2024/2025

MYEVENT

Documentazione Tecnica

Arianna D’Aniello ISPW

Sommario

1 REQUISITI SOFTWARE

1.1 INTRODUZIONE

1.1.1 Scopo del documento

Il presente documento ha l'obiettivo di delineare in modo chiaro e strutturato i requisiti fondamentali e specifici del sistema Myevent.
In apertura, vengono descritte le tre user stories relative alla piattaforma, seguite dall'analisi dei requisiti funzionali. Successivamente, l'attenzione si sposta sulla modellazione del sistema attraverso i diagrammi riportati qui in basso, al fine di fornire una rappresentazione dettagliata degli aspetti interni ed esterni più rilevanti

- Class Diagram
- Activity Diagram
- Sequence Diagram
- State Diagram

1.1.2 Overview di sistema

Il sistema è stato progettato per offrire una piattaforma moderna e interattiva che consenta a utenti, organizzatori e amministratori di gestire in modo semplice ed efficace gli eventi e le relative prenotazioni.
L'obiettivo principale è quello di semplificare la ricerca, la prenotazione e la gestione dei biglietti, garantendo un servizio intuitivo e sempre aggiornato.

Gli utenti registrati avranno la possibilità di esplorare un ampio catalogo di eventi, filtrandoli in base a data, luogo, categoria e prezzo. Potranno prenotare i biglietti e, se desiderano, ricevere notifiche sugli eventi che corrispondono ai loro interessi.

Gli organizzatori registrati potranno creare e modificare eventi, gestire le prenotazioni ricevute e consultare report statistici per monitorare l'andamento delle vendite.

Infine, gli amministratori avranno accesso a funzionalità avanzate per supervisionare l'intera piattaforma e generare report, fornendo così un supporto costante alla gestione complessiva del sistema.

1.1.3 Requisiti Hardware e Software

Per garantire un corretto funzionamento dell'applicazione, il sistema deve soddisfare i seguenti requisiti minimi:

Requisiti Hardware: un PC con i suoi componenti necessari e spazio libero sul disco per l'installazione e la gestione del database

Requisiti Software: si richiede l'ambiente di sviluppo Java, nello specifico l'installazione Oracle JDK 21 (o OpenJDK 21 con supporto JavaFX) ed il Driver JDBC MySQL presente nel classpath dell'applicazione. Inoltre, è richiesto un server che ospiti il DBMS, in modo che le informazioni possano essere ottenute tramite query SQL al server .

1.1.4 Competitor, Pro e Contro

In questo paragrafo verranno analizzati alcuni servizi concorrenti alla piattaforma Quizlytic, con l'obiettivo di evidenziarne le principali caratteristiche, i punti di forza e le limitazioni.

Eventzilla

Pro:

- Economica e semplice da configurare, con moduli per sito evento, pagamento tramite Stripe/PayPal, app mobile per check-in e reportistica base
- Ideale per eventi di PMI o organizzazioni no profit.

Contro:

- Personalizzazione e integrazioni limitate
- Interfaccia meno raffinata e supporto spesso lento.

Whova

Pro:

- Supporta l'intero ciclo dell'evento, dalla registrazione al check-in, all'app mobile e gestione sponsor ed espositori.
- Ottimi strumenti di coinvolgimento dei partecipanti come ritrovo, networking, gestione badge.

Contro:

- Abbastanza costoso, soprattutto per eventi su larga scala.
- Può risultare eccessivo per eventi semplici o piccoli.

1.2 USER STORIES

1.2.1 US-1

Come cliente, **voglio** visionare il catalogo degli eventi, **in modo da** valutare se effettuare una prenotazione.

1.2.2 US-2

Come cliente, **voglio** prenotare un biglietto, **in modo da** assicurarmi l'accesso nella data stabilita.

1.2.3 US-3

Come organizzatore, **voglio** aggiungere i miei eventi nel catalogo, **in modo da** poter essere visualizzati dai clienti e prenotati.

1.3 FUNCTIONAL REQUIREMENTS

1.3.1 FR-1

Il sistema deve offrire la ricerca di eventi basata su *filtri*.

1.3.2 FR-2

Il sistema deve *verificare* che l'utente possa avanzare una richiesta di prenotazione.

