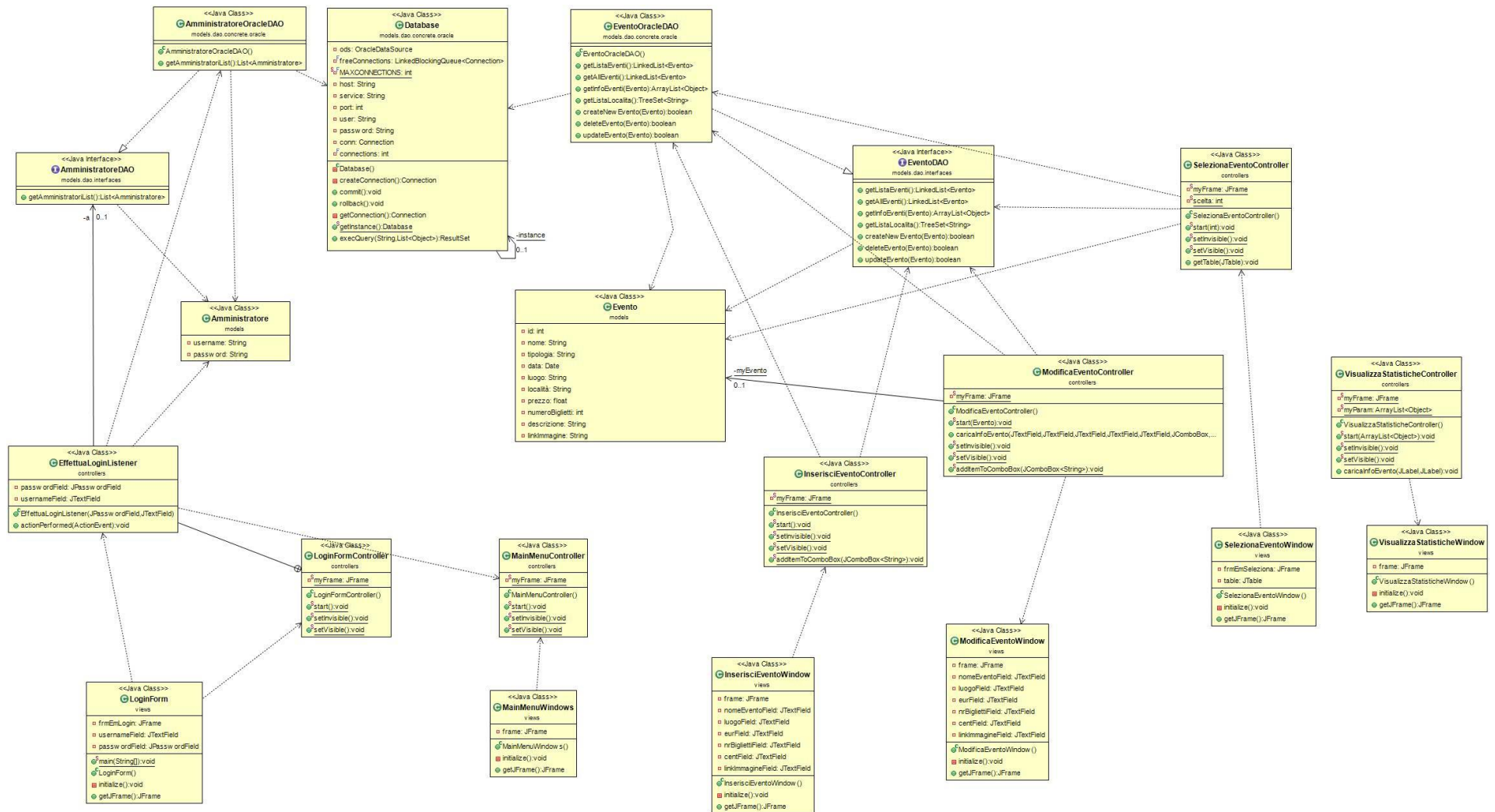
4.3 Object Design

4.3.1 System

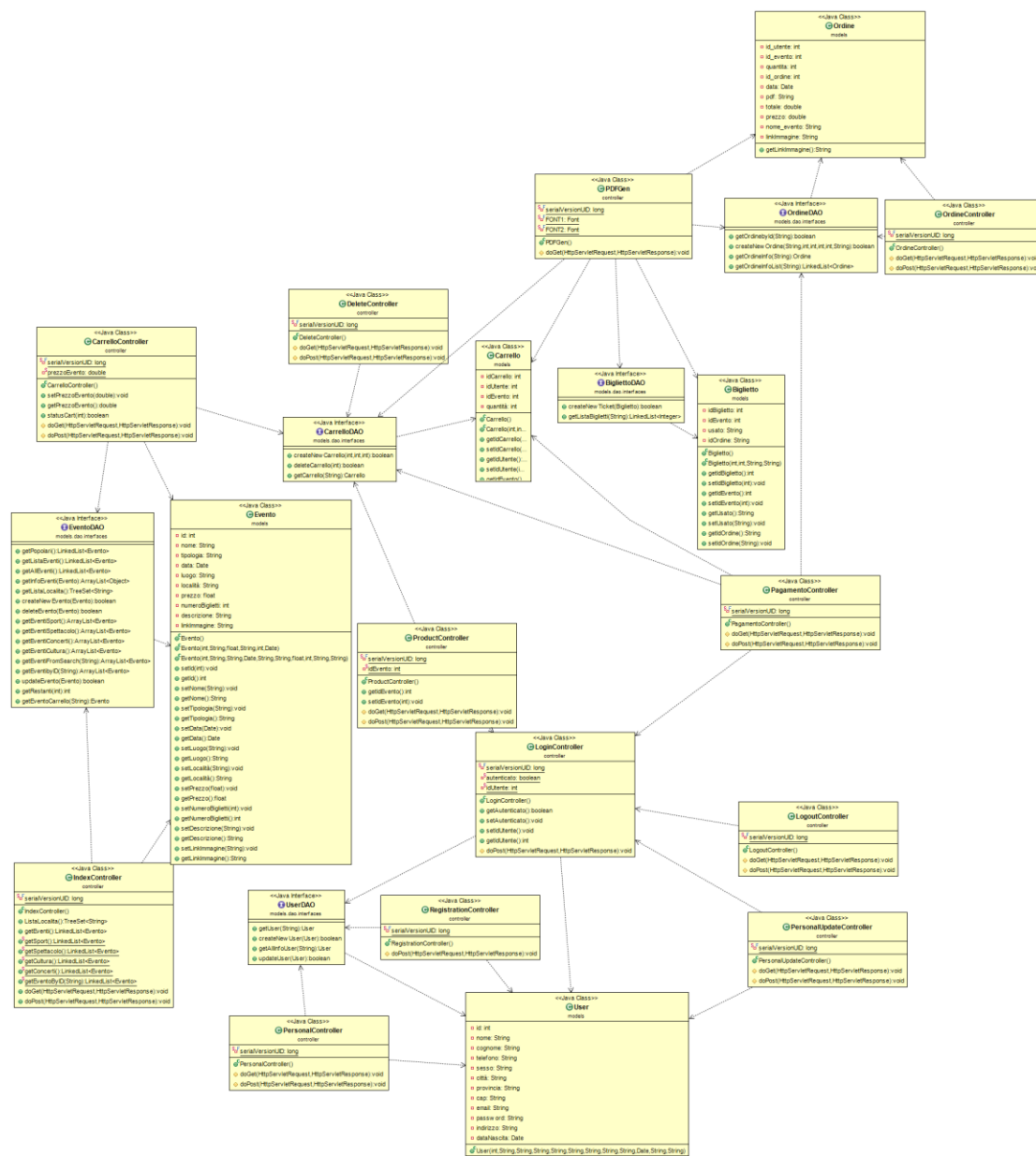


4.3.2 Class Diagrams

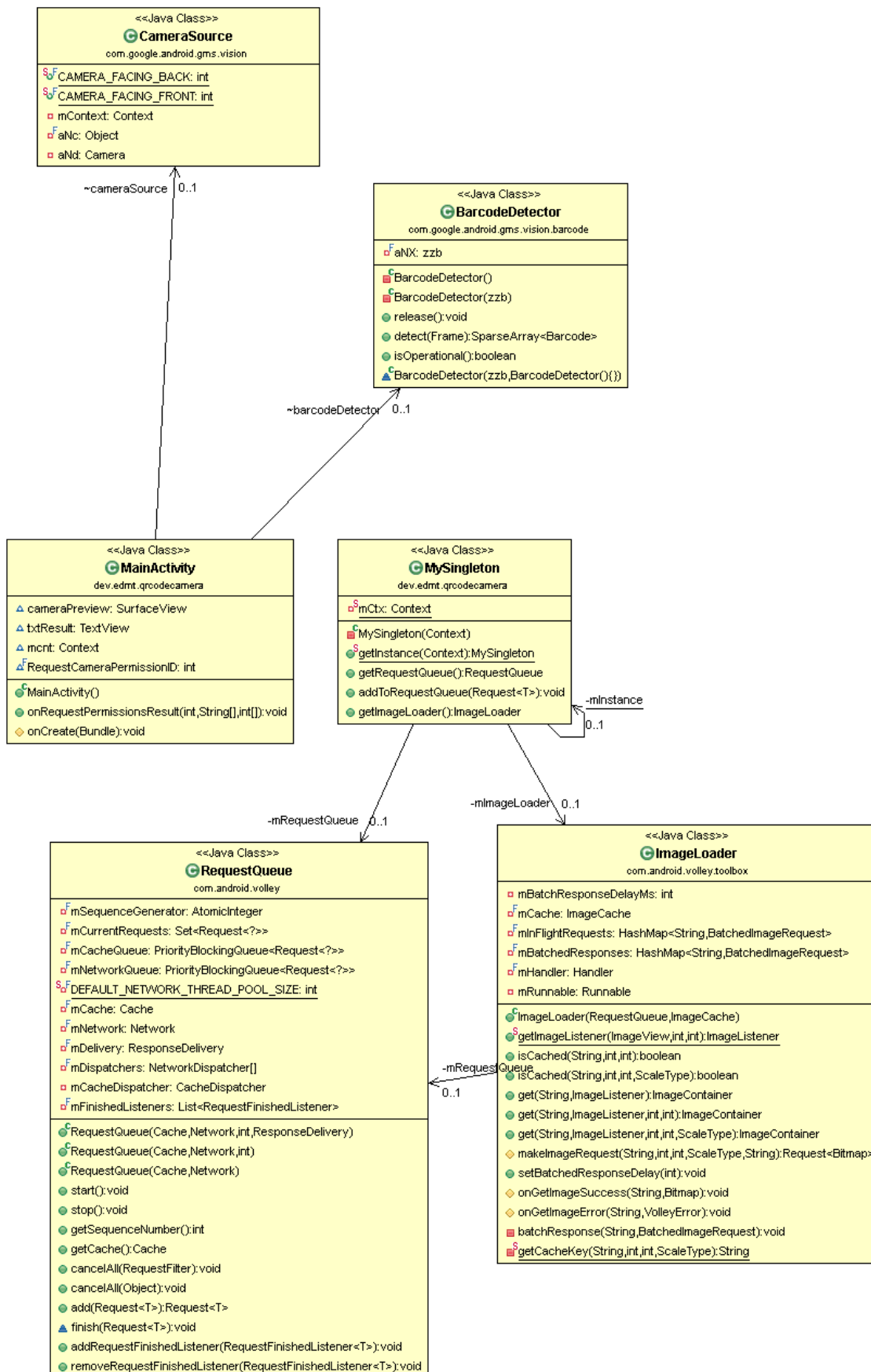
4.3.2.1 Desktop Application



4.3.2.2 Web Application

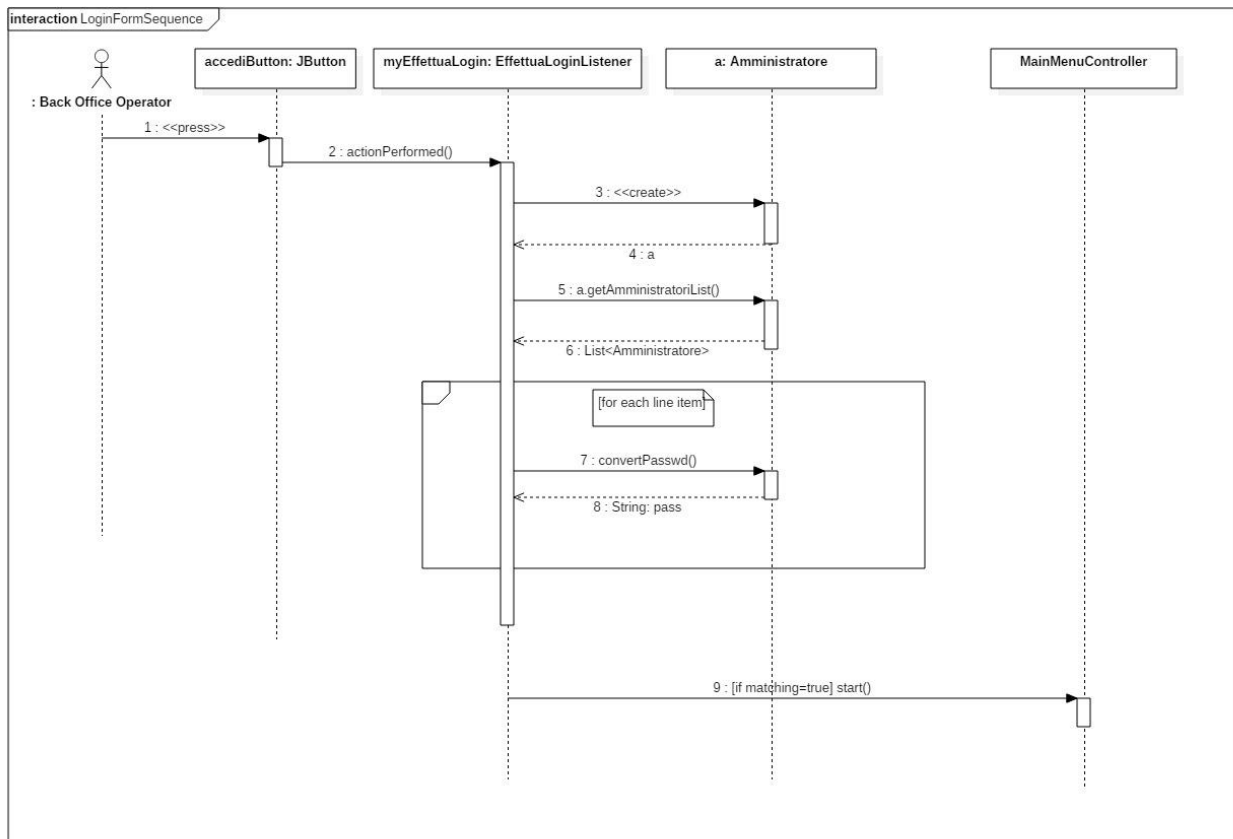


4.3.2.3 Mobile Application

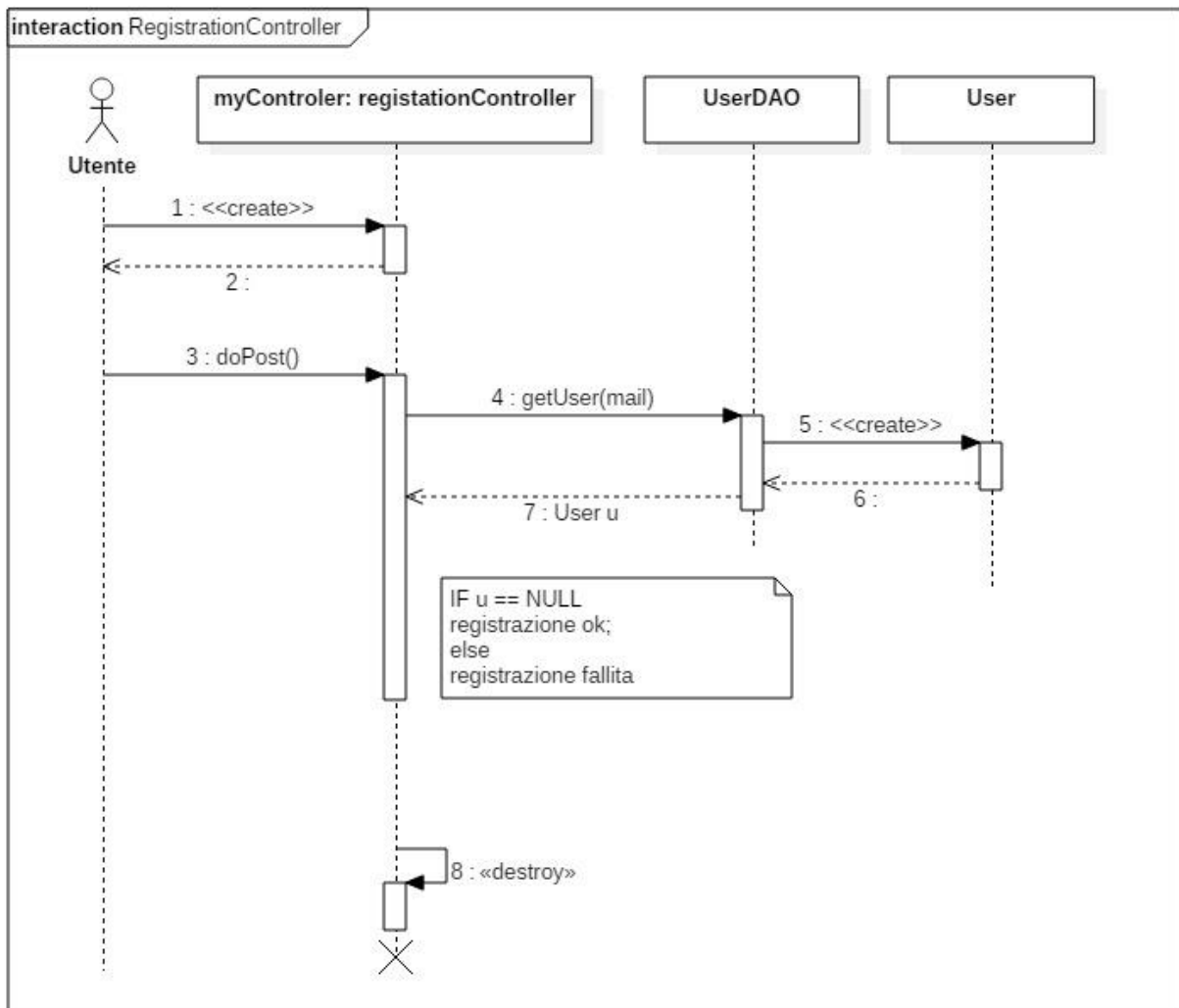


4.3.3 Sequence Diagrams

4.3.3.1 Desktop Application



4.3.3.2 Web Application



4.3.4 CRC Cards

4.3.4.1 Desktop Application

Class Name	LoginFormController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Gestire l'accesso al sistema da parte di un amministratore.		AmministratoreDAO , Amministratore, LoginFormWindow

Class Name	MainMenuController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Richiamare la funzionalità necessaria in base alla scelta effettuata dall'utente.		

Class Name	InserisciEventoController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Gestire il funzionamento di creazione di nuovi eventi all'interno del sistema.		EventoDAO, Evento , InserisciEventoWindow

