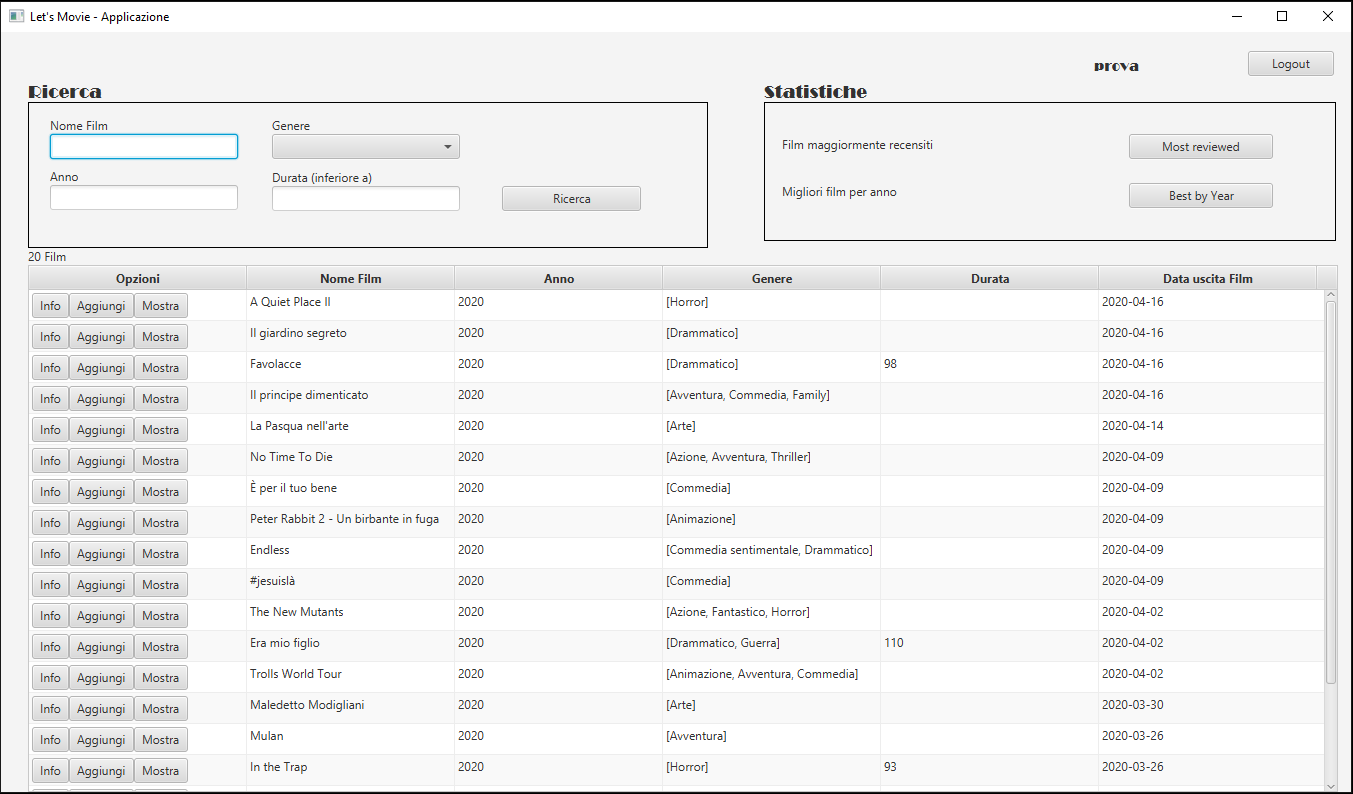
**Università di Pisa**

**Large-Scale and Multi-Structured Databases**

**(task2)**

****

**Let’s Movie**



**Davide Vigna**

Indice

[**Fase di analisi** 5](#_Toc39004934)

[Descrizione 5](#_Toc39004935)

[Attori principali 5](#_Toc39004936)

[Scenari 5](#_Toc39004937)

[Requisiti funzionali 7](#_Toc39004938)

[Requisiti non funzionali 8](#_Toc39004939)

[Diagramma dei casi d’uso 9](#_Toc39004940)

[**Fase di progettazione** 10](#_Toc39004941)

[Progettazione scenari 10](#_Toc39004942)

[Scenario di Autenticazione 10](#_Toc39004943)

[Scenario di Ricerca Film (Standard) 11](#_Toc39004944)

[Scenario di Dettaglio Film 12](#_Toc39004945)

[Scenario di Aggiungi Recensione 13](#_Toc39004946)

[Scenario di Mostra Recensioni 14](#_Toc39004947)

[Scenario di Mostra Statistiche 15](#_Toc39004948)

[Scenario di Ricerca Film (Admin) 16](#_Toc39004949)

[Scenario Aggiungi Film 17](#_Toc39004950)

[Scenario Statistiche Utenti 18](#_Toc39004951)

[Diagramma delle classi di analisi 19](#_Toc39004952)

[Scelta del database 20](#_Toc39004953)

[Scelta degli indici 22](#_Toc39004954)

[Scelta del replica set 23](#_Toc39004955)

[**Fase di implementazione** 24](#_Toc39004956)

[Struttura dei sorgenti 24](#_Toc39004957)

[Dettaglio query 26](#_Toc39004958)

[Test eseguiti 29](#_Toc39004959)

[Integrazione dei dati 30](#_Toc39004960)

[Configurazione del Replica Set 32](#_Toc39004961)

[**Valutazioni delle performance** 33](#_Toc39004962)

[Indice username: query di autenticazione 33](#_Toc39004963)

[Indice nome\_text: query ricerca film per nome 33](#_Toc39004964)

[Indice anno: query ricerca film per anno 34](#_Toc39004965)

[Indice genere: query ricerca film per genere 34](#_Toc39004966)

[Indici num\_recensioni e media\_voto: query ricerca film più recensiti 34](#_Toc39004967)

[Indice id\_film: query ricerca tutte le recensioni di un film 35](#_Toc39004968)

[Valutazione delle funzionalità: teorema CAP 36](#_Toc39004969)

[Valutazione della consistenza 36](#_Toc39004970)

[Valutazione della tolleranza ai guasti 37](#_Toc39004971)

[**Manuale d’uso** 39](#_Toc39004972)

[Utente Standard 40](#_Toc39004973)

[Utente Amministratore 44](#_Toc39004974)

# **Fase di analisi**

## Descrizione

Let’s Movie è un’applicazione per la ricerca e la recensione di film proiettati al cinema. I principali utilizzatori sono identificabili in due categorie di utenti: standard e amministratore(admin). Entrambi per poter utilizzare l’applicazione devono essere registrati. La fase di registrazione non è prevista all’interno di questa applicazione. Per poter utilizzare tutte le funzionalità è necessaria una fase preliminare di autenticazione.

Un utente di tipo standard dopo essersi autenticato può ricercare film impostando più filtri di ricerca, visualizzare informazioni dettagliate di un film, come i paesi di produzione, la trama e il cast; aggiungere una recensione; modificare o rimuovere una recensione precedentemente fatta e leggere la lista delle recensioni fatte da altri utenti relativamente ad un film. Una recensione consiste in un breve commento abbinato una valutazione numerica da 0 a 10. L’applicazione offre a questa tipologia di utenti la possibilità di visualizzare la lista dei film maggiormente recensiti e lista dei migliori film per ciascun anno.

Un utente di tipo amministratore, dopo essersi autenticato, può ricercare, aggiungere, modificare, rimuovere un film e può rimuovere le recensioni degli utenti che ritiene inopportune. L’applicazione offre a questa tipologia di utenti la possibilità di analizzare il numero di recensioni totali effettuate da ciascun utente della categoria standard, facendo una suddivisione tra numero di recensioni positive, negative e calcolando per ciascun utente il valore medio del voto relativo alle proprie recensioni.

## Attori principali

Gli attori principali sono due: utente standard e utente amministratore.

Un *utente standard* è quella persona che principalmente utilizza l’applicazione per ricercare film ed esprimere un proprio commento sui film che ha visto.

Un *utente amministratore* è quella persona gestisce i contenuti dell’applicazione: ricerca, aggiunge, modifica o rimuove un film nell’elenco dei film da mostrare e può eliminare le recensioni che ritiene inopportune.

Le due categorie si identificano entrambe con due informazioni *username* e *password*. Per poterle distinguere è necessaria una terza informazione: *ruolo*.

## Scenari

Entrambi gli attori condividono alcuni scenari che presentano delle leggere differenze a seconda della categoria di appartenenza.

Gli scenari condivisi sono:

* **Autenticazione**: il sistema mostra l’interfaccia per l’inserimento delle credenziali d’accesso.
* **Ricerca di un film**: il sistema mostra una serie di filtri e pulsanti per la ricerca di film, insieme ad una tabella contenente i film e il totale dei film ricercati; Su ciascuno di essi è possibile effettuare operazioni diverse a seconda della categoria di appartenenza.
* **Visualizza Recensioni**:il sistema visualizza in una nuova interfaccia la lista delle recensioni effettuate dagli utenti relativamente al film ricercato. Su ciascuna di esse è possibile effettuare operazioni diverse a seconda della categoria di appartenenza.

Un *utente standard* ha gli scenari di:

* **Visualizza film in dettaglio**: il sistema mostra maggiori informazioni sul film ricercato, in una nuova interfaccia.
* **Recensione di un film**: il sistema mostra una nuova interfaccia per l’inserimento di un commento e una valutazione numerica relativa ad un film. Questo scenario è accessibile anche in modifica.
* **Visualizza Statistiche Film**: il sistema mostra in una nuova finestra, la lista dei film che rispettano una certa caratteristica particolare.

Un *utente amministratore* ha gli scenari di:

* **Gestione di un Film**: il sistema apre una nuova interfaccia per l’inserimento o la modifica di un film.
* **Visualizza Statistiche Utenti:** il sistema mostra in una nuova finestra, la lista degli utenti dettagliando il numero di recensioni effettuate.

## Requisiti funzionali

Per poter accedere all’applicazione è necessario essere registrati. La fase di registrazione non viene implementata in questa applicazione. Un utente registrato può accedere all’applicazione inserendo le proprie credenziali che consistono in username e password. In caso di errata autenticazione, il sistema mostra all’utente un messaggio di errore.

Di seguito vengono dettagliati i requisiti funzionali per un *utente standard*.

A seguito dell’autenticazione, il sistema mostra un’interfaccia, dove sono già presenti gli ultimi 20 film, elencati in ordine decrescente per data di uscita. L’utente ha la possibilità di restringere il campo per la ricerca di un film attraverso la compilazione di appositi filtri, identificabili nel pannello *Ricerca*. Quando l’utente preme il pulsante *Ricerca*, il sistema mostra una nuova lista di film e il numero dei film totali ricercati. Per ciascun film nella lista il sistema mostra 3 pulsanti, ciascuno dei quali permette un’opzione diversa.

Premendo il primo pulsante, *Info*, il sistema mostra una nuova finestra contenente tutte le informazioni dettagliate del film, non presenti nella tabella di ricerca principale.

Premendo il secondo pulsante, *Aggiungi*, il sistema permette all’utente di esprime un proprio giudizio aggiungendo una recensione. Il sistema mostra in un’altra interfaccia un campo testuale, nel quale l’utente dovrà inserire una descrizione dettagliata della sua opinione e un campo numerico il quale indica il livello di gradimento del film.

Premendo il terzo pulsante, *Mostra*, il sistema apre un’interfaccia contenente la lista delle recensioni effettuate da tutti gli utenti, relativamente a quel film. Per ciascuna recensione nella lista, in cui l’utente recensore corrisponde all’utente attualmente autenticato, il sistema offre la possibilità di modificare oppure eliminare quella recensione. In tale casistica, se premuto il pulsante *Modifica*, l’utente verrà rimandato all’interfaccia per l’aggiunta/modifica recensione, altrimenti, pulsante *Elimina*, la recensione verrà rimossa.

L’interfaccia principale mostra anche una sezione relativa alle statistiche nella quale sono presenti 2 pulsanti. Il primo*, Most reviwed*, se cliccato, apre una nuova interfaccia dove mostra la lista dei film maggiormente recensiti. Il secondo, *Best by Year*, se cliccato, apre una nuova interfaccia dove mostra la lista dei migliori film per anno di uscita.

Infine nell’interfaccia principale è presente anche un pulsante di *Logout* che chiude l’interfaccia corrente ed apre nuovamente quella di autenticazione.

Di seguito vengono dettagliati i requisiti funzionali per un *utente amministratore*.

A seguito dell’autenticazione, il sistema mostra un’interfaccia per la ricerca di film uguale a quella utilizzata per utenti di tipo standard. La nuova interfaccia differisce dalla precedente per la presenza di un nuovo pulsante nel pannello di Ricerca, denominato *Aggiungi*. Se l’utente preme questo pulsante, il sistema apre una nuova interfaccia per l’inserimento di un film.

Per ciascun film nella lista il sistema mostra 3 pulsanti, ciascuno dei quali permette un’opzione diversa.

Premendo il primo pulsante *Mostra*, il sistema apre un’interfaccia contenente la lista delle recensioni effettuate da tutti gli utenti, relativamente a quel film. Per ciascuna recensione nella lista, è presente un pulsante *Elimina* per la rimozione della recensione.

Premendo il secondo pulsante *Modifica* il sistema apre l’interfaccia per l’inserimento di un film, caricando i dati relativi al film selezionato e consentendone la modifica.

Premendo il terzo pulsante *Elimina* il sistema rimuove dalla lista dei film, il film selezionato.