1.3.3 FR-3

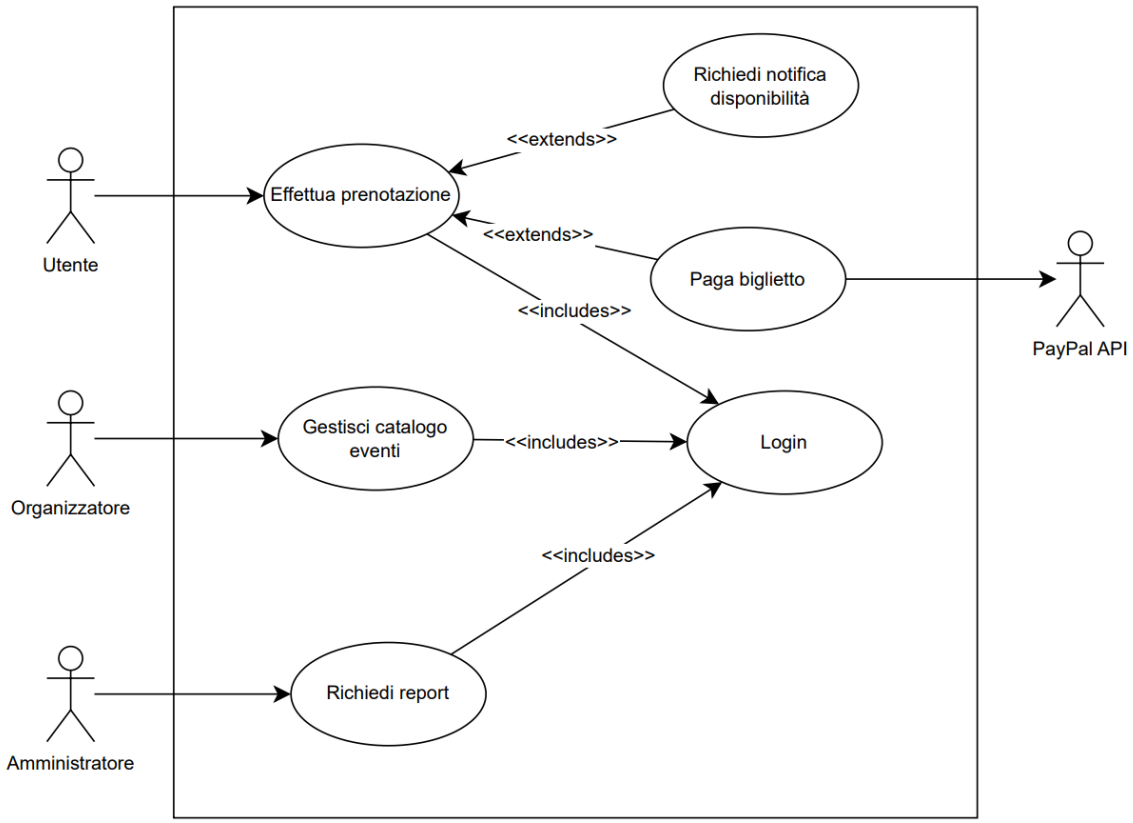
Il sistema deve *notificare* all'utente l'esito di ogni richiesta di prenotazione.

Dizionario:

Filtri: Categoria, Data, Prezzo, Località.
Verificare: il sistema controlla che all'utente sia associata al più una richiesta di prenotazione pendente.
Notificare: attraverso messaggio di errore o schermata di conferma.

1.4 USE CASES

1.4.1 Overview Diagram



*È possibile che un biglietto sia gratuito o che si possa pagare in loco.

1.4.2 Internal Steps

US-3:
Effettua prenotazione

*include login

1. L'utente seleziona l'opzione: "Prenota un evento".
2. Il sistema mostra i filtri di ricerca.
3. L'utente selezione i filtri desiderati.
4. L'utente conferma la selezione.
5. Il sistema verifica i filtri inseriti.
6. Il sistema effettua la ricerca.
7. Il sistema mostra i risultati.
8. L'utente seleziona un risultato.
9. Il sistema mostra il modulo dell'evento selezionato.
10. L'utente conferma i dati del modulo.
11. Il sistema valuta la richiesta di prenotazione.
12. Il sistema salva la prenotazione.
13. Il sistema notifica l'utente con l'esito della prenotazione.

Extensions:

6a. il database non trova risultati corrispondenti: il sistema notifica il fallimento e termina il caso d'uso.

11a. L'utente ha più di una prenotazione pendente: il sistema rifiuta la richiesta di prestito e notifica all'utente che ha raggiunto il numero massimo di richieste pendenti.