Class Name	ModificaEventoController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Gestire il funzionamento di modifica di eventi all'interno del sistema.		EventoDAO,Evento,ModificaEventoWindow.

Class Name	SelezionaEventoController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Selezionare un evento da modificare.		EventoDAO,Evento,SelezionaEventoWindow
Selezionare un evento da eliminare.		EventoDAO,Evento,SelezionaEventoWindow

Class Name	SelezionaUtenteController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Gestione del meccanismo di selezione di un utente.		UtenteDAO,Utente,SelezionaUtenteWindow
Gestione del meccanismo di eliminazione di un utente.		UtenteDAO,Utente,SelezionaUtenteWindow

Class Name	VisualizzaStatisticheController	
Superclass		
Subclasses		
Responsabilities		Collaborators
Gestire il meccanismo di visualizzazione delle statistiche.		EventoDAO, Evento, VisualizzaStatisticheWindow

Interface Name	EventoDAO	
Implementation	EventoMySQLDAO	
Responsabilities		Collaborators
Funzionalità CRUD sulla tabella Evento, contenuta nella base di dati.		Evento

Interface Name	UserDAO	
Implementation	UserMySQLDAO	
Responsabilities		Collaborators
Funzionalità CRUD sulla tabella Utente, contenuta nella base di dati.		User

Interface Name	AmministratoreDAO	
Implementation	AmministratoreMySQLDAO	
Responsabilities		Collaborators
Funzionalità CRUD sulla tabella Amministratore, contenuta nella base di dati.		Amministratore

Class Name	LoginFormWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che permette all'utente di inserire username e password.		LoginFormController

Class Name	MainMenuWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che permette all'utente di scegliere l'operazione da compiere.		MainMenuController

Class Name	InserisciEventoWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che permette all'utente di inserire tutte le informazioni necessarie a creare un nuovo evento.		InserisciEventoController

Class Name	ModificaEventoWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che mostra tutte le informazioni relative ad un evento e permette all'utente di modificarle.		ModificaEventoWindow

Class Name	SelezionaEventoWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che permette all'utente di selezionare uno degli eventi disponibili.		SelezionaEventoController
Finestra che permette all'utente di selezionare uno degli eventi ed eliminarlo.		SelezionaEventoController

Class Name	SelezionaUtenteWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che permette all'amministratore di visualizzare i dati relativi agli utenti.		SelezionaUtenteController
Finestra che permette all'amministratore di eliminare l'utente selezionato		SelezionaUtenteController

Class Name	VisualizzaStatisticheWindow	
Superclass		
Subclasses		
Responsabilities		Collaborators
Finestra che permette di visualizzare le statistiche relative ad un evento.		VisualizzaStatisticheController

Class Name	Evento	
Superclass		
Subclasses		
Responsabilities		Collaborators
Memorizzare le informazioni relative ad un evento durante il ciclo di vita dell'applicazione.		