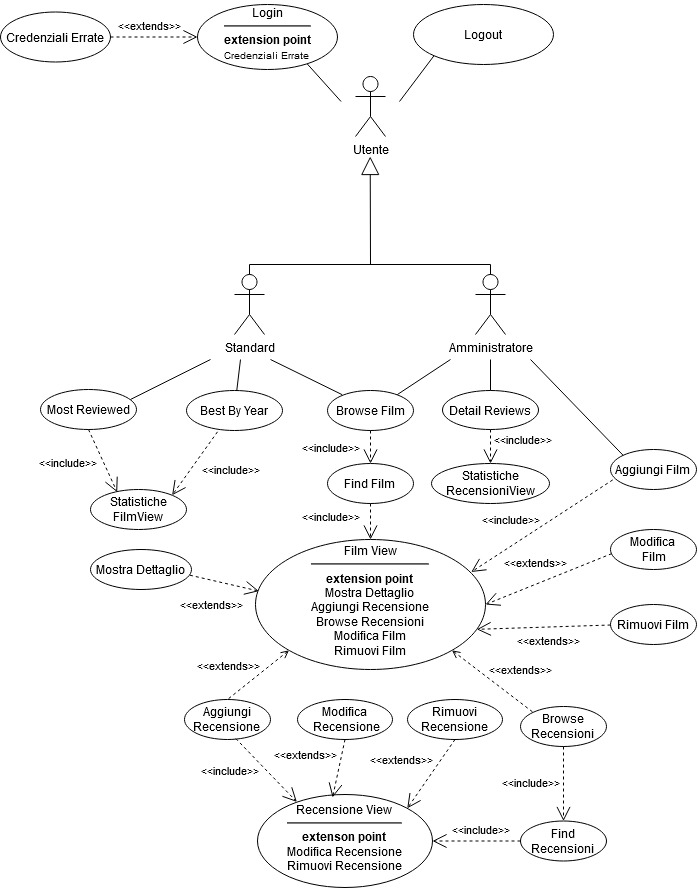
L’interfaccia principale mostra anche una sezione relativa alle statistiche nella quale è presente un solo pulsante *Details Reviews*, il quale se premuto, il sistema apre una nuova interfaccia per la visualizzazione del numero di recensioni nel dettaglio per ciascun utente.

## Requisiti non funzionali

I requisiti non funzionali si possono riassumere nei seguenti punti:

* **Flessibilità**: deve essere possibile lavorare con strutture dati non rigide e facilmente modificabili senza che questo comporti grandi modifiche.
* **Scalabilità**: l’applicazione deve essere in grado di gestire un grande volume di informazioni. La suddivisione del carico di lavoro su più server garantisce un minor tempo di risposta;
* **Replicabilità e Consistenza:** si ritiene necessario effettuare più copie dei dati dell’applicazione. Queste, devono essere costantemente aggiornate in modo da garantire letture consistenti, a scapito di una minore disponibilità. In questo modo si riduce la possibilità che l’intero sistema rimanga offline permettendo comunque all’utente di accedere ed utilizzare l’applicazione, anche in caso di guasti sui singoli server.
* **Modularità**: la struttura dell’applicazione deve essere ben organizzata e modulare, in modo da poter utilizzare funzionalità comuni a più utenti, in punti diversi dell’applicazione. Una struttura modulare permette inoltre di effettuare più agevolmente modifiche o estensioni delle funzionalità presenti nell’applicazione.
* **Performance:** il sistema deve rispondere velocemente alle richieste dell’utente, riducendo al minimo i tempi di attesa;
* **Semplicità:** l’applicazione deve essere *user-friendly*, la maggior parte delle operazioni deve essere intuibile da un utente medio.
* **Sicurezza:** è necessario garantire riservatezza alle informazioni salvate nella base di dati, pertanto è necessario che ciascun utente effettui un’autenticazione preliminare. Questa permette anche di determinare il livello di privilegio interno, discriminando gli utenti di categoria standard da quelli di categoria amministratore.

## Diagramma dei casi d’uso



Note:

La figura precedente indica il diagramma dei casi d’uso complessivo. Su ciascuna delle view principali (Film View e Recensione View) è possibile avere casi d’uso differenti a seconda della categoria di utente con cui si accede all’applicazione.

La ***Film View*** di un utente standard presenta delle estensioni ai casi d’uso, *Mostra Dettaglio*, *Aggiungi Recensione*, mentre per un utente amministratore presenta delle estensioni ai casi d’uso *Modifica Film*, *Rimuovi* Film e un’inclusione al caso d’uso *Aggiungi Film*. Entrambe condividono l’estensione al caso d’uso *Brose Recensioni* e l’inclusione al caso d’uso *Brose-Find Film.*

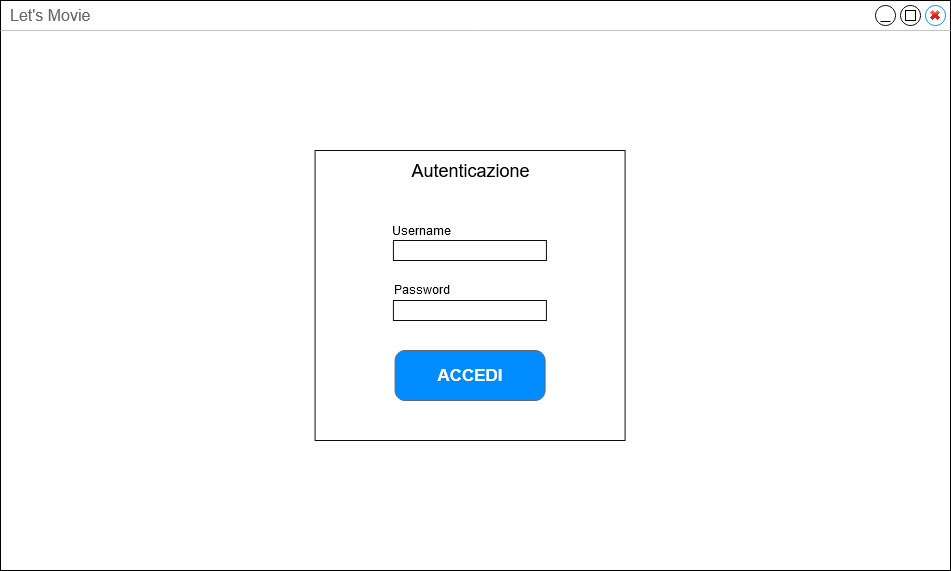
La ***Recensione View*** di un utente standard presenta un’estensione al caso d’uso *Modifica Recensione*, e condivide l’estensione al caso d’uso *Rimuovi Recensione* con un utente di categoria amministratore.

# **Fase di progettazione**

In questa fase vengono illustrati i vari scenari dell’applicazione, descrivendo il loro funzionamento approssimativo mediante l’ausilio di pseudo codice e mockup. Successivamente vengono individuate le principali strutture dati per la memorizzazione e la gestione delle informazioni. Infine viene effettuato un breve studio di fattibilità riguardo alla tipologia di database da utilizzare per l’implementazione dell’applicazione e su come organizzare le varie strutture dati.

## Progettazione scenari

### **Scenario di Autenticazione**



1. Il sistema mostra i campi *Username*, *Password* e il pulsante *Accedi*.
2. L’utente inserisce il proprio username e la propria password.
3. IF utente preme *Accedi*.

3.1 Il sistema controlla credenziali.

3.2 IF credenziali sono corrette.

3.2.1 IF credenziali sono di Utente Categoria Standard

3.2.1.1 Il sistema mostra Scenario Ricerca Film (Standard)

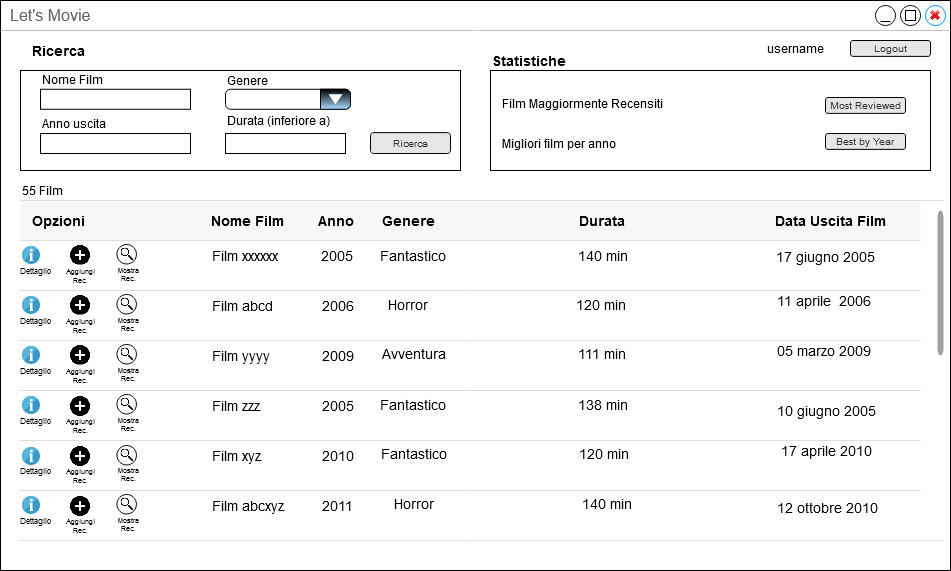
3.2.2 ELSE IF credenziali sono di Utente Categoria Admin

3.2.2.1 Il sistema mostra Scenario Ricerca Film (Admin)

3.3 ELSE

3.3.1 Il sistema mostra etichetta “Credenziali Errate”.

### **Scenario di Ricerca Film (Standard)**



1. Il sistema mostra il pannello con i campi di *Ricerca*, il pannello delle *Statistiche*, un’etichetta con *Username* utente autenticato, il pulsante di *Logout,* la *Tabella dei Film* con gli ultimi 20 Film, in ordine cronologico di uscita e un’etichetta con il *Totale* dei film ricercati.
2. L’utente inserisce *Nome Film* e/o *Genere* e/o *Anno Uscita* e/o *Durata (inferiore a)*, all’interno del pannello di Ricerca.
3. IF utente preme *Ricerca*
   1. Il sistema mostra nella *Tabella dei Film*, la lista dei film che corrisponde ai criteri di ricerca e aggiorna *Totale* film trovati.
   2. FOR EACH Film
      1. Il sistema mostra i pulsanti *Dettaglio*, *Aggiungi*, *Mostra* e le informazioni del film: *Nome Film*, *Anno*, *Genere*, *Durata*, *Data Uscita*.
      2. IF utente preme *Dettaglio*
         1. Il sistema mostra Scenario Dettaglio Film.
      3. IF utente preme *Aggiungi*
         1. Il sistema mostra Scenario Aggiungi Recensione.
      4. IF utente preme *Mostra*
         1. Il sistema mostra Scenario Visualizza Recensioni.
4. IF Utente preme *Most Reviewed*
   1. Il sistema carica lista dei film con caratteristica “*Più recensiti in assoluto*”.
   2. Il sistema mostra Scenario Statistiche Film.
5. IF Utente preme *Best by Year*
   1. Il sistema carica lista dei film con caratteristica “*Migliori film per anno*”.
   2. Il sistema mostra Scenario Statistiche Film.
6. IF Utente preme Logout
   1. Il sistema mostra Scenario Autenticazione.

Criteri di ricerca:

* IF *Nome Film* presente, la ricerca avviene sui film il cui nome contiene la stringa inserita.
* IF *Genere* presente, la ricerca avviene sui film il cui genere è quello indicato.
* IF *Anno Uscita* presente, la ricerca avviene sui film il cui anno è quello indicato.
* IF *Durata (inferiore a)* presente, la ricerca avviene sui film la cui durata è inferiore a quella indicata.

“*Più recensiti in assoluto*”: rientrano in questa tipologia, tutti quei film che hanno ricevuto almeno una recensione, con durata maggiore o uguale a 60 minuti, con un genere diverso da documentario elencati in ordine decrescente per numero recensioni – media voto.

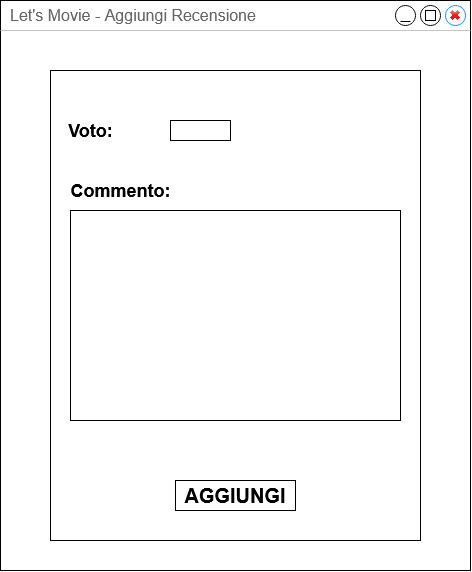
“*Migliori film per anno*”: rientrano in questa tipologia, tutti quei film che hanno ottenuto più di 5 recensioni e la cui media voto è maggiore degli altri film dello stesso anno, elencati per anno crescente.

### **Scenario di Dettaglio Film**



1. Il Sistema mostra una nuova finestra con informazioni del Film: *Titolo*, *Genere*, *Uscita*, *Durata*, *Paesi Produzione*, *Numero Recensioni*, *Media Voto*, *Trama Completa*, *Cast*.