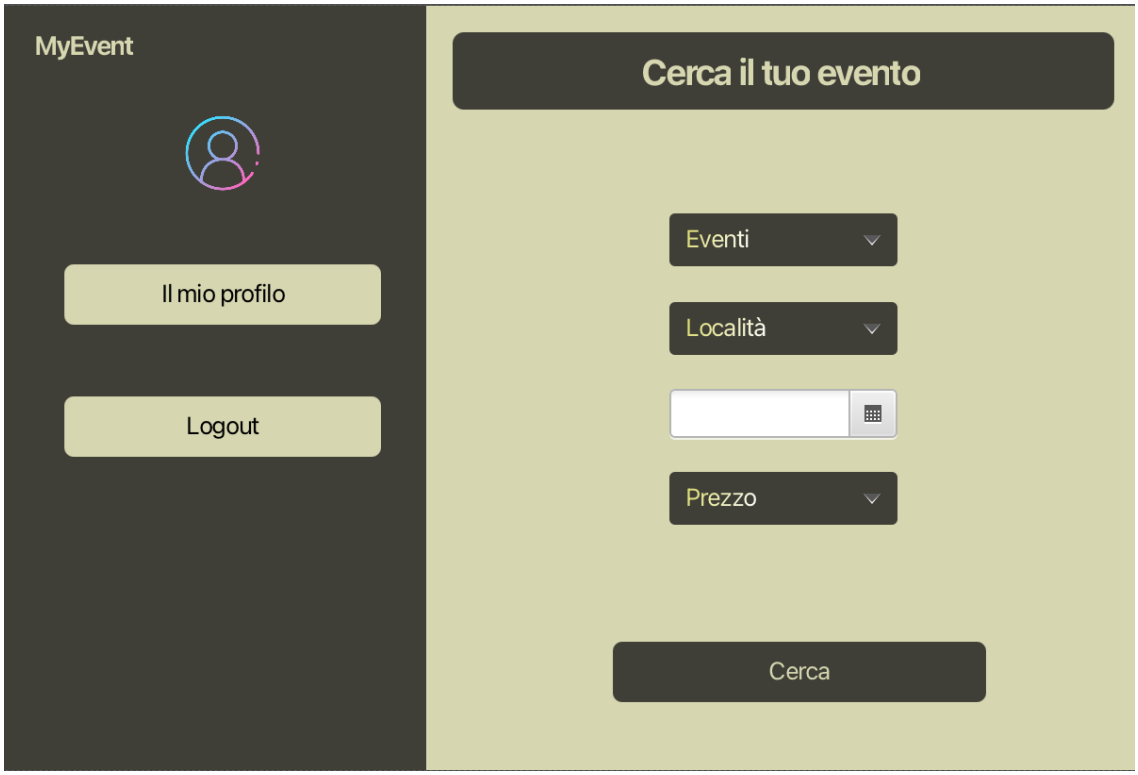
15a. L'utente non paga il biglietto entro le 24h successive all'accettazione della richiesta di prenotazione: il sistema elimina la richiesta di prenotazione e aggiorna lo stato dell'evento e del numero di posti ancora disponibili.

Dizionario:

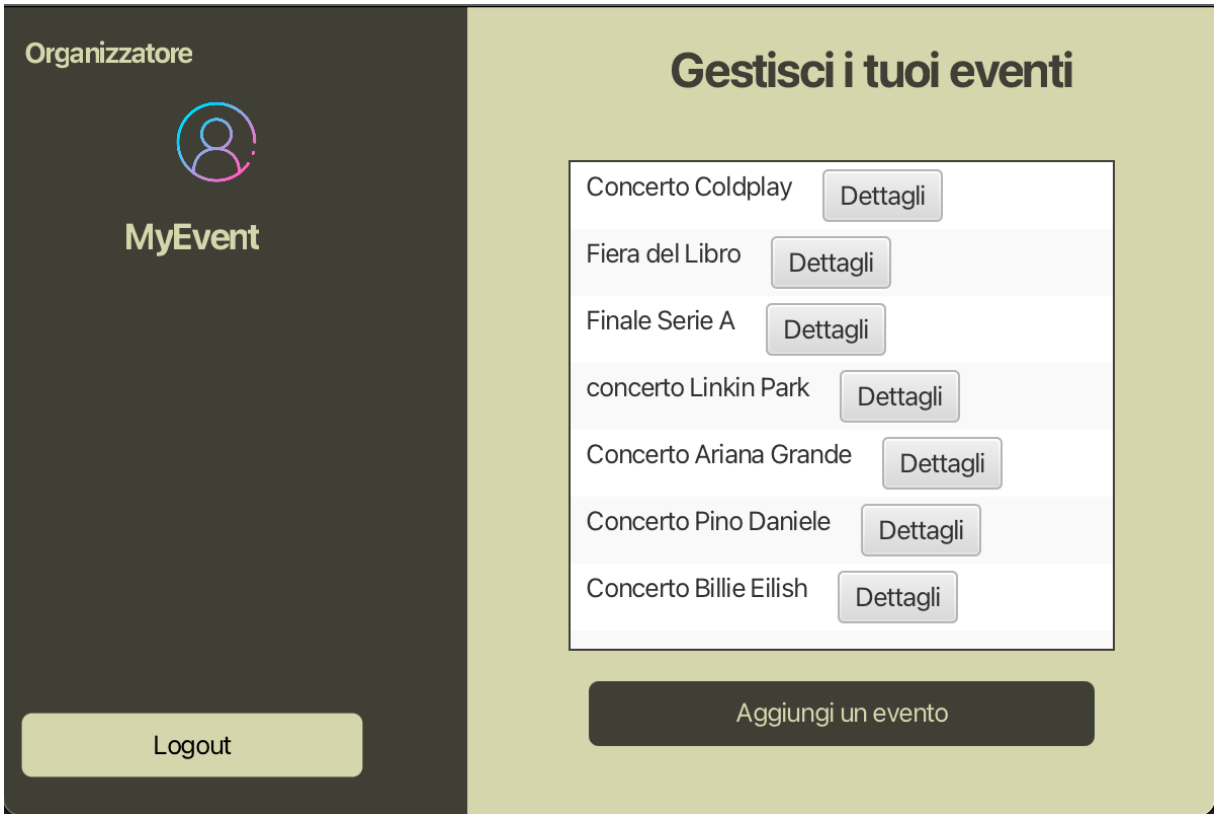
Filtri di ricerca: Categoria, Data, Prezzo, Località.
Risultati: Lista eventi.
Modulo dell'evento: Descrizione, Organizzatori, Orario di inizio.
Valuta la richiesta di prenotazione: Il sistema controlla che all'utente sia associata al più una richiesta di prestito.

2 STORYBOARDS

Schermata Home (Cliente)



Schermata Home (Organizzatore)