Class Name	User	
Superclass		
Subclasses		
Responsabilities		Collaborators
Memorizzare le informazioni relative ad un utente durante il ciclo di vita dell'applicazione.		

Class Name	Amministratore	
Superclass		
Subclasses		
Responsabilities		Collaborators
Memorizzare le informazioni relative ad un amministratore durante il ciclo di vita dell'applicazione.		

4.3.4.2 Web Application

Class Name	CarrelloController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di caricare le informazioni necessarie al carrello dell'utente autenticato.		EventoDAO, CarrelloDAO, Evento

Class Name	DeleteController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di eliminare un carrello esistente.		CarrelloDAO

Class Name	IndexController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di caricare tutte le informazioni necessarie per l'home page.		EventoDAO, Evento

Class Name	LoginController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di caricare tutte le informazioni necessarie all'autenticazione.		UserDAO, User

Class Name	LogoutController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet necessaria ad effettuare il logout dalla piattaforma		

Class Name	OrdineController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di generare un nuovo ordine a seguito di un pagamento.		OrdineDAO,Ordine

Class Name	PagamentoController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di creare un nuovo pagamento.		LoginController,Carrello,CarrelloDAO,OrdineDAO

Class Name	PDFGen	
Superclass		
Subclasses		
Responsibilities		Collaborators
Classe che si occupa di generare un nuovo file .pdf con tutte le informazioni relative all'acquisto.		Biglietto,Carrello,Ordine,BigliettoDAO,CarrelloDAO,OrdineDAO

Class Name	PersonalController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di caricare le info personali di un utente		UserDAO, User

Class Name	PersonalUpdateController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di caricare le info personali di un utente per poi modificarle		User

Class Name	ProductController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che si occupa di aggiungere un evento nel carrello di un utente		CarrelloDAO

Class Name	RegistrationController	
Superclass		
Subclasses		
Responsibilities		Collaborators
Servlet che gestisce il processo di registrazione		User, UserDAO

Interface Name	BigliettoDAO	
Implementation	BigliettoMySQLDAO	
	Responsabilities	Collaborators
Classe che si occupa di effettuare operazioni CRUD sui biglietti.		

Interface Name	CarrelloDAO	
Implementation	CarrelloMySQLDAO	
	Responsabilities	Collaborators
Classe che si occupa di effettuare operazioni CRUD sui carrelli.		

Interface Name	EventoDAO	
Implementation	EventoMySQLDAO	
	Responsabilities	Collaborators
Classe che si occupa di effettuare operazioni CRUD sugli eventi.		

Interface Name	OrdineDAO	
Implementation	OrdineMySQLDAO	
	Responsabilities	Collaborators
Classe che si occupa di effettuare operazioni CRUD sugli ordini.		

Interface Name	UserDAO	
Implementation	UserMySQLDAO	
	Responsabilities	Collaborators
Classe che si occupa di effettuare operazioni CRUD sugli utenti.		

Class Name	User	
Superclass		
Subclasses		
Responsibilities		Collaborators
Classe che si occupa di salvare i dati relativi agli utente durante il ciclo di vita dell'applicazione		

Class Name	Ordine	
Superclass		
Subclasses		
Responsibilities		Collaborators
Classe che si occupa di salvare i dati relativi agli ordini durante il ciclo di vita dell'applicazione		

Class Name	Evento	
Superclass		
Subclasses		
Responsibilities		Collaborators
Classe che si occupa di salvare i dati relativi agli eventi durante il ciclo di vita dell'applicazione		

Class Name	Carrello	
Superclass		
Subclasses		
Responsibilities		Collaborators
Classe che si occupa di salvare i dati relativi ai carrelli durante il ciclo di vita dell'applicazione		

Class Name	Biglietto	
Superclass		
Subclasses		
Responsibilities		Collaborators
Classe che si occupa di salvare i dati relativi ai biglietti durante il ciclo di vita dell'applicazione		

4.3.4.3 Mobile Application

Class Name	MainActivity	
Superclass		
Subclasses		
Responsabilities		Collaborators
Classe che si occupa di scannerizzare i QR-Code e di inoltrare una nuova richiesta di connessione.		MySingleton

Class Name	MySingleton	
Superclass		
Subclasses		
Responsabilities		Collaborators
Classe che si occupa di gestire le richieste di connessione.		

5 Testing

5.1 System Testing

Come concordato con il cliente, abbiamo creato un piano di testing del sistema che verifica ogni comportamento dell'interfaccia utente e ogni caso d'uso definito nella sezione "Requisiti funzionali".

5.1.1 Web Application

Test ID	1		
Test Name	Eventi disponibili		
Test Description	Test della ricerca degli eventi disponibili.		
Input	System status	Oracle	Output
L'Utente accede alla Home Page.	Il sistema funziona correttamente.	L'applicazione mostra in Home Page gli eventi disponibili più popolari.	
L'Utente inserisce dei valori nella barra di ricerca e preme il tasto "Cerca"	Il sistema correttamente.	L'applicazione mostra una pagina con gli eventi relativi alla chiave di ricerca utilizzata.	
L'Utente lascia vuota la barra di ricerca e preme il tasto "Cerca"	Il sistema funziona correttamente.	L'applicazione mostra una pagina con tutti gli eventi disponibili.	
L'Utente inserisce dei valori nella barra di ricerca (oppure la lascia vuota) e preme il tasto "Cerca"	Il sistema non funziona correttamente.	L'applicazione mostra una pagina di errore.	
L'Utente accede alla Home Page	Il sistema non funziona correttamente.	L'applicazione mostra una pagina di errore.	