### **Scenario di Aggiungi Recensione**

****

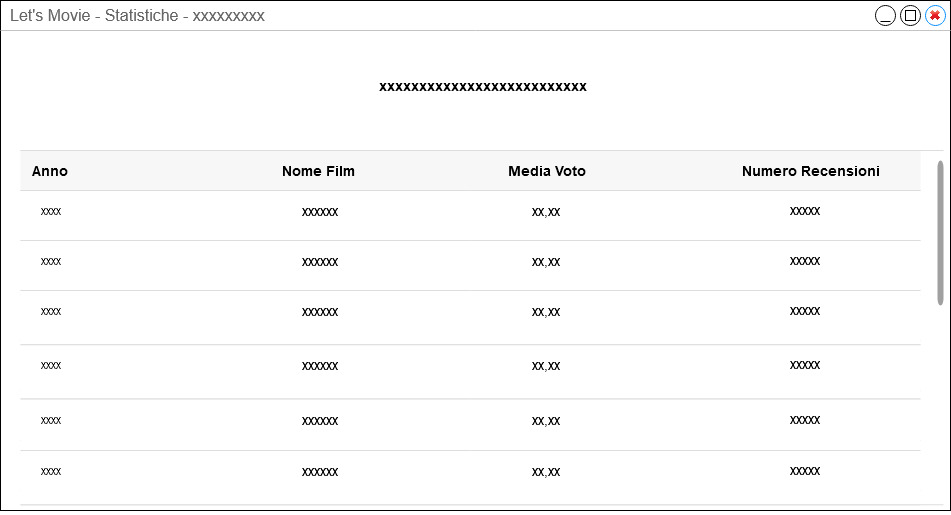
1. Il Sistema mostra una nuova finestra con due campi di testo *Voto*, *Commento*.
2. L’utente inserisce il proprio *Voto* e *Commento*.
3. IF accesso in inserimento
   1. Il Sistema mostra pulsante *Aggiungi*.
   2. IF Utente preme *Aggiungi*
      1. Il Sistema registra *Voto* e *Commento* negli archivi.
      2. Il Sistema chiude la finestra corrente.
4. IF accesso in modifica
   1. Il Sistema mostra pulsante *Modifica*
   2. IF Utente preme *Modifica*
      1. Il Sistema aggiorna *Voto* e *Commento* negli archivi.
      2. Il Sistema chiude la finestra corrente.

### **Scenario di Mostra Recensioni**

****

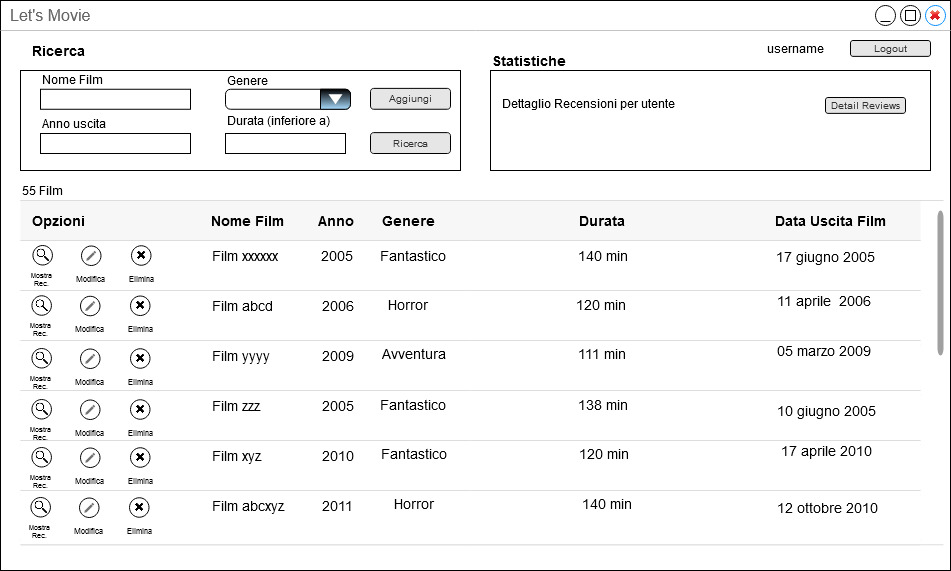
1. Il Sistema mostra una nuova finestra con una *Tabella Recensioni* nella quale è presente la lista delle Recensioni associata al Film.
2. FOR EACH Recensione
   1. IF Utente corrente è di tipologia Standard
      1. IF Utente corrente = Utente Recensione
         1. Il Sistema mostra i pulsanti *Modifica*, *Elimina*.
         2. IF Utente preme *Modifica*
            1. Il Sistema carica informazioni Recensione.
            2. Il sistema mostra Scenario Aggiungi Recensione
         3. IF Utente preme *Elimina*
            1. Il Sistema rimuove la Recensione dagli archivi.
            2. Il Sistema rimuove la Recensione da *Tabella Recensioni*.
   2. IF Utente corrente è di tipologia Admin
      1. Il Sistema mostra pulsante *Elimina*.
      2. IF Utente preme *Elimina*
         1. Il Sistema rimuove la Recensione dagli archivi.
         2. Il Sistema rimuove la Recensione da Tabella Recensioni.
   3. Il Sistema mostra informazioni *Utente*, *Voto*, *Commento*.

### **Scenario di Mostra Statistiche**



1. Il sistema mostra una nuova finestra con un’etichetta *Descrizione Statistiche* e una *Tabella di Statistiche Film*.
2. FOR EACH Statistica Film
   1. Il sistema mostra le informazioni relative a *Anno*, *Nome Film*, *Media Voto*, *Numero recensioni*.

### **Scenario di Ricerca Film (Admin)**

****

1. Il sistema mostra il pannello con i campi di *Ricerca*, il pulsante *Aggiungi*, il pannello delle *Statistiche*, un’etichetta con *Username* utente autenticato, il pulsante di *Logout,* la *Tabella dei Film* con gli ultimi 20 Film, in ordine cronologico di uscita e un’etichetta con il *Totale* dei film ricercati.
2. L’utente inserisce *Nome Film* e/o *Genere* e/o *Anno Uscita* e/o *Durata*, all’interno del pannello di Ricerca.
3. IF utente preme *Ricerca*
   1. Il sistema mostra nella *Tabella dei Film*, la lista dei film che corrisponde ai criteri di ricerca e aggiorna *Totale* film trovati.
   2. FOR EACH Film
      1. Il sistema mostra i pulsanti *Mostra*, *Modifica*, *Elimina* e le informazioni del film: *Nome Film*, *Anno*, *Genere*, *Durata*, *Data Uscita*.
      2. IF utente preme *Mostra*
         1. Il Sistema mostra Scenario Visualizza Recensioni.
      3. IF utente preme *Modifica*
         1. Il Sistema mostra Scenario Aggiungi Film.
      4. IF utente preme *Elimina*
         1. Il Sistema rimuove Film dagli archivi.
         2. Il Sistema rimuove Film dalla *Tabella Film*.
4. IF Utente preme *Detail Reviews*
   1. Il sistema carica lista degli utenti con caratteristica “*Dettaglio recensioni*”.
   2. Il sistema mostra Scenario Statistiche Utenti.
5. IF Utente preme *Logout*
   1. Il sistema mostra Scenario Autenticazione.

Criteri di ricerca: sono gli stessi indicati nello scenario di ricerca (Standard).

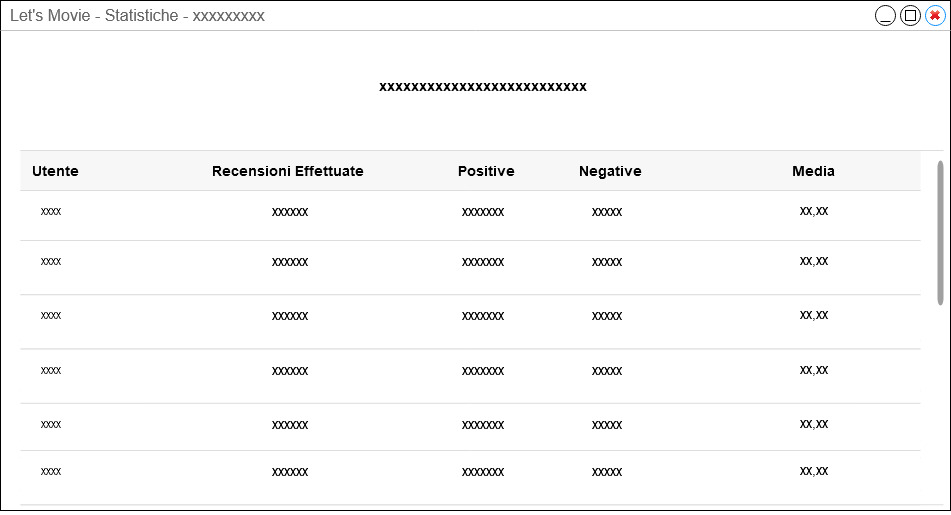
“Dettaglio recensioni”: gli utenti recensori, visualizzati in ordine decrescente di numero recensioni effettuate, specificando quante sono quelle positive, quante negative e la media voto dell’utente.

### **Scenario Aggiungi Film**

****

* + - 1. Il Sistema mostra una nuova finestra con i campi di testo *Titolo*, *Anno*, *Genere*, *Paesi produzione*, *Durata*, *Data Uscita* e *Trama Completa*.
      2. L’utente inserisce i campi *Titolo*, *Anno*, *Genere*, *Paesi produzione*, *Durata*, *Data Uscita* e *Trama Completa*.
      3. IF accesso in inserimento
  1. Il Sistema mostra pulsante *Aggiungi*.
  2. IF Utente preme *Aggiungi*
     1. IF *Titolo* AND *Anno* AND *Genere* presenti
        1. Il Sistema registra un nuovo Film negli archivi.
        2. Il Sistema chiude la finestra corrente.
        3. IF accesso in modifica
  3. Il Sistema mostra pulsante *Modifica*.
  4. IF Utente preme *Modifica*
     1. IF *Titolo* AND *Anno* AND *Genere* presenti
        1. Il Sistema aggiorna Film negli archivi.
        2. Il Sistema chiude la finestra corrente.

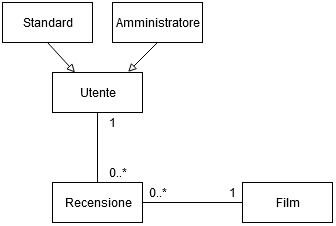
### **Scenario Statistiche Utenti**

****

* + - 1. Il sistema mostra una nuova finestra con un’etichetta *Descrizione Statistiche* e una *Tabella di Statistiche Utenti*.
      2. FOR EACH Statistica Utente
  1. Il sistema mostra le informazioni relative a *Utente, Recensioni Effettuate*, *Positive*, *Negative*, *Media*.

## Diagramma delle classi di analisi

Le principali strutture dati individuate sono tre e sono in relazione fra loro come illustrato nel seguente diagramma.



Un utente può appartenere solamente a due tipologie: standard o amministratore. Un utente *può* rilasciare nessuna o molte recensioni. Al contrario una recensione deve essere in relazione con un solo Utente. Ogni recensione è associata necessariamente ad un film, ed un film *può* avere nessuna o più recensioni.

La classe *utente* contiene le seguenti informazioni:

* Username (nome necessario per l’autenticazione)
* Password (parola segreta per completare l’autenticazione)
* Ruolo (discriminante tra categoria Standard e Amministratore)

La classe *film* contiene le seguenti informazioni:

* Nome (il titolo)
* Anno (l’anno di uscita)
* Data Uscita (giorno di uscita film gg/mm/aaaa)
* Durata (il numero di minuti complessivo)
* Trama Completa (la descrizione completa)
* Genere (il/i genere/i di appartenenza)
* Paesi Produzione (il/i paese/i dove è stato prodotto)
* Cast (elenco di nomi di persone e ruoli che hanno interpretato nella produzione)

La classe *recensione* contiene le seguenti informazioni:

* Voto (valutazione numerica)
* Commento (giudizio testuale)
* Autore (colui che ha recensito)
* Film (film su cui è stata fatta la recensione)

## Scelta del database

Analizzando i requisiti funzionali e non funzionali si conviene che la scelta migliore è l’utilizzo di un database non relazionale. In particolare la tipologia *Document* *Database* è quella che meglio soddisfa i requisiti. Questa tipologia prevede l’archiviazione delle informazioni in *documenti* all’interno di raccolte chiamate *collection*. Ciascun documento non ha una struttura fissa e ciò ne permette un’alta flessibilità. Inoltre è possibile suddividere il carico di lavoro dell’intera base di dati su più server attraverso la tecnica di Sharding ed insieme ad un’opportuna replicazione dei dati ne garantisce una maggior disponibilità.

Per questa applicazione si utilizza il database MongoDB per l’archiviazione dei dati.

L’organizzazione dei documenti e delle collection si basa su come i dati vengono utilizzati e richiesti dall’utente.

Per questa ragione si sono identificate principalmente 3 Collection. Di seguito vengono elencate insieme ai campi che ricorrono spesso nei vari documenti.

* **Utenti** 
  + \_id (objectId)
  + username (stringa)
  + password (stringa)
  + ruolo (stringa)
* **Film**
  + \_id (objectId)
  + nome (stringa)
  + anno (intero)
  + data\_uscita (data)
  + tramaCompleta(stringa)
  + genere (array di stringhe)
  + paesi\_prod (array di stringhe)
  + durata\_min(intero)
  + cast (array di embedded document)
  + media\_voto (double)
  + num\_recensioni (intero)
* **Recensioni**
  + \_id (objectId)
  + commento (stringa)
  + voto (double)
  + id\_film (objectId)
  + id\_utente (objectId)
  + username (stringa)

La collection *Utenti* viene utilizzata principalmente per la fase di autenticazione dove avviene una ricerca in lettura della informazioni username-password. Non vi si accede mai in scrittura, poiché non è prevista una fase di registrazione. Per questi motivi è stato ritenuto opportuno separarla delle altre.