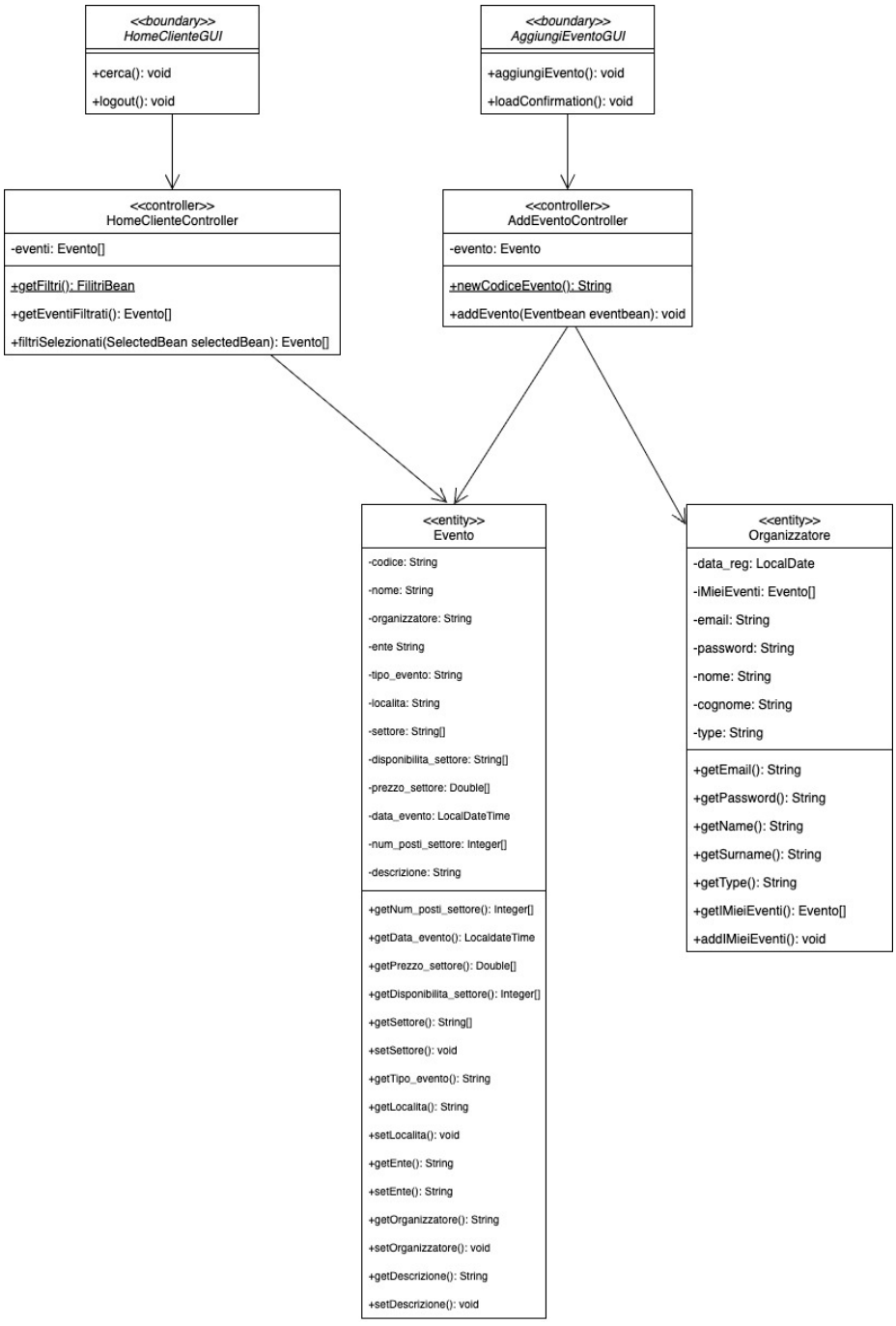
3 DESIGN

Nella parte di Design definiamo l'architettura e la scalabilità del progetto attraverso vari tipi di diagramma