Test ID	2		
Test Name	Web App Login		
Test Description	Test sul funzionamento del meccanismo di login		
Input	System status	Oracle	Output
L'Utente inserisce la giusta combinazione di username e password.	Il sistema funziona correttamente.	Il sistema logga l'Utente e mostra la Home Page.	
L'Utente inserisce una combinazione sbagliata di username e password.	Il sistema funziona correttamente.	Il sistema mostra un messaggio d'errore "Utente non esistente" oppure "Password errata"	
L'Utente lascia uno dei due campi (oppure entrambi) vuoti.	Il sistema funziona correttamente	Il sistema mostra un messaggio d'errore "Utente non esistente" oppure "Password errata"	

L'Utente non lascia campi vuoti (la combinazione può essere giusta oppure no)	Il sistema non funziona correttamente.	Il sistema mostra una pagina di errore.	
-------------------------------------------------------------------------------	----------------------------------------	-----------------------------------------	--

Test ID	3		
Test Name	Web App Registrazione		
Test Description	Test sul funzionamento del meccanismo di registrazione		
Input	System status	Oracle	Output
L'Utente inserisce tutti i dati in maniera corretta.	Il sistema funziona correttamente.	Il sistema mostra una pagina di conferma registrazione.	
L'Utente inserisce almeno un campo in maniera non corretta.	Il sistema funziona correttamente.	Il sistema colora di rosso il box relativo al campo/i sbagliato.	
L'Utente inserisce un indirizzo e-mail già utilizzato.	Il sistema funziona correttamente	Il sistema mostra una pagina di registrazione fallita.	
L'Utente compila tutti i campi	Il sistema non funziona correttamente.	Il sistema mostra una pagina di errore.	

Test ID	4		
Test Name	Web App Aggiungi al carrello		
Test Description	Test sull'aggiunta di un evento al carrello.		
Input	System status	Oracle	Output
L'Utente Autenticato (può modificare il nr di biglietti desiderati e)clicca su "Aggiungi al carrello"	Il sistema funziona correttamente.	Il sistema aggiunge l'evento al Carrello e torna all'Home Page.	
L'Utente Autenticato clicca su "Aggiungi al carrello"	Il sistema non funziona correttamente.	Il sistema mostra una pagina di errore.	

Test ID	5		
Test Name	Web App Rimuovi dal carrello		
Test Description	Test sulla rimozione di un evento dal carrello.		
Input	System status	Oracle	Output
L'Utente Autenticato clicca su "Rimuovi" di fianco al nome dell'evento.	Il sistema funziona correttamente.	Il sistema rimuove l'evento dal carrello e torna alla Home Page.	
L'Utente Autenticato clicca su "Rimuovi" di fianco al nome dell'evento	Il sistema non funziona correttamente.	Il sistema mostra una pagina d'errore.	

Test ID	6		
Test Name	Web App Pagamento		
Test Description	Test sul funzionamento del meccanismo di pagamento		
Input	System status	Oracle	Output
L'Utente Autenticato avvia la procedura di pagamento.	Il sistema funziona correttamente.	Il sistema mostra un pop-up che permetta di completare il pagamento.	
L'Utente annulla la procedura di pagamento.	Il sistema funziona correttamente.	Il sistema torna alla pagina Carrello.	
L'Utente porta a termine correttamente la procedura di pagamento.	Il sistema funziona correttamente.	Il sistema mostra una pagina di riepilogo ordine, generando un PDF e i Qr-Code relativi ai biglietti acquistati.	
La procedura di pagamento fallisce.	Il sistema funziona correttamente.	Il sistema mostra una pagina di pagamento fallito.	
L'Utente porta a termine la procedura di pagamento(correttamente o no).	Il sistema non funziona .	Il sistema mostra una pagina di errore.	

Test ID	7		
Test Name	Web App Dati Personali		
Test Description	Test sul funzionamento della visualizzazione/modifica dati personali.		
Input	System status	Oracle	Output
L'Utente Autenticato visualizza il suo profilo.	Il sistema funziona correttamente.	Il sistema mostra i dati relativi all'utente attualmente loggato.	
L'Utente Autenticato visualizza il suo profilo.	Il sistema non funziona.	Il sistema mostra una pagina di errore.	
L'Utente Autenticato clicca su "Modifica i miei dati personali"	Il sistema funziona correttamente.	Il sistema abilita la modifica dei dati personali.	
L'Utente Autenticato modifica in modo errato i dati.	Il sistema funziona correttamente.	Il sistema colora di rosso il box relativo al campo o ai campi in questione.	
L'Utente Autenticato modifica in modo corretto i dati	Il sistema funziona correttamente.	Il sistema aggiorna e mostra i dati aggiornati.	
L'Utente Autenticato modifica i dati	Il sistema non funziona.	Il sistema mostra una pagina di errore.	

Test ID	8		
Test Name	Web App Ordini Passati		
Test Description	Test sul funzionamento della pagina "I miei ordini"		
Input	System status	Oracle	Output
L'Utente Autenticato visualizza i suoi ordini.	Il sistema funziona correttamente.	Il sistema mostra tutti gli ordini relativi all'utente attualmente loggato.	
L'Utente Autenticato visualizza i suoi ordini.	Il sistema non funziona.	Il sistema mostra una pagina di errore.	
L'Utente Autenticato scarica un PDF	Il sistema funziona correttamente.	Il sistema mostra il relativo PDF.	
L'Utente Autenticato scarica un PDF.	Il sistema non funziona.	Il sistema mostra una pagina di errore.	