La collection *Film* viene utilizzata sia in lettura che in scrittura in diversi punti dell’applicazione. Si tiene conto del fatto che la frequenza con cui viene fatta una scrittura è molto minore rispetto a quella di una lettura, perché può avvenire solamente da parte di un utente di tipologia Admin, durante il caso d’uso di aggiungi e modifica film. Le letture invece possono essere fatte da entrambe le categorie, durante la fase di ricerca di un film e dalla categoria utente Standard durante la ricerca delle statistiche (film maggiormente recensiti, film migliori per anno). Si sceglie di incapsulare informazioni come *genere*, *paesi\_prod*, *cast* all’interno di ogni singolo documento Film. Questa scelta è dettata dal fatto che queste informazioni vengono richieste insieme e non sono richieste particolari gestioni per tali campi. Il caso d’uso Mostra dettaglio ne è un esempio.

È stato ritenuto necessario inserire anche le informazioni *media\_voto* e *num\_recensioni*, al fine di evitare la ricerca e il calcolo di questi valori all’interno della collection Recensioni ogni volta che l’utente richiede la visualizzazione delle statistiche dei film. Infatti, solo dalle recensioni si è in grado di determinare quante ne sono state fatte e quale è la media che identifica un film migliore rispetto ad un altro. Introducendo queste “ridondanze” si è consapevoli del fatto che è necessario mantenere questi valori aggiornati ad ogni inserimento, modifica e cancellazione di una recensione. In questo modo si privilegiano ancora una volta le letture, riducendone i tempi di attesa, a scapito delle scritture.

La collection *Recensioni* viene utilizzata per la raccolta di tutte le recensioni relative ai Film. Anche questa è soggetta a letture e scritture in diversi punti dell’applicazione. I motivi per cui si è scelto di raggruppare le Recensioni in una collection a parte invece che inserire un array di embedded document Recensioni all’interno del documento Film sono diversi:

Velocità nell’ottenimento informazione di una recensione: filtrare una collection distinta rispetto a scorrere l’intero array di embedded document risulta più veloce ed è possibile utilizzare indici per ridurre il numero di letture.

Facilità nel trovare l’utente che ha effettuato una recensione: per la ricerca del dettaglio di tutte le recensioni effettuate da un utente. Nel caso d’uso “dettaglio recensioni”, risulta più rapido trovare l’utente recensore e contare il numero di recensioni effettuate da ciascun utente. Utilizzando embedded document invece risulta problematico in quanto occorre scorrere l’intera collection di documenti film e per ciascuno, scorrere la lista delle recensioni associate, quindi effettuare il conto relativo a ciascun utente.

Ciascun documento è di dimensione limitata: sia i documenti della collection Film che quelli di Recensione, non corrono il rischio di superare la dimensione massima consentita per i documenti in MongoDB (16MB). Utilizzando un array di embedded document, invece, all’aumentare degli utenti e del volume di recensioni anche la dimensione del documento Film aumenta e il database deve provvedere ad una rilocazione della memoria. Con una collection distinta si evitano anche riduzioni di performance dovute a tale rilocamento.

Anche in questa collection è stata introdotta una ridondanza, *username*, per evitare di accedere in lettura, per ogni recensione, alla collection Utenti, per reperire il nome dell’utente che ha effettuato una recensione. Considerando Recensioni una collection a sé stante è necessario introdurre due riferimenti, *id\_film* e *id\_utente*, necessari per riconoscere rispettivamente a che film e a che utente è associata una recensione. È bene sottolineare che in un document database, non esiste l’integrità referenziale e che qualora questi campi dovessero essere modificati nelle collection Film e Utenti, i due valori presenti nella collection Recensioni non verrebbero aggiornati. Per esempio nel caso d’uso Rimuovi Film, se viene eliminato un Film a cui sono già state fatte delle recensioni, queste rimangono memorizzate all’interno della base di dati. In generale un’operazione di aggiornamento/rimozione di un riferimento a documento di altra collection è abbastanza rara e non è prevista in altri casi d’uso dell’applicazione.

## Scelta degli indici

Si prevede che l’applicazione debba gestire grandi volumi di dati, pertanto è necessario definire opportunamente degli indici sulla base di dati in modo da velocizzare la gestione dei dati e ridurre il tempo di attesa durante le ricerche. Di seguito vengono elencati gli indici introdotti su ciascuna collection, omettendo gli indici che MongoDB crea default durante la creazione di una collection.

Utenti Collection:

* indice *username*: per velocizzare la fase di autenticazione. Si prevede infatti che gli accessi siano molti e che l’utente debba avere accesso all’applicazione nel minor tempo possibile.

Film Collection:

* indice *nome\_text*: per velocizzare la ricerca di film i cui nomi contengono una determinata stringa. Questo indice rientra nella categoria *Advance Indexes* - *Text Index* messa a disposizione da MongoDB. È pensato per tutti quegli utenti che non si ricordano come si chiama completamente un film, ma ricordano solo una parte di nome.
* indice *anno*: per velocizzare la ricerca di film il cui anno di uscita viene indicato nel campo anno. Questo indice è utile se si vogliono ricercare tutti i film usciti in uno specifico anno.
* indice *genere*: per velocizzare la ricerca di film il cui genere viene indicato nella combo genere. Questo indice è utile per la ricerca di tutti i film con uno specifico genere.
* indice composto su *num\_recensioni* e *media\_voto* velocizzare il riordinamento nel caso d’uso Most Reviewed.

Per il campo durata (inferiore a) non è stato ritenuto opportuno gestire un indice, in quanto non in tutti i documenti film viene specificato.

Recensioni Collection:

* indice *id\_film*: per velocizzare la ricerca di tutte le recensioni associate ad un determinato film. Questa operazione viene effettuata frequentemente da entrambe le tipologie di utenti (Standard e Admin) nel caso d’uso “Mostra Recensioni”.

**Architettura dell’applicazione**

Col fine di soddisfare i requisiti di modularità, si è deciso di sviluppare l’applicazione in linguaggio Java, linguaggio orientato agli oggetti che permette l’organizzazione del codice in classi. Per lo sviluppo si utilizza l’IDE Eclipse. Con l’aggiunta del plug-in *SceneBuilder* all’interno di Eclipse è possibile costruire le varie interfacce dell’applicazione in maniera interattiva e veloce. Grazie alle librerie grafiche di Java FX si può inserire molti componenti interattivi nelle proprie interfacce, rendendo l’approccio con l’utilizzatore finale molto semplice (*user-friendly*).

L’architettura dell’applicazione è basata sul modello MVC (Model-View-Controller). Questo pattern architetturale prevede la suddivisione dei vari componenti software che compongono l’applicazione a seconda dei compiti che svolgono:

* Model: rappresenta i dati da gestire e fornisce i metodi per accedere ad essi
* View: mostra i dati all’utente il quale a sua volta vi può interagire
* Controller: implementa la logica di gestione dei dati, a seguito di interazione con l’utente o di un cambiamento di stato del sistema, provvede alla mostra a video degli stessi.

Ciascuna delle 3 è indipendente dalle altre ed esegue un compito diverso.

In questa applicazione i model e i controller sono rappresentati mediante classi java (in Eclipse), mentre le view mediante file fxml (SceneBuilder). Ogni View ha associato un proprio Controller.

Per la visualizzazione dei risultati di ricerca in ciascuna tabella, si ritiene opportuno prevedere un meccanismo che mostri solamente un gruppo ristretto di risultati alla volta. Il numero dei risultati è un parametro modificabile da costante locale. L’utente utilizzando la barra di scorrimento laterale, può scorrere tutti i risultati senza dover attendere tempi eccessivi di risposta da parte dell’applicazione, durante fase di popolamento delle righe della tabella. In questo modo si riducono i tempi di attesa e si riduce l’utilizzo della memoria qualora si debbano gestire grossi volumi di risultati.

## Scelta del replica set

MongoDB offre la possibilità di replicare i dati su più server in modo da fornire un servizio quanto più continuativo possibile all’utente che ne fa utilizzo. Per questa applicazione si è stabilito che sono sufficienti almeno 2 server per la replicazione dei dati. Tuttavia è consigliabile scegliere un numero dispari di server di replica. Quindi si sono scelti 3 server per avere una maggior garanzia. MongoDB suddivide i server in due categorie: primario e secondario. Il server primario è quello che accetta letture e scritture da parte degli utenti, mentre i secondari sono coloro che hanno il compito di effettuare scritture solo su ordine del primario: *le repliche*. Il meccanismo con cui viene scelto un server primario avviene mediante elezione della maggioranza, ovvero la metà dei server di replica più uno. In questo modo si evita che si formino più server primari in caso di guasti alla rete generale. Un numero dispari di server ed una corretta redistribuzione nella rete, garantisce che ci sia una maggioranza tale da permettere un’elezione di un nuovo primario in caso di guasti di diverso tipo. È comunque possibile scegliere un numero pari di server di replica, con la possibilità di aggiungere un processo *“arbitro”* con l’unico compito di partecipare alla elezione del primario, tuttavia può comportare alcuni rallentamenti.

# **Fase di implementazione**

## Struttura dei sorgenti

L’applicazione si compone di 5 package al cui interno sono presenti le varie classi:

* **Application**: package di base al cui interno sono presenti le classi
  + - **GestioreRisorse**: contiene le costanti utilizzate nell’applicazione, come il nome e l’indirizzo del database/ set di replica del database, inizializza staticamente la connessione con la base di dati e fornisce una serie di metodi per l’interrogazione, l’inserimento, la modifica e l’eliminazione delle informazioni. Si può considerare una classe di back-end.
    - **Main**: classe che viene eseguita all’avvio dell’applicazione, necessaria per la preparazione dell’interfacce FXML.
* **Container**: package per la raccolta di classi atte alla rappresentazione di un oggetto *riga*, nelle varie *TableView* presenti nel progetto. Su ogni elemento nella tabella infatti possono essere presenti più o meno opzioni, pertanto è opportuno considerare ciascuna riga come un contenitore, contenente opzioni, più informazioni da rappresentare.
  + - **RigaFilm**: contiene opzioni e informazioni relative ad un film. Le opzioni variano a seconda dell’utente che le richiede. Se l’utente è di tipo Standard, prepara le opzioni Dettaglio Film, Aggiungi Recensione, Visualizza Recensioni. Se l’utente è di tipo Admin prepara le opzioni, Visualizza Recensioni, Modifica Film, Elimina Film.
    - **RigaRecensioni**: contiene opzioni e informazioni relative ad una recensione. Le opzioni variano a seconda dell’utente che le richiede. Se l’utente è di tipo Standard ed è il recensore delle recensioni, prepara le opzioni Modifica e Elimina Recensione. Se l’utente è di tipo Admin prepara solo Elimina Recensione.
    - **RigaStatistiche**: contiene solamente informazioni relative ad una statistica.
* **Controller**: package per la raccolta di tutte le classi che gestiscono la logica ed il funzionamento dell’applicazione.
  + - **AutenticazioneController**: questa classe contiene i metodi per la gestione dell’accesso di un utente all’applicazione.
    - **ModificaFilmController**: questa classe contiene i metodi per la gestione dell’inserimento e la modifica di un Film.
    - **ModificaRecensioneController**: questa classe contiene i metodi per la gestione dell’inserimento e la modifica di una Recensione. Contiene un metodo per il popolamento di un campo voto se l’utente decide di gestire l’inserimento dell’informazione da tastiera invece che da mouse (SpinnerFX).
    - **MostraFilmDettaglioController**: questa classe contiene metodi per l’inizializzazione e la preparazione dell’interfaccia per la mostra del dettaglio di un film. Non sono previste ulteriori interazioni con l’utente.
    - **MostraRecensioniController**: questa classe contiene metodi per la visualizzazione della lista di recensioni relativa ad un film a video. Sono previste funzioni per l’aggiornamento dei record a seguito di una modifica/rimozione di una recensione. Viene gestita la possibilità di visualizzare estesa dell’intero contenuto di una cella nella tabella delle recensioni, mediante l’ausilio di un tooltip.
    - **MostraStatisticheController**: questa classe viene richiamata per la gestione di tutte le statistiche sia per utenti Standard che Admin. Contiene metodi per l’inizializzazione e la preparazione dell’interfaccia a seconda della statistica richiesta.
    - **RicercaController**: questa classe contiene i metodi per la preparazione dell’interfaccia di ricerca, per la ricerca effettiva e la visualizzazione di più film. In questa classe si gestiscono anche i metodi per l’apertura delle altre interfacce come quella per le Statistiche e per la gestione del Logout.
* **Model**: package per la raccolta di classi atte a rappresentare informazioni nel database.
  + - **Film**: classe che rappresenta un generico Film. Contiene tutti gli attributi che compaiono frequentemente in un document della Collection Film. Nel metodo costruttore sono previsti dei controlli sulla correttezza e sulla presenza delle informazioni al momento della creazione di un nuovo oggetto. All’interno è prevista una sottoclasse Cast, per la rappresentazione del ruolo interpretato da una persona all’interno di un film.
    - **Recensione**: classe che rappresenta una generica Recensione. Contiene tutti gli attributi che compaiono frequentemente in un document della Collection Recensioni. Nel metodo costruttore sono previsti dei controlli sulla correttezza e sulla presenza delle informazioni al momento della creazione di un nuovo oggetto.
    - **Statistica**: classe che rappresenta una Statistica. Viene utilizzata per più statistiche diverse e contiene tutti gli attributi necessari per la visualizzazione di una determinata statistica.
    - **Utente**: classe che rappresenta un generico Utente. Contiene tutti gli attributi che compaiono in un document della Collection Utenti. Nel metodo costruttore sono previsti dei controlli sulla presenza delle informazioni al momento della creazione di un nuovo oggetto.
* **Test**: package contenente le classi per il test di alcuni componenti dell’applicazione.
  + - **GestoreRisorseTest**: classe contenente test di unità dei metodi di GestoreRisorse.java

## Dettaglio query

In questa sezione sono commentati i metodi della classe Gestione Risorse che effettuano le query su MongoDB.