3.1 Class Diagram

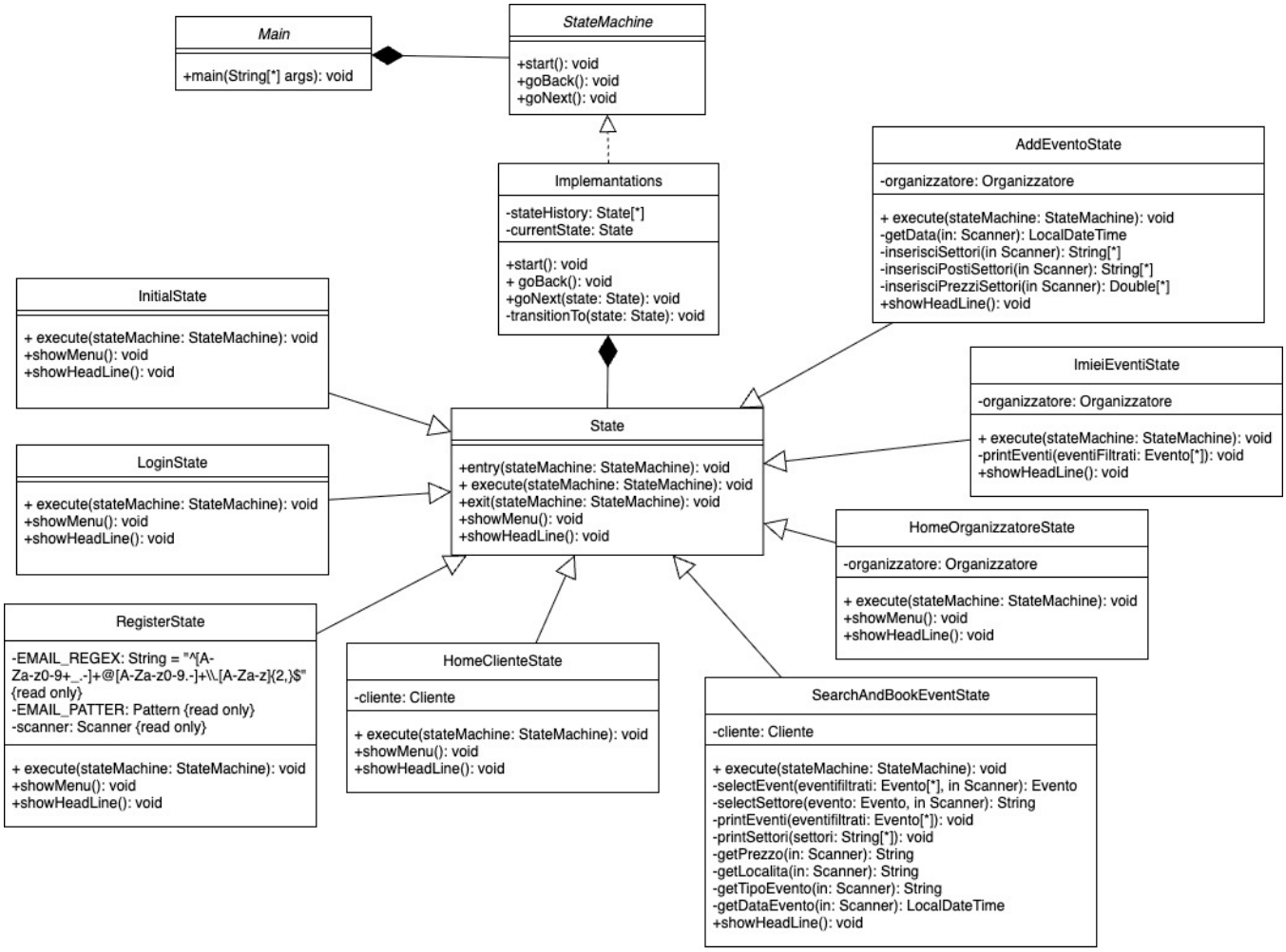
3.1.1 VOPC

-VOPC BCE relativo al caso d'uso "Aggiungi Evento"