5.1.2 Mobile Application

Test ID	9		
Test Name	Mobile App		
Test Description	Test sul funzionamento della scannerizzazione dei QR-Code.		
Input	System status	Oracle	Output
L'Addetto scannerizza un QR-Code di terze parti.	Il sistema funziona correttamente.	Il sistema mostra un messaggio d'errore e ritorna alla fotocamera.	
L'Addetto scannerizza un QR-Code già utilizzato	Il sistema funziona correttamente.	Il sistema mostra un messaggio d'errore e ritorna alla fotocamera.	
L'Addetto scannerizza un QR-Code non ancora utilizzato	Il sistema funziona correttamente.	Il sistema mostra una finestra di dialogo con i dati relativi all'evento, nella quale è possibile confermare l'accesso oppure annullare l'operazione.	
L'Addetto conferma la validazione del biglietto.	Il sistema funziona correttamente.	Il sistema imposta lo stato del biglietto come usato.	
L'Addetto scannerizza un QR-Code	Il sistema non funziona.	Il sistema mostra un messaggio d'errore.	

5.1.3 Desktop Application

Test ID	10		
Test Name	Desktop App Login		
Test Description	Test sul funzionamento del meccanismo di login		
Input	System status	Oracle	Output
L'Amministratore inserisce la giusta combinazione di username e password.	Il sistema funziona correttamente.	Il sistema logga l'Utente e mostra la pagina principale.	
L'Amministratore inserisce una combinazione sbagliata di username e password.	Il sistema funziona correttamente.	Il sistema mostra un pop-up di errore.	
L'Amministratore lascia uno dei due campi (oppure entrambi) vuoti.	Il sistema funziona correttamente	Il sistema non abilita il tasto "Accedi".	
L'Amministratore inserisce una combinazione di username e password.	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore chiude l'applicazione	Il sistema funziona correttamente.	Il sistema chiude la finestra e l'applicazione.	

Test ID	11		
Test Name	Desktop App Inserimento		
Test Description	Test sul funzionamento del meccanismo di inserimento di un evento		
Input	System status	Oracle	Output
L'Amministratore compila tutti i dati del form correttamente.	Il sistema funziona correttamente.	Il sistema mostra un pop-up di riepilogo, con la possibilità di confermare o annullare l'operazione.	
L'Amministratore compila almeno un dato del form in modo errato.	Il sistema funziona correttamente.	Il sistema mostra un pop-up di errore.	
L'Amministratore conferma un inserimento.	Il sistema funziona correttamente	Il sistema mostra un messaggio di conferma, chiude la finestra e torna al menu principale.	
L'Amministratore prova ad inserire un evento	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore chiude la finestra	Il sistema funziona correttamente	Il sistema chiude la finestra e torna al menu principale.	

Test ID	12		
Test Name	Desktop App Modifica		
Test Description	Test sul funzionamento del meccanismo di modifica di un evento		
Input	System status	Oracle	Output
L'Amministratore seleziona un evento modificabile	Il sistema funziona correttamente.	Il sistema apre l'evento selezionato e apre una finestra dal quale sarà possibile modificarlo.	
L'Amministratore seleziona un evento.	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore compila tutti i dati del form correttamente.	Il sistema funziona correttamente.	Il sistema mostra un pop-up di riepilogo, con la possibilità di confermare o annullare l'operazione.	
L'Amministratore compila almeno un dato del form in modo errato.	Il sistema funziona correttamente.	Il sistema mostra un pop-up di errore.	
L'Amministratore conferma una modifica.	Il sistema funziona correttamente	Il sistema mostra un messaggio di conferma, chiude la finestra e torna al menu principale.	
L'Amministratore prova modificare un evento	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore chiude la finestra di modifica	Il sistema funziona correttamente	Il sistema chiude la finestra e torna al menu per selezionare un evento da modificare.	

Test ID	13		
Test Name	Desktop App Elimina Evento		
Test Description	Test sul funzionamento del meccanismo di cancellazione di un evento		
Input	System status	Oracle	Output
L'Amministratore seleziona l'evento da eliminare e clicca su "Conferma"	Il sistema funziona correttamente.	Il sistema elimina l'evento e mostra un messaggio di conferma.	
L'Amministratore prova ad eliminare un evento.	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore chiude la finestra di eliminazione	Il sistema funziona correttamente	Il sistema chiude la finestra e torna al menu principale.	

Test ID	14		
Test Name	Desktop App Elimina Utente		
Test Description	Test sul funzionamento del meccanismo di cancellazione di un utente		
Input	System status	Oracle	Output
L'Amministratore seleziona l'utente da eliminare e clicca su "Conferma"	Il sistema funziona correttamente.	Il sistema elimina l'utente e mostra un messaggio di conferma.	
L'Amministratore prova ad eliminare un utente.	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore chiude la finestra di eliminazione	Il sistema funziona correttamente	Il sistema chiude la finestra e torna al menu principale.	

Test ID	15		
Test Name	Desktop App Visualizza Statistiche		
Test Description	Test sul funzionamento del meccanismo di visualizzazione statistiche.		
Input	System status	Oracle	Output
L'Amministratore seleziona l'evento che vuole visualizzare.	Il sistema funziona correttamente.	Il sistema mostra una pagina relativa alle statistiche dell'evento scelto.	
L'Amministratore prova visualizzare le statistiche	Il sistema non funziona.	Il sistema mostra un pop-up di errore.	
L'Amministratore chiude la finestra di visualizzazione statistiche	Il sistema funziona correttamente	Il sistema chiude la finestra e torna al menu principale.	