* **calcolaMediaVotoRecensioniFilm**: esegue la query per il calcolo della media voto di un film passato da parametro e restituisce un documento contenente id\_film, media, num\_recensioni. La query che viene eseguita è la seguente:

db.recensioni.aggregate(

{$match: {"id\_film": ObjectId(“param”}},

{$group:

{\_id:"$id\_film",

media:{$avg:"$voto"},

count:{$sum:1}}

})

* **aggiornaMediaFilm**: richiama la precedente funzione ed aggiorna il valore del film specificato da parametro.
* **ricercaUtente**: esegue la ricerca di un username e password passata come parametro.

La query che viene eseguita è la seguente:

db.utenti.find({"username":"param1","password":"param2"})

* **caricaUltimiNFilm**: riceve in ingresso il numero N di documenti da restituire e ricerca tutti i film dell’anno corrente, la cui data di uscita è inferiore a quella attuale e li visualizza in ordine decrescente per data di uscita. La query che viene eseguita è la seguente:

db.film.find({"anno":{"$eq":anno\_corrente},

"data\_uscita":{"$lte":data\_corrente}})

.sort({“data\_uscita”:-1}).limit(N).

Anno e data corrente vengono impostati programmaticamente, N è il parametro passato.

* **effettuaQueryRicercaFilm**: riceve in ingresso un filtro di ricerca e ricerca un film all’interno della collection Film. La query che viene eseguita è la seguente:

db.film.find({Filtro in input})

* **inserisciFilm**: inserisce un documento film, passato come parametro, all’interno della collection Film. La query che viene eseguita è la seguente:

db.film.insertOne({Documento})

* **aggiornaFilm**: modifica un documento film, passato come parametro, all’interno della collection Film. La query che viene eseguita è la seguente:

db.film.updateOne({\_id:ObjectId(“param”},{documentoDaAggiornare})

* **rimuoviFilm**: rimuove un documento film, il cui Object id è passato come parametro, all’interno della collection Film. La query che viene eseguita è la seguente:

db.film.deleteOne{\_id:ObjectId(“param”}}

* **effettuaQueryRicercaRecensioni**: effettua una query sulla collection Recensioni, applicando un filtro di ricerca passato da parametro.

db.recensioni.find({filtro})

* **inserisciRecensione**: inserisce un documento Recensione, passato come parametro, all’interno della collection Recensioni ed aggiorna la media voto e il numero di recensioni del film a cui è associata.

La query che viene eseguita è la seguente:

db.recensioni.insertOne({Documento})

* **aggiornaRecensione**: modifica un documento recensione, passato come parametro, all’interno della collection Recensioni ed aggiorna la media voto e il numero di recensioni del film a cui è associata. La query che viene eseguita è la seguente:

db.recensioni.updateOne({\_id:ObjectId(“param”},{documentoDaAggiornare})

* **rimuoviRecensione**: rimuove un documento recensione, il cui Object id è passato come parametro, all’interno della collection Film ed aggiorna la media voto e il numero di recensioni del film a cui è associata. La query che viene eseguita è la seguente:

db.recensioni.deleteOne{\_id:ObjectId(“param”}}

* **statisticheMostReviewed**: ricerca tutti i film che hanno ricevuto almeno 1 recensione e con durata maggiore o uguale a 60 minuti, che non hanno genere di tipo documentario elencati in ordine decrescente per numero recensioni, media voto, proiettando solamente i campi *anno*, *nome*, *num\_recensioni*, *media\_voto*. La query eseguita è la seguente:

db.film.find({"num\_recensioni":{$gt:0},

"durata\_min":{$gte:60},

"genere":{$nin:["Documentario"]}

},

{anno:1, nome:1, num\_recensioni:1, media\_voto:1}

).sort({num\_recensioni:-1,media\_voto:-1})

* **statisticheBestByYear**: ricerca di tutti i migliori Film per anno i quali hanno ricevuto almeno 5 recensioni e hanno la media voto maggiore degli altri nel rispettivo anno. A parità di media voto si considera il film con più recensioni. Questa Lista viene restituita in ordine crescente per anno. La query eseguita è la seguente:

db.film.aggregate([

{$match:{"num\_recensioni":{$gt:5}}},

{$sort:{"media\_voto":-1,"num\_recensioni":-1}},

{$group:{\_id:{anno:"$anno"},voto:{$max:"$media\_voto"},

nome\_film:{$first:"$nome"},

num\_recensioni:{$first:"$num\_recensioni"}}},

{$project:{\_id:0,

"anno":"$\_id.anno",

"nome":"$nome\_film",

"media\_voto":"$voto",

"num\_recensioni":"$num\_recensioni"}

{$sort:{"anno": 1}}],{allowDiskUse:true})

* **statisticheDetailReviews**: ricerca del numero di recensioni effettuate da ciascun Utente, suddivise in totaleRecensioni, numero recensioni positive, numero recensioni negative e media voto complessiva per Utente. La lista degli Utenti viene restituita in ordine decrescente per numero di recensioni totali effettuate. La query effettuata è la seguente:

db.recensioni.aggregate([

{$group:{\_id:{utente:"$id\_utente"},

utente:{$first:"$username"},

totRec:{$sum:1},

media:{$avg:"$voto"},

positive:{$sum:{$cond:[{$gte :["$voto",6]},1,0]}},

negative:{$sum:{$cond:[{$lt:["$voto",]},1,0]}}}},

{$project:{\_id:0,utente:1,"num\_recensioni":"$totRec", "media\_voto":"$media",positive:1,negative:1}},

{$sort:{"num\_recensioni":-1,"media\_voto":1}

}],

{allowDiskUse:true}

)

## Test eseguiti

Per l’esecuzione dei test di unità, si utilizza il framework *Junit*. Per poterlo utilizzare all’interno dell’ambiente di sviluppo Eclipse è necessario aggiungere alle dipendenze del progetto:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

Il suo utilizzo è semplice. Occorre annotare tutte le classi che si vogliono testare con l’annotazione @Test. Per la verifica dei test, occorre cliccare con il tasto destro sulla classe di test e selezionare l’opzione: Run as.. JUnit Test

Vista la suddivisione dei compiti dovuta al modello MVC adottato, si è deciso di effettuare i test unicamente sulla classe Gestore Risorse.java dove sono presenti tutti i metodi per l’interazione con la base di dati. Le restanti classi del controller si limitano a richiamare questi metodi e a mostrare a video i risultati.

In GestioneRisorseTest.java, si istanziano degli oggetti fittizi di tipo Document (Utente, Film, Recensione) da utilizzare durante i test nelle varie ricerche delle rispettive collection e si va a creare la connessione necessaria per connettersi al database. Questa operazione avviene prima di iniziare ad eseguire tutti i test, pertanto si utilizza l’annotazione @BeforeClass sulla funzione di inizializzazione di questa classe. Analogamente al termine di tutti i test è presente un metodo per la chiusura e rimozione di tutti i documenti fittizi creati durante i test e viene annotato con l’annotazione @AfterClass. A ciascun documento aggiunto in fase di test viene introdotto un nuovo campo *type* con valore ‘*test’*, per identificare tutti i documenti da rimuovere al termine dell’esecuzione. I test consistono nel richiamare un metodo alla volta della classe GestoreRisorse.java e verificare che la sua logica sia la stessa per cui è stato progettato e restituisca i risultati aspettati. Per ciascun test è presente nel codice un breve commento che specifica cosa si va a testare in ogni metodo.

## Integrazione dei dati

Al fine di avere una base di dati popolata con un grosso volume di dati reali si sono adottate diverse strategie per ciascuna collection.

La collection Utenti viene popolata con 31 documenti, creati manualmente.

La collection Film viene popolata con documenti ottenuti effettuando *Web scraping and crawling* dal sito internet <https://www.mymovies.it/>.

La collection Recensioni viene popolata con documenti ottenuti dalla manipolazione di 3 dataset relativi a commenti di utenti reali sul sito IMDB. I tre dataset sono scaricabili dal sito <https://www.kaggle.com> rispettivamente:

* <https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>
* <https://www.kaggle.com/utathya/imdb-review-dataset>
* <https://www.kaggle.com/nltkdata/movie-review>

Complessivamente la dimensione della base di dati supera i 200 MB.

Per la collection Utenti si sono creati 30 documenti di utenti tutti di tipologia *standard* in modo che ciascuno di essi possieda un numero considerevole di recensioni. Infine è stato aggiunto un solo utente di tipologia *amministratore*.

L’operazione di *Web scraping* prevede l’analisi della struttura del documento di una pagina HTML del sito di MYmovies. Si nota subito che sono presenti strutture simili per l’impaginazione e la visualizzazione dei film. All’ interno di ciascuna pagina è possibile identificare contenitori di più film, da ciascuno dei quali è possibile estrapolare informazioni come *Titolo*, *Anno,* *Durata*, *Generi*, *Data Uscita*, *Trama Completa* e *Paesi Produzione*. Si nota inoltre la presenza di un link il quale a sua volta rimanda ad una pagina web in cui compare l’intera elencazione del *Cast* di tutto il personale che compone il film. Cambiando i parametri nell’ url della richiesta della pagina web è possibile ricercare insiemi di film raggruppati per anno.

Esempio di richiesta

<https://www.mymovies.it/film/yyyy/?p=n>

dove

yyyy, rappresenta l’anno di uscita dei film

n, è il numero della pagina di visualizzazione dei film riferita a quell’anno

Facendo variare l’anno da 1990 a 2020 e variando il numero di pagina per ciascun anno è possibile estrarre un cospicuo numero di informazioni, le quali a sua volta vengono memorizzate in un file esterno in formato JSON. Successivamente si è importato il file generato nella collection Film.

I tre dataset trovati sulla rete presentano parzialmente le informazioni necessarie per questo progetto. In tutti è presente un commento testuale, al quale è stato associato un livello di sentimento espresso come positivo/negativo oppure come 0/1. Si importano quindi in una collection e si rimuovono tutte le altre informazioni che non occorrono. Questo è possibile grazie alla flessibilità di MongoDB. Si attribuisce una valutazione random che varia da 0 a 5.9 per tutte le recensioni che presentano un livello di sentimento negativo oppure 0. Si attribuisce una valutazione random che varia da 6 a 10 per tutte le recensioni che presentano un livello di sentimento positivo oppure 1. In questo modo si ottiene una collection contenente solamente *Commento* e *Voto*.

Anche l’associazione tra utente e recensione e tra recensione e film avviene in maniera casuale.

Per poter associare gli utenti alle recensioni si procede nel seguente modo:

* per ciascun documento utente si aggiunge un nuovo campo numerico *id\_utente* incrementando di uno per ciascun utente standard (da 1 a 30).
* per ciascuna recensione si estrae casualmente un numero da (1 a 30) e si inseriscono rispettivamente l’ObjectId *\_id* e *username* del documento utente con *id\_utente* uguale al numero estratto, all’interno della recensione con il nome di *id\_utente* e *username*.
* si rimuove da tutti i documenti utenti il campo numerico *id\_utente*, precedentemente aggiunto.

Per poter associare le recensioni ai film si procede nel seguente modo:

* per ciascun documento film si aggiunge un nuovo campo numerico *id\_film* incrementandolo di uno per ciascun film da (1 a NUM\_FILM).
* per ciascuna recensione si estrae casualmente un numero da (1 a NUM\_FILM) e si inserisce l’ObjectId del documento film con *id\_film* uguale al numero estratto, all’interno della recensione con il nome di *id\_film*.
* si rimuove da tutti i documenti film il campo numerico *id\_film*, precedentemente aggiunto.

Una volta terminati gli abbinamenti, calcolo la media voto e il numero di recensioni che ha ricevuto ciascun film e li inserisco rispettivamente in ciascun documento film nei campi *media\_voto* e *num\_recensioni*.

Le operazioni eseguite su MongoDB vengono riportate in un file allegato nel progetto nel percorso *doc/DataIntegration.txt*. Nello stesso percorso è presente la JavaDoc di tutti i metodi del progetto.

## Configurazione del Replica Set

La configurazione che viene mostrata fa riferimento ad una rete locale. Per poterla utilizzare in una rete pubblica è necessario conoscere il proprio indirizzo IP pubblico, ed impostare le regole di port forwarding per permettere l’accesso dell’esterno.