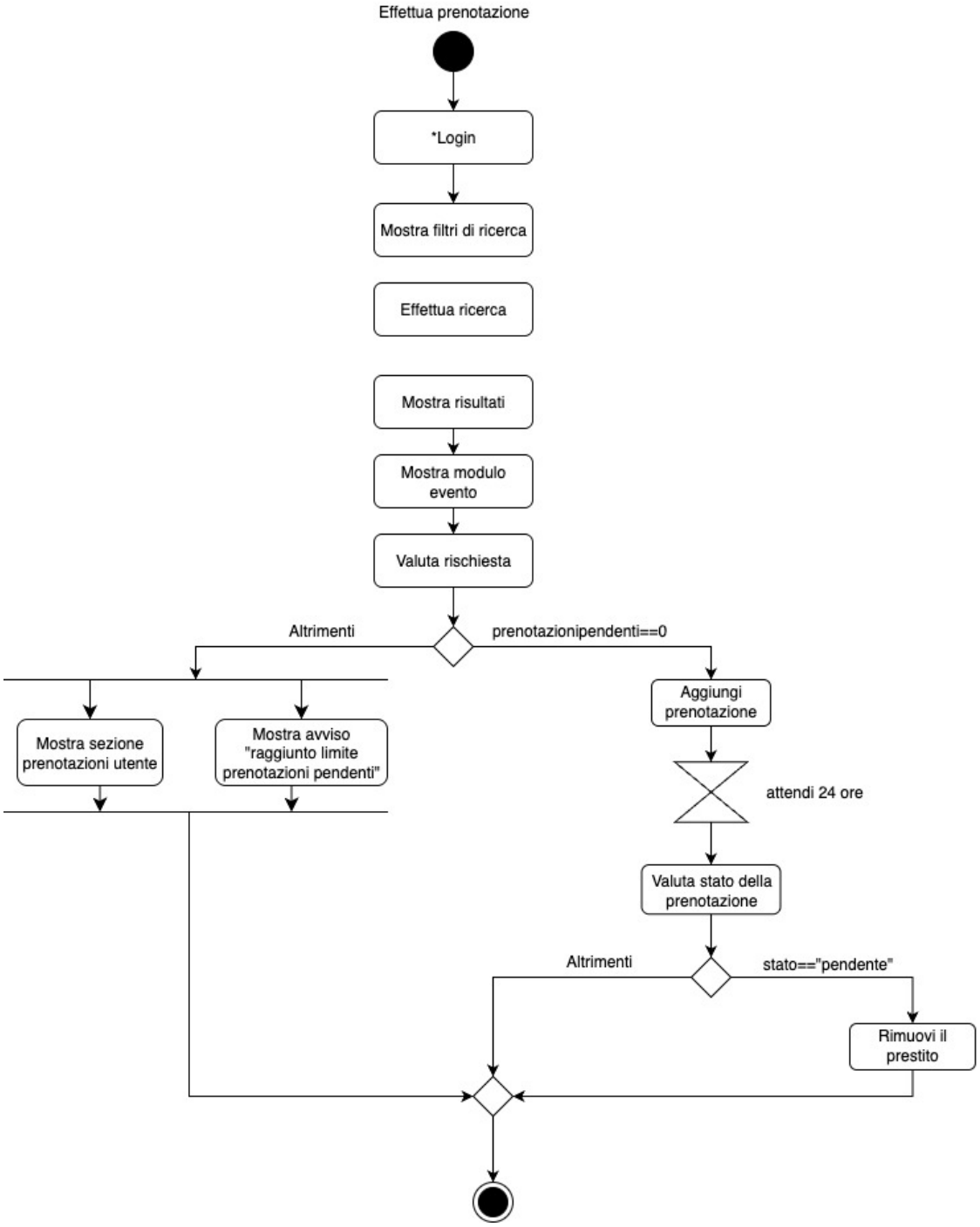
3.1.1.1 Design Level Diagram

Il Design Level Diagram riportato è relativo ad un pattern State



3.2 Activity Diagram:

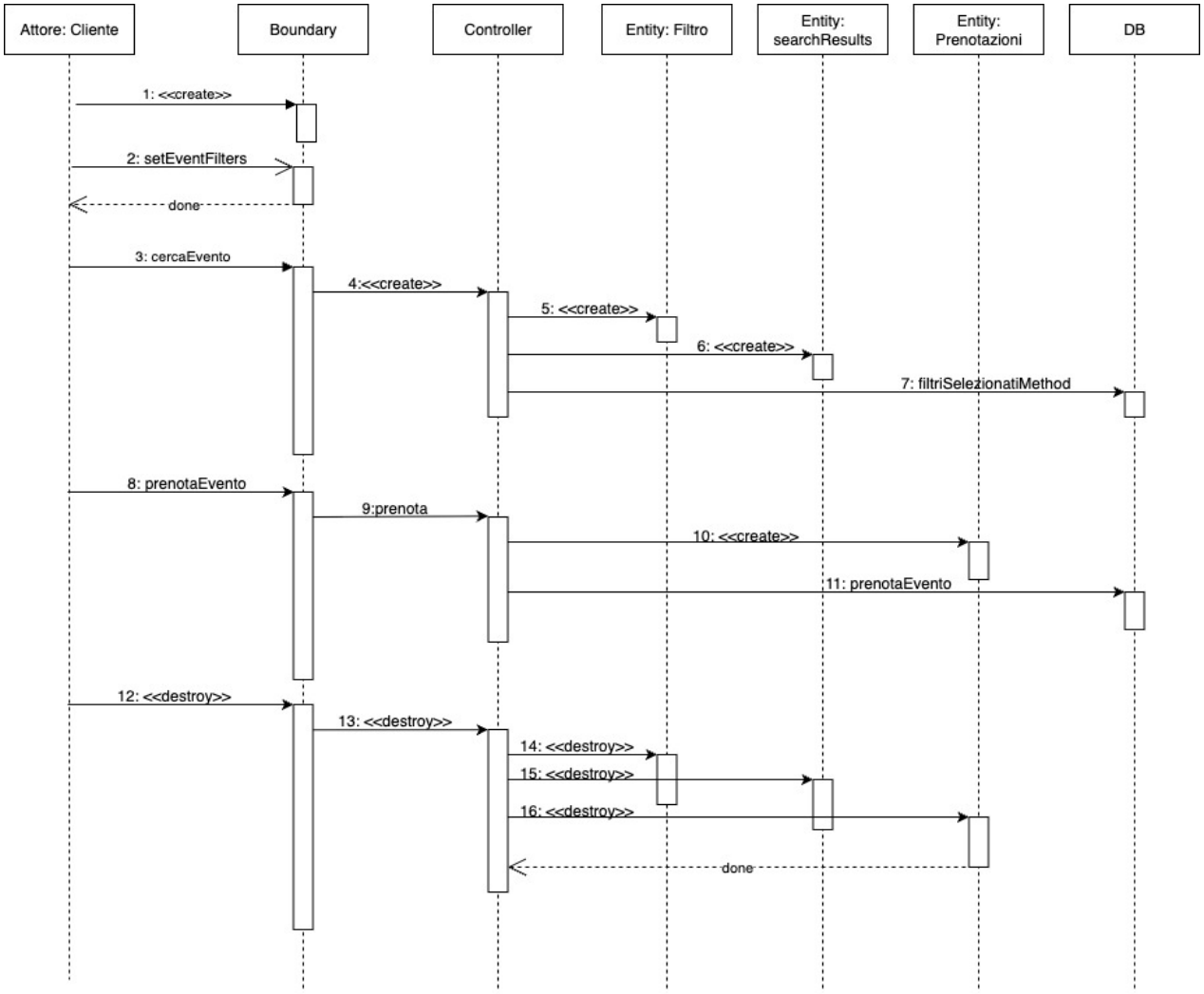
Activity Diagram relativo al caso d'uso: Effettua prenotazione.