5.2 Unit Testing

Abbiamo deciso di testare due funzioni con JUnit, con i seguenti casi di test, utilizzando il metodo **SECT** (Strong Equivalence Class Testing). In particolare abbiamo testato il metodo *convertPasswd* (usato sia nella web app che nella desktop app) che si occupa di convertire una stringa (utilizzata come password) nel formato MD5 per essere salvata nella nostra base di dati in maniera crittografata. Il secondo metodo testato è *getOrdinebyId* (usato nella Web App) che si occupa di ritornare true se esiste un ordine identificato dalla stringa passata in ingresso, false altrimenti.

5.2.1 Primo metodo: convertPasswd

5.2.1.1 Classi di equivalenza

Parameters domain	Description	Equivalence classes	Values
A	Solo caratteri alfabetici	A1	admin
	Solo caratteri numerici	A2	230495
	Alfabetici+Numerici	A3	adm1n23

5.2.1.2 Test cases

Test case	A	Output
1	A1	21232f297a57a5a743894a0e4a801fc3
2	A2	8b8d91f2dc4e7d972604ca0bfd2a132c
3	A3	5022272afb8d65c0e37c34630ad77968

5.2.1.3 Codice

```
package models;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class AmministratoreTest {
    private Amministratore a;

    @BeforeEach
    void setUp() throws Exception {
        a=new Amministratore();
        assertNotNull(a);
    }

    @AfterEach
    void tearDown() throws Exception {
        a=null;
        assertNull(a);
    }

    @Test
    //Test password solo caratteri alfabetici
    void test1() {
        a.setUsername("admin");
```

```

        a.setPassword("admin");
        String pass = Amministratore.convertPasswd(a.getPassword());
        String oracolo = "21232f297a57a5a743894a0e4a801fc3";
        assertEquals(oracolo, pass);
    }

```

```

@Test
//Test password solo caratteri numerici
void test2() {
    a.setUsername("admin");
    a.setPassword("230495");
    String pass = Amministratore.convertPasswd(a.getPassword());
    String oracolo = "8b8d91f2dc4e7d972604ca0bfd2a132c";
    assertEquals(oracolo, pass);
}

```

```

@Test //Test password numeri e alfabeto
void test3() {
    a.setUsername("admin");
    a.setPassword("adm1n23");
    String pass = Amministratore.convertPasswd(a.getPassword());
    String oracolo = "5022272afb8d65c0e37c34630ad77968";
    assertEquals(oracolo, pass);
}

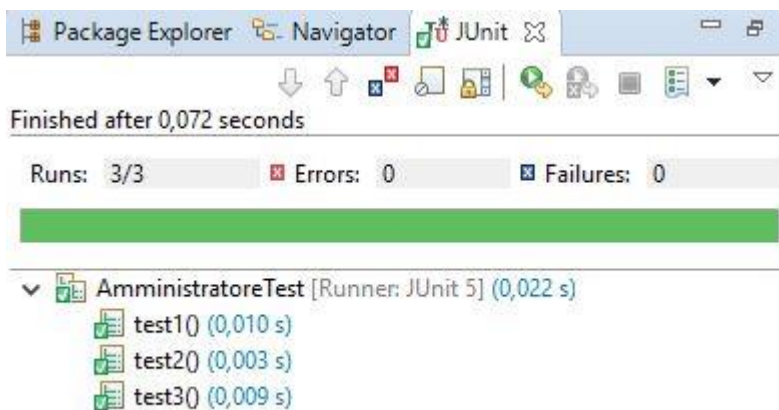
```

```

}

```

5.2.1.4 JUnit test output



5.2.2 Secondo metodo: getOrdinebyId

5.2.2.1 Classi di equivalenza

Parameters domain	Description	Equivalence classes	Values
A	Parametro numerico, presente nella base di dati.	A1	"9"
	Parametro numerico, non presente nella base di dati	A2	"135"
	Parametro stringa	A3	"nove"
	Numero negativo	A4	"-9"

Test case	A	Output
1	A1	TRUE
2	A2	FALSE
3	A3	FALSE
4	A4	FALSE

5.2.2.2 Codice

```
package models.dao.concrete.MySQL;

import static org.junit.jupiter.api.Assertions.*;

import java.text.ParseException;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class OrdineMySQLDAOTest {

    OrdineMySQLDAO prova;

    @BeforeEach
    void setUp() throws Exception {
        prova = new OrdineMySQLDAO();
        assertNotNull(prova);
    }

    @AfterEach
    void tearDown() throws Exception {
        prova=null;
        assertNull(prova);
    }

    @Test //Id numerico positivo conenuto nel db.
    void test1() throws ParseException {
        boolean myTest= prova.getOrdinebyId("9");
        boolean oracolo = true;
        assertEquals(oracolo,myTest);
    }
}
```

```

@Test //Id numerico positivo non contenuto nel db.
void test2() throws ParseException {
    boolean myTest= prova.getOrdinebyId("135");
    boolean oracolo = false;
    assertEquals(oracolo,myTest);
}

```

```

@Test //Id numerico positivo conenuto nel db.
void test3() throws ParseException {
    boolean myTest= prova.getOrdinebyId("nove");
    boolean oracolo = false;
    assertEquals(oracolo,myTest);
}

```

```

@Test //Id numerico negativo.
void test4() throws ParseException {
    boolean myTest= prova.getOrdinebyId("-9");
    boolean oracolo = false;
    assertEquals(oracolo,myTest);
}

```

```

}

```

5.2.2.3 JUnit test output