Per prima cosa si identificano i server (indirizzo + porta) su cui si vanno ad effettuare le repliche. Disponendo di 2 soli computer, si configurano i 3 server nel seguente modo:

* **server 1 "192.168.1.192:27017" (computer 1)**
* **server 2 "192.168.1.191:27017" (computer 2)**
* **server 3 "192.168.1.191:27018" (computer 2)**

Assicurarsi che da ciascun server sia possibile connettersi a tutti gli altri mediante mongo shell:

mongo --host indirizzo --port num\_porta

Qualora non fosse possibile è necessario aggiungere delle regole di firewall in entrata che ne permettano l’accesso. Prima di avviare i server assicurarsi che non siano presenti, altri servizi attivi sulla porta selezionata. Fatto ciò, si avviano i server singolarmente, passando come parametri:

* l’indirizzo ip della macchina su cui si avvia il server
* il nome del replica set (uguale per tutti)
* il percorso e la porta dove verrà replicato il database sul singolo server
* la dimensione dell’oplog, per la lista di scritture da eseguire in replica:



Accedere ad uno dei 3 server mediante mongo shell:

mongo --host 192.168.1.192 --port 27017

Creare il seguente documento di configurazione:

rsconf = {\_id: "test",

members: [

{\_id: 0, host: "192.168.1.192:27017"},

{\_id: 1, host: "192.168.1.191:27017"},

{\_id: 2, host: "192.168.1.191:27018"}

]

}

Impostare tale documento, come documento di configurazione del replica set:

rs.initiate(rsconf)

I tre server iniziano a comunicare tra loro eleggendo il server primario e successivamente procedono nella fase di sincronizzazione iniziale, in cui i dati presenti sul primario vengono replicati sui secondari. Il replica set è configurato.

È necessario adesso configurare il driver dell’applicazione in modo che questa comunichi con l’intero set di replica. Si modifica la stringa di connessione, presente in GestioneRisorse.java specificando i server che fanno parte del set di replica:

mongodb://192.168.1.192:27017,192.168.1.191:191:27017,192.168.1.191:27018

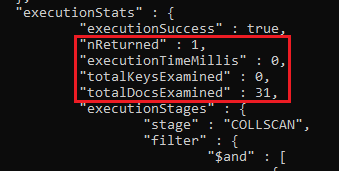
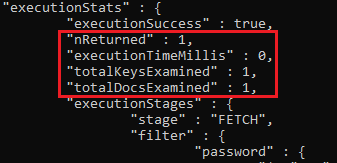
# **Valutazioni delle performance**

In questa parte si valutano i benefici tratti dall’aggiunta degli indici nelle collection. Si analizzano singolarmente la query effettuate e si fa una comparazione delle loro performance con e senza l’utilizzo di indici. Infine si testa il funzionamento della applicazione accedendo da una rete esterna e si vanno a valutare le caratteristiche di risposta in relazione al teorema CAP.

## Indice username: query di autenticazione

db.utenti.find({"username":{$eq:"prova"},"password":{$eq:"00"}}

In questa query si ricerca il documento con *username “*prova*”* e *password* “00”.

Senza indice Con Indice

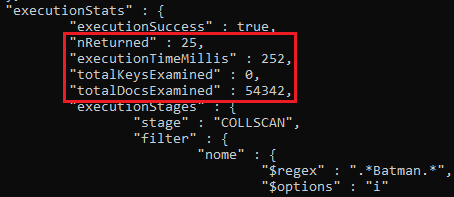
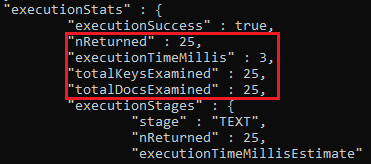
Essendo il totale dei documenti nella collection *utenti* uguale a 31 non se ne percepisce un immediato beneficio, in quanto il tempo di esecuzione è per entrambi i casi inferiore al millisecondo. Diventa invece importante con l’aumentare degli utenti poiché permette di non scorrere l’intera collection per la ricerca di un singolo documento,velocizzando la fase di autenticazione.

## Indice nome\_text: query ricerca film per nome

db.film.find({"nome" : /.\*Batman.\*/i})- senza indice

db.film.find({ $text: { $search: "Batman"}}) – con indice

In questa query si ricercano i film il cui titolo contiene la scritta “Batman”. La ricerca inoltre è case insensitive.

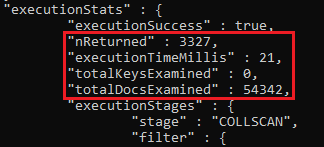
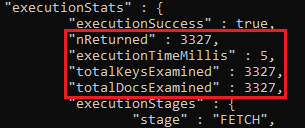
Senza indice Con Indice

La differenza tra le prestazioni è evidente. Il numero dei documenti esaminati è nettamente inferiore, come lo è il tempo di esecuzione. Sebbene 252 millisecondi non siano molti, bisogna considerare anche i tempi di ritardo della rete che possono andarsi a sommare. Pertanto l’utilizzo di questo indice avanzato risulta molto utile.

## Indice anno: query ricerca film per anno

db.film.find({"anno":{$eq:2016}})

In questa query si ricercano tutti i film usciti nell’*anno* 2016.

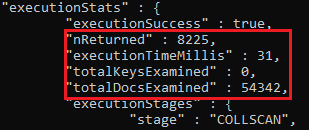
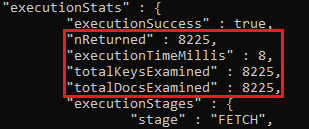
Senza indice Con Indice

Anche questo indice riduce il tempo di esecuzione della query evitando di scorrere tutti i film nella collection. Il numero dei documenti totali esaminati è molto inferiore.

## Indice genere: query ricerca film per genere

db.film.find({"genere":"Commedia"})

In questa query si ricercano tutti i film con *genere* Commedia.

Senza indice Con Indice

Anche questo indice riduce il tempo di esecuzione della query evitando di scorrere tutti i film nella collection. Il numero dei documenti totali esaminati è molto inferiore.

## Indici num\_recensioni e media\_voto: query ricerca film più recensiti

db.film.find({"num\_recensioni":{$gt:0},

"durata\_min":{$gte:60},

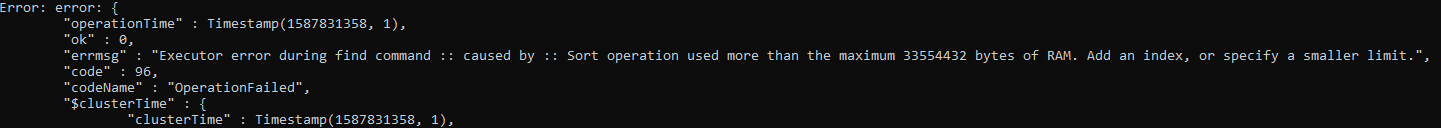
"genere":{$nin:["Documentario"]}

},

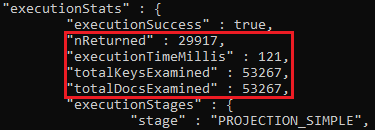
{anno:1, nome:1, num\_recensioni:1, media\_voto:1}

).sort({num\_recensioni:-1,media\_voto:-1})

In questa query si ricercano tutti i film che hanno ricevuto almeno una recensione e con *durata* maggiore o uguale a 60 minuti e che non hanno *genere* di tipo documentario elencati in ordine decrescente per *numero recensioni*, *media voto.*



Senza indice



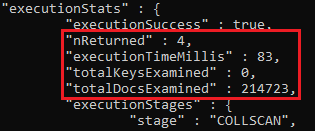
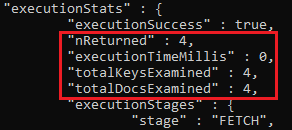
Con Indice

Questa query prevede che un grande numero di documenti venga mostrato in ordine decrescente per numero recensioni. Eseguendo la query senza indice, si riceve un messaggio di errore, poiché tutti i record trovati vengono caricati in memoria per l’operazione di ordinamento. L’utilizzo dell’indice permette l’esecuzione senza errori riducendo l’utilizzo di memoria per l’ordinamento.

## Indice id\_film: query ricerca tutte le recensioni di un film

db.recensioni.find({"id\_film":ObjectId("5e49c01afde967f95b37b754")})

La query ricerca tutte le recensioni il cui ObjectId è “5e49c01afde967f95b37b754”.

Senza indice Con Indice

Anche in questa query, l’utilizzo dell’indice si conferma fondamentale, specialmente se si considera che la sua esecuzione avviene con maggior frequenza, poiché presente nei casi d’uso di entrambe le tipologie di utente.

## Valutazione delle funzionalità: teorema CAP

Il teorema CAP è basato sui concetti di Consistenza (C), Disponibilità (A) e Tolleranza ai Guasti (P). Un sistema distribuito può avere al massimo due delle tre le caratteristiche indicate. MongoDB garantisce Consistenza e Tolleranza ai guasti. Tutte le operazioni di scrittura devono infatti essere effettuate dal server primario che a sua volta, mantiene i secondari aggiornati. Le letture invece possono essere fatte anche sui secondari, se richiesto esplicitamente, ma di norma avvengono anch’esse sul primario.

All’avvio dell’applicazione client, si stabilisce una connessione con il set di replica e viene subito identificato il server primario.

La consistenza si basa sul fatto che ciascun server di replica possiede sempre le informazioni aggiornate. Le sincronizzazioni avvengono durante la configurazione iniziale del set di replica e ad ogni scrittura sul server primario. Se il server primario si disconnette, ne viene eletto uno nuovo che ne prende il posto. Se dovesse tornare online e operativo, viene eseguita un’operazione di rollback in cui si annullano le operazioni che precedentemente non sono state replicate. Queste vengono memorizzate in un file di log e successivamente, ed eventualmente recuperate, mantenendo quindi la consistenza tra i dati sui server.

L’intero sistema è tollerante ai guasti fintanto che la *maggioranza* dei server sono attivi e funzionanti, all’interno del set di replica. La maggioranza è calcolata come la metà più uno dei server. Fintanto che il numero dei server operativi è superiore o uguale alla maggioranza è possibile eleggere un nuovo server primario e l’applicazione non risente alcun cambiamento.

In questa parte si vanno a testare e valutare queste due proprietà. Si decide di connettersi al replica set da una rete esterna, andando a verificare che i tempi di risposta siano brevi come da requisito. Per fare ciò si riconfigura il set di replica, specificando gli indirizzi ip pubblici e si inseriscono regole di *port forwarding* nel router in modo da consentire l’accesso dall’esterno. Di seguito vengono indicati come vengono tradotti gli indirizzi precedentemente elencati:

* **server 1 "192.168.1.192:27017"🡪 79.51.200.37:27016 (computer 1)**
* **server 2 "192.168.1.191:27017"🡪 79.51.200.37:27017 (computer 2)**
* **server 3 "192.168.1.191:27018"🡪 79.51.200.37:27018 (computer 2)**

### **Valutazione della consistenza**

Sebbene il driver di MongoDB comunichi direttamente col server primario, permette di effettuare letture anche ai server secondari, modificando la stringa di connessione. In questa parte si testa il funzionamento di replica andando a verificare che il server primario effettui la replica su entrambi i server secondari.

Si avvia l’applicazione normalmente, connettendosi al set di replica, indicando il seguente indirizzo nella classe *GestioneRisorse.java*:

INDIRIZZO\_DATABASE= "mongodb://79.51.200.37:27016,79.51.200.37:27017,79.51.200.37:27018"

Si accede con un utente di tipo amministratore e si effettua una scrittura sul primario, inserendo un film.

Successivamente si avviano altre 2 istanze dell’applicazione presenti su un altro computer di una rete esterna, modificando l’indirizzo di connessione e inserendo singolarmente per ciascuna istanza i due indirizzi dei server secondari.

INDIRIZZO\_DATABASE=79.51.200.37:27017

INDIRIZZO\_DATABASE=79.51.200.37:27018

Si accede ad entrambe le istanze con un qualsiasi tipo di utente e si ricerca il film precedentemente inserito. Entrambe visualizzano il film correttamente. Il film può essere modificato ulteriormente dall’utente amministratore. Effettuando nuovamente la ricerca su entrambi i secondari si può osservare che nel giro di pochi millisecondi tale modifica viene aggiornata su entrambi.

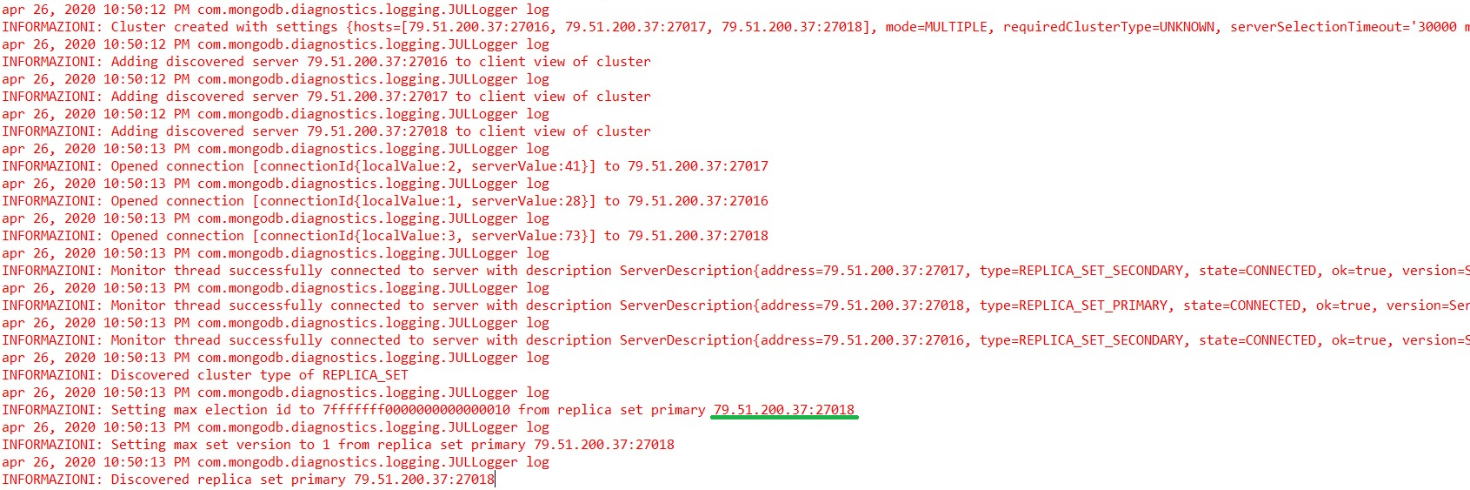
### **Valutazione della tolleranza ai guasti**

L’obbiettivo è verificare il regolare funzionamento dell’applicazione qualora un server nel set di replica vada offline. Fintanto che la maggioranza dei 3 server, cioè 2, è online, è possibile effettuare letture e scritture senza che l’applicazione si accorga dell’assenza di un server. Quando 2 server su 3 vanno offline, è impossibile effettuare la scelta del primario e l’applicazione genera dei messaggi di errore nel file di log.