Nota: Il timer non è stato implementato e la gestione nel caso di richiesta pendente è stata implementata diversamente.

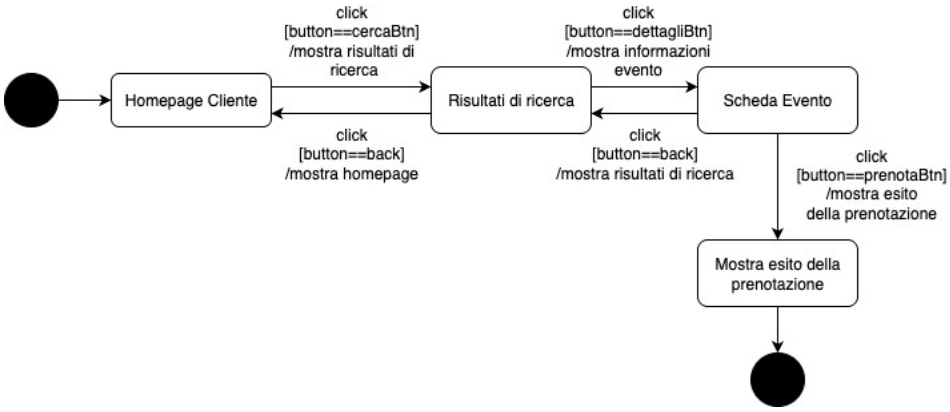
3.3 Sequence Diagram

Sequence Diagram relativo al caso d'uso: Effettua prenotazione



3.4 State Diagram

State Diagram relativo al caso d'uso: Prenota un evento



4 TESTING

"RegTest"
Classe di test JUnit che verifica la **registrazione dei clienti** in tre modalità di persistenza: **MYSQL**, **JSON** e **demo**. Modifica dinamicamente il tipo di persistenza, esegue la registrazione tramite **RegisterController** e controlla l'esito. Dopo i test, elimina l'utente di prova dal database o dal file JSON. Garantisce che il flusso di registrazione funzioni correttamente in tutti gli scenari.

"LoginTest"
Classe di test JUnit che verifica il **login** di **clienti** e **organizzatori** con tre modalità di persistenza: **MYSQL**, **JSON** e **demo**. Configura dinamicamente il tipo di persistenza, utilizza il **LoginController** per autenticare gli utenti e controlla che il recupero dei dati sia corretto. Garantisce il funzionamento della procedura di login in tutti gli scenari previsti.

"RicercaTest"
Classe di test JUnit che verifica il corretto funzionamento della **ricerca di eventi** e dell'applicazione dei **filtri** utilizzando il **HomeClienteController**. Esegue test con diverse modalità di persistenza (**MYSQL** e **demo**) e controlla che gli eventi vengano recuperati correttamente in base ai filtri selezionati.

"PrenotaTicketTest"
Classe di test JUnit che verifica il corretto funzionamento della **generazione dei codici di prenotazione** e della **prenotazione di eventi** tramite il **PrenotazioniController**. Simula il login di un cliente, la creazione di una nuova prenotazione e il salvataggio nel database o in modalità demo, con pulizia finale delle prenotazioni di test.

"AddEventoTest"
Classe di test JUnit che verifica il corretto funzionamento dell'**aggiunta di nuovi eventi** tramite **AddEventoController**. Simula il login dell'organizzatore, la creazione di un evento con dettagli completi (settori, prezzi, posti, disponibilità) e il salvataggio nel database o in modalità demo. Inoltre, testa la **generazione del nuovo codice evento** e include metodi per **cancellare l'evento e i relativi settori** dopo il test.

"IMieiEventiTest"
Classe di test JUnit che verifica il corretto funzionamento della funzionalità “i miei eventi” per un Organizzatore. Simula il login, recupera gli eventi organizzati dall’utente e li stampa a console. Include test separati per la persistenza in modalità demo e MySQL, e permette di cambiare dinamicamente il tipo di persistenza tramite il file `configurations.properties`.

5 SONARCLOUD

Per verificare il corretto funzionamento di questo progetto, è stato utilizzato il sistema SonarCloud. Questo sistema segnala i possibili bug, le vulnerabilità e verifica che le funzioni siano implementate correttamente. Puoi consultare i risultati del progetto al seguente link: https://sonarcloud.io/summary/overall?id=0310654_ispw2&branch=main