Si avvia l’applicazione normalmente, connettendosi al set di replica, indicando il seguente indirizzo nella classe *GestioneRisorse.java*:

INDIRIZZO\_DATABASE= "mongodb://79.51.200.37:27016,79.51.200.37:27017,79.51.200.37:27018"

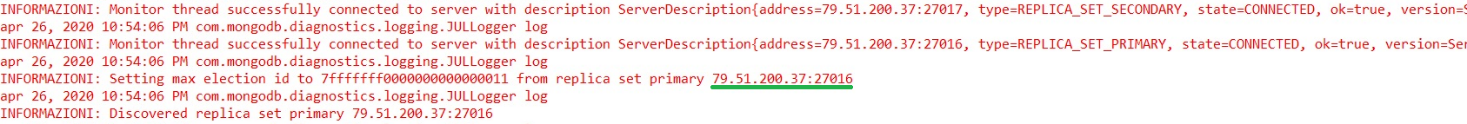
All’avvio dell’applicazione i messaggi di log sono i seguenti:



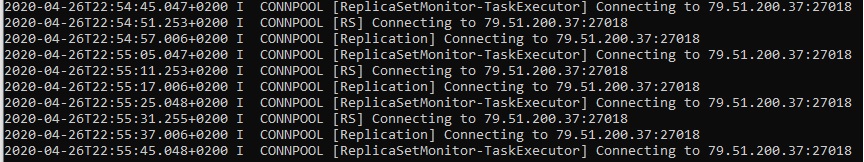
Il driver dell’applicazione testa la connessione con tutti i membri del replica set e identifica quale fra questi è il server primario a cui manderà richieste.

Successivamente si decide di disconnettere il server primario, quello all’indirizzo 79.51.200.37:27018.

I due server rimanenti iniziano quindi il meccanismo di elezione di un nuovo server primario. In questo caso viene eletto il quello a indirizzo 79.51.200.37:27016, come si legge dai successivi file di log. Questa operazione è trasparente all’applicazione.



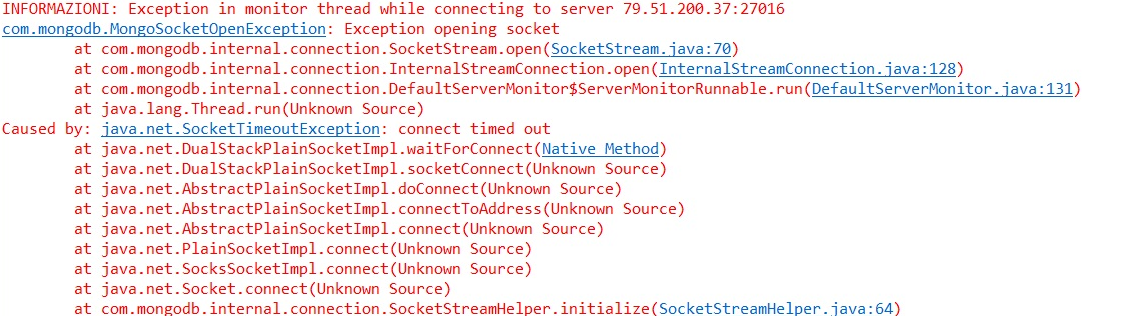
Entrambi i due server rimasti online, continuano a mandare richieste all’indirizzo che è andato offline, effettuando dei ping per verificare se si fosse riconnesso:



In questo momento l’applicazione può effettuare operazioni di lettura e scrittura.

Disconnettendo un altro server, ad esempio quello a indirizzo 79.51.200.37:27016, si osserva che l’applicazione è impossibilitata a fare letture e scritture, poiché non è possibile raggiungere la maggioranza per eleggere un nuovo server primario.

In console vengono generati i seguenti messaggi di errore:

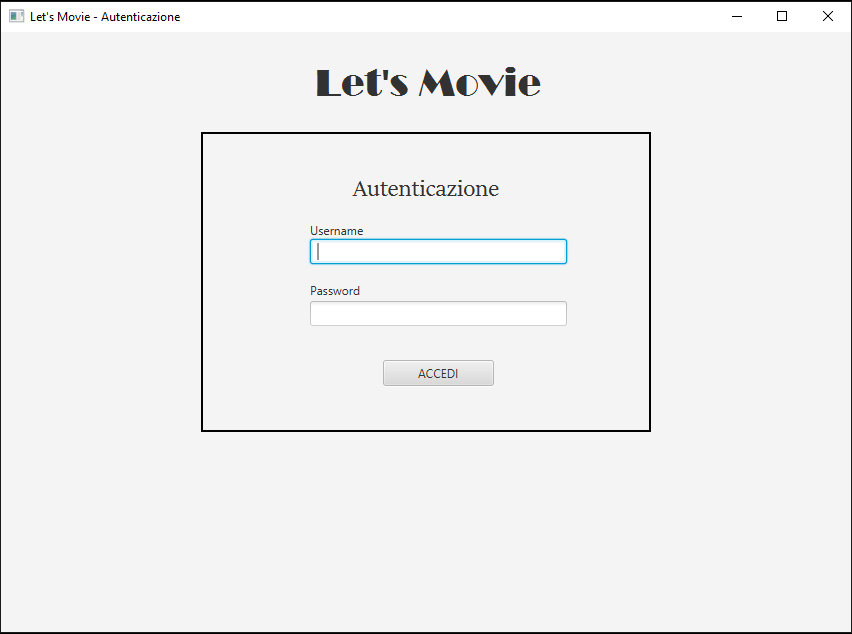


Non appena uno dei due server offline tornerà online, l’applicazione riprende il normale funzionamento.

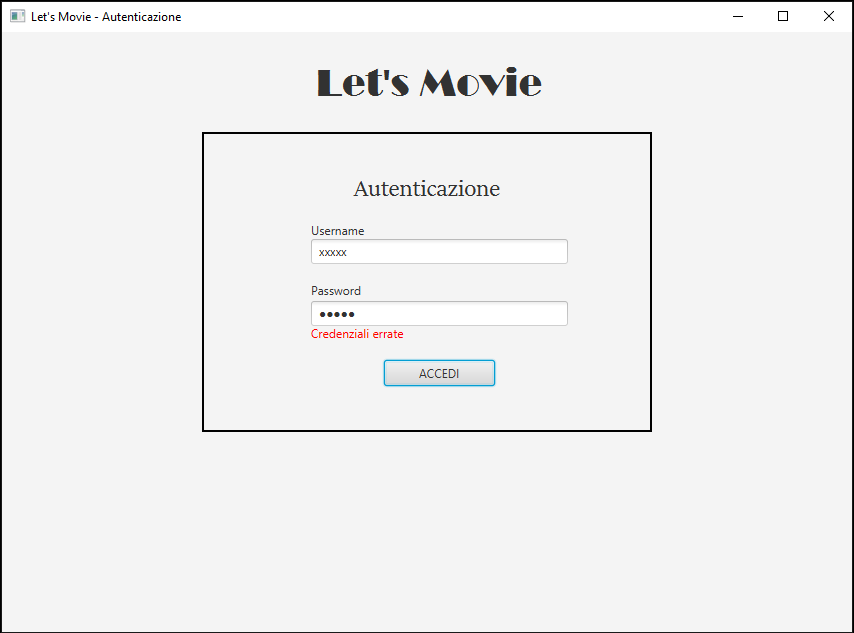
# **Manuale d’uso**

Per prima cosa occorre assicurarsi che sia stato avviato correttamente il/i server su cui è presente il database dell’applicazione. Assicurarsi di aver importato le collection prodotte a seguito dell’integrazione dei dati.

Al primo avvio l’applicazione si presenta nel seguente modo:



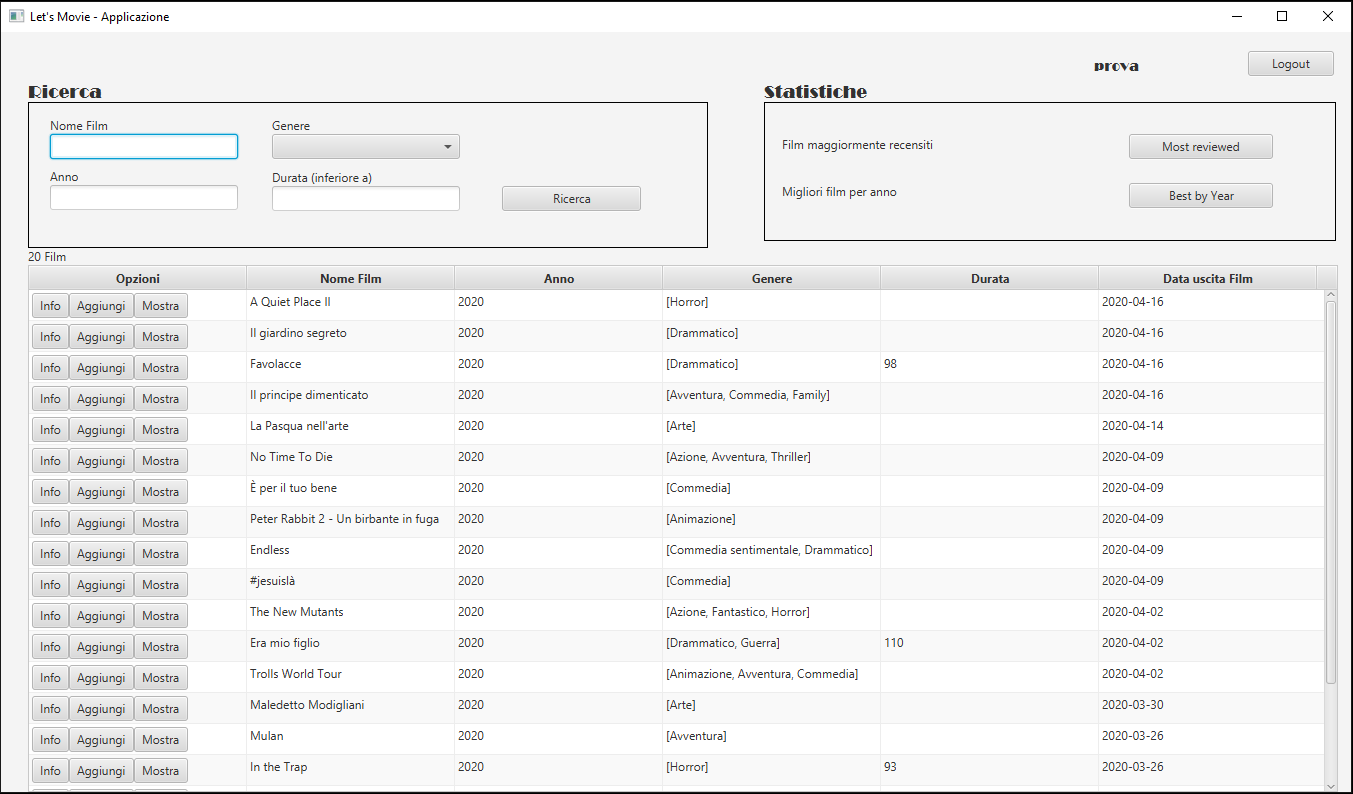
L’utente inserisce le proprie credenziali, *username* e *password*, e preme il pulsante *ACCEDI*. Se le credenziali immesse sono corrette l’applicazione distingue la tipologia di utente con cui si sta effettuando l’accesso e reindirizza l’utente alla specifica interfaccia di ricerca film, altrimenti l’applicazione mostra il messaggio di errore, *“Credenziali errate”*.



## Utente Standard

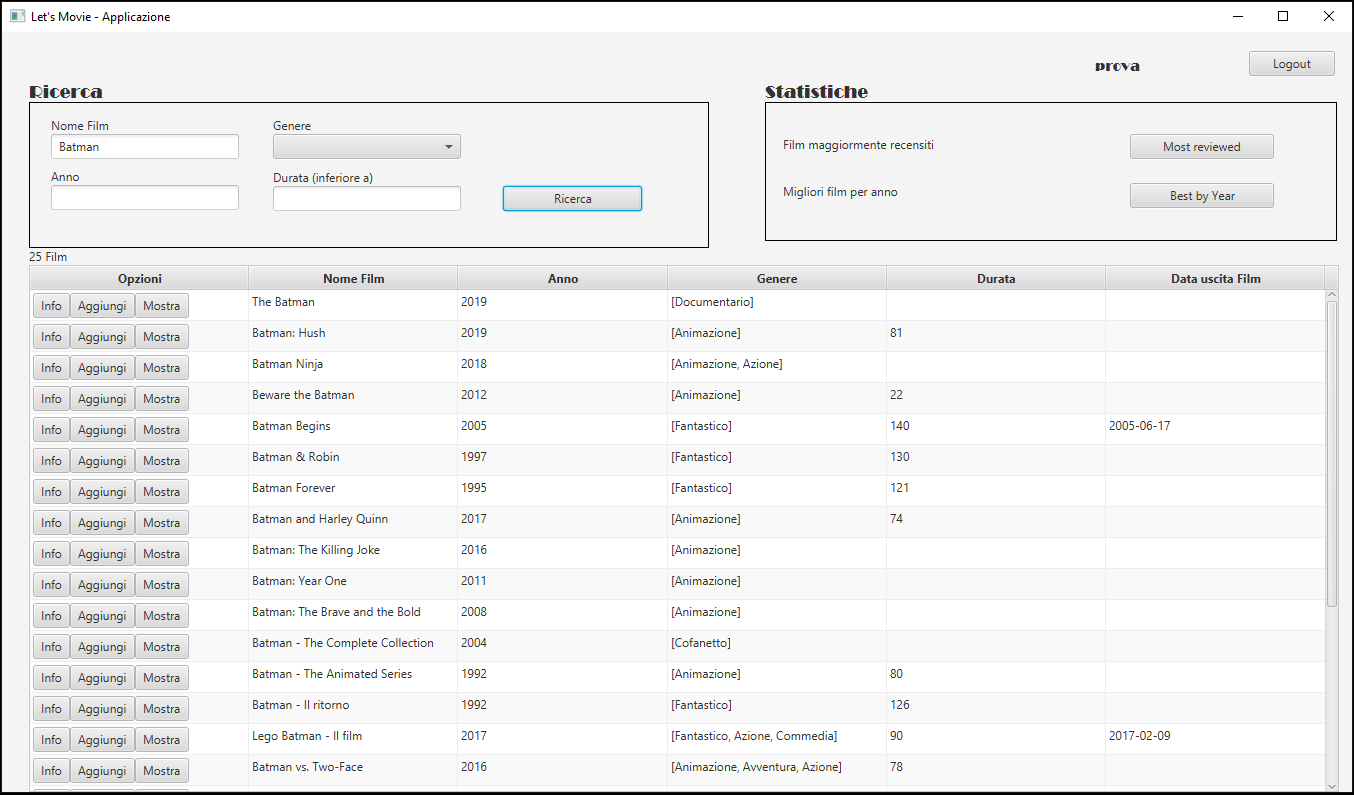
Per accedere come utente standard, è stato inserito un utente di test accessibile con le credenziali username = “*prova”* e password = “*00”*.

L’interfaccia di ricerca film di un utente standard è la seguente:



L’applicazione mostra gli ultimi 20 film in ordine decrescente di data uscita.

L’utente per ricercare un film può inserire valori nei campi di ricerca *Nome Film*, *Genere*, *Anno* e *Durata*. Successivamente premendo il pulsante *Ricerca*, l’applicazione mostra il risultato della ricerca effettuata, indicando il numero dei film trovati e la lista dei film che corrisponde alla ricerca inserita. Ad esempio, inserendo come ricerca il nome film “Batman” l’applicazione ricerca tutti i film che contengono la stringa passata nel campo nome film, il risultato è il seguente:



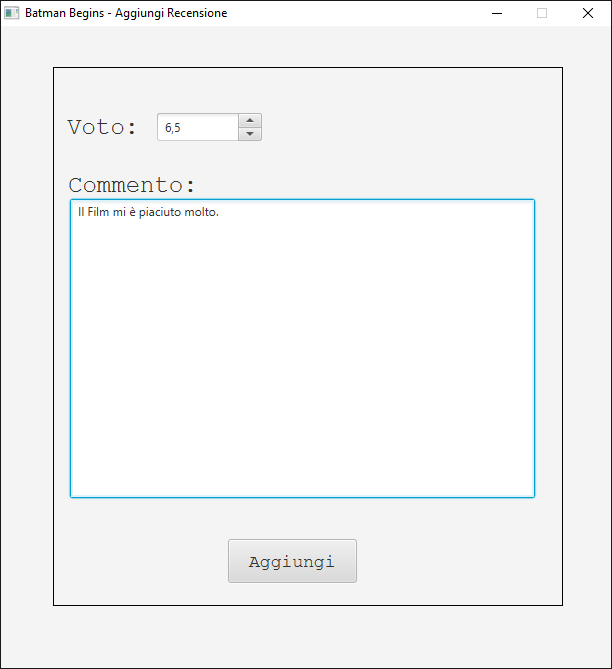
Per ciascun film presente nella tabella è possibile eseguire delle opzioni.

Cliccando su *Info*, l’applicazione mostra nel dettaglio, in una nuova finestra, tutte le informazioni relative al film su quella riga.

Es. sul film “*Batman Begins*”:

****

Cliccando su *Aggiungi*, l’applicazione apre una nuova interfaccia che consente l’inserimento di una nuova recensione relativa a quel film.



L’utente inserisce il *Voto* e il *Commento*, quindi preme *Aggiungi* per registrare la propria recensione.

Cliccando su *Mostra*, l’applicazione apre una nuova finestra contenente la lista delle recensioni effettuate da tutti gli utenti relative a quel film.

**

Su ciascuna recensione in cui l’utente autenticato è lo stesso che ha effettuato la recensione sono visibili due pulsanti.

Cliccando su *Modifica*, l’applicazione carica i dati di quella recensione e rimanda all’interfaccia precedente (*Aggiungi)*.

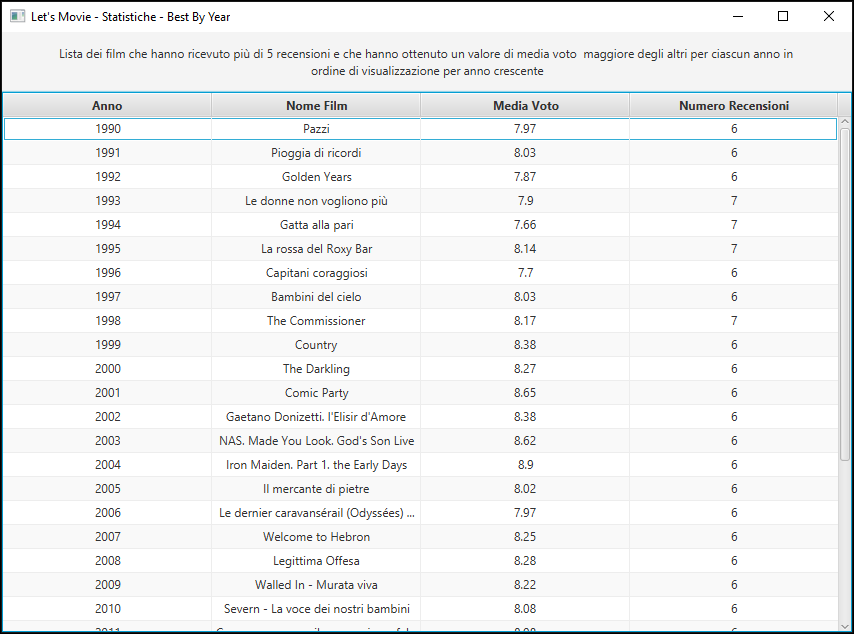
Cliccando su *Elimina*, l’applicazione rimuove la recensione della lista.

L’utente che è interessato a vedere alcune statistiche, come il livello di gradimento dei film da parte degli altri utenti, può cliccare su due pulsanti presenti nel pannello statistiche.

Cliccando su *Most Reviewed*, l’applicazione apre un’interfaccia per la visualizzazione di tutti i film maggiormente recensiti, ordinati in ordine decrescente per numero recensioni. L’interfaccia è la seguente:



Cliccando su *Best by Year*, l’applicazione apre un’interfaccia per la visualizzazione di tutti i film raggruppati per anno, che hanno la media voto più alta degli altri film dello stesso anno, ordinati in ordine crescente per anno di uscita. L’interfaccia è la seguente:

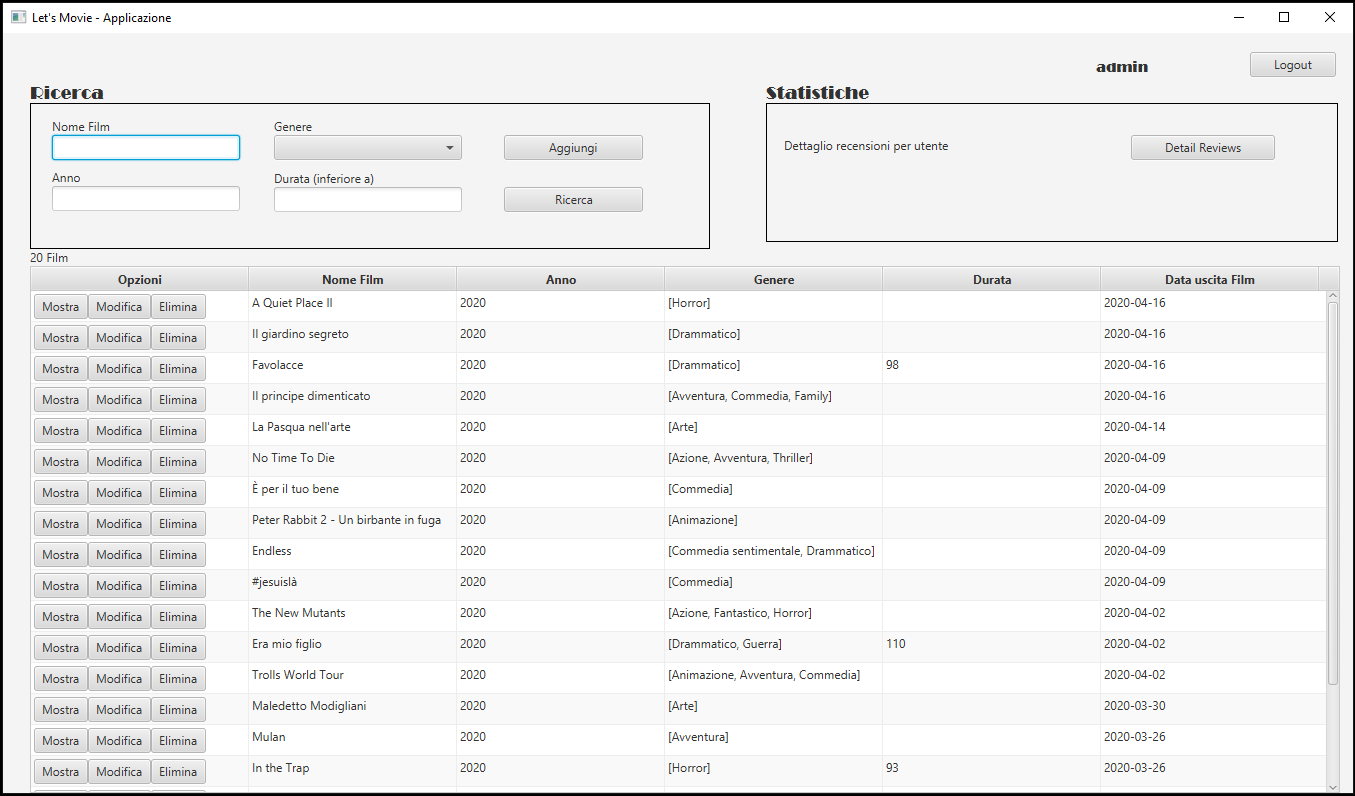


Cliccando su *Logout*, l’applicazione chiude l’interfaccia corrente e apre nuovamente l’interfaccia di autenticazione.

## Utente Amministratore

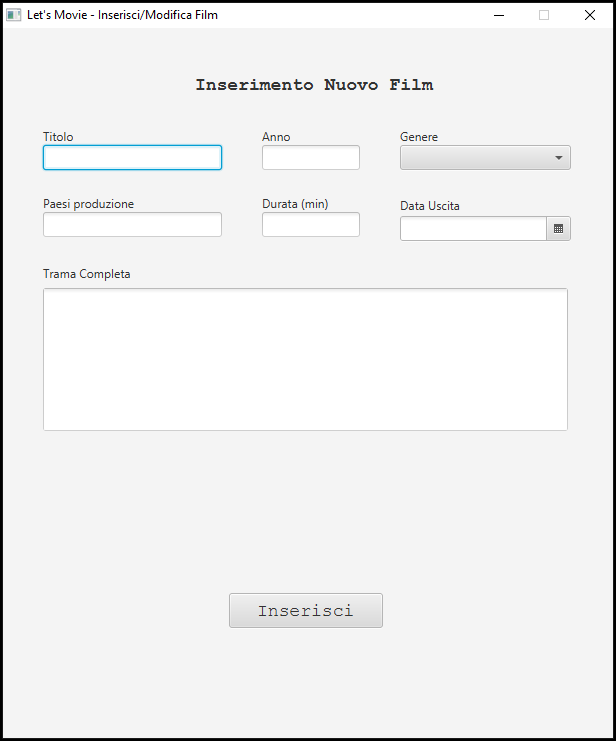
Per accedere come utente admin, è stato inserito un utente di test accessibile con le credenziali username = “*admin”* e password = “*00”*.

L’interfaccia di ricerca film di un utente amministratore è la seguente:



L’utente per ricercare un film può inserire valori nei campi di ricerca *Nome Film*, *Genere*, *Anno* e *Durata*. Successivamente premendo il pulsante *Ricerca*, l’applicazione mostra il risultato della ricerca effettuata, indicando il numero dei film trovati e la lista dei film che corrisponde alla ricerca inserita. Questa funzionalità è in comune con la categoria utente Standard.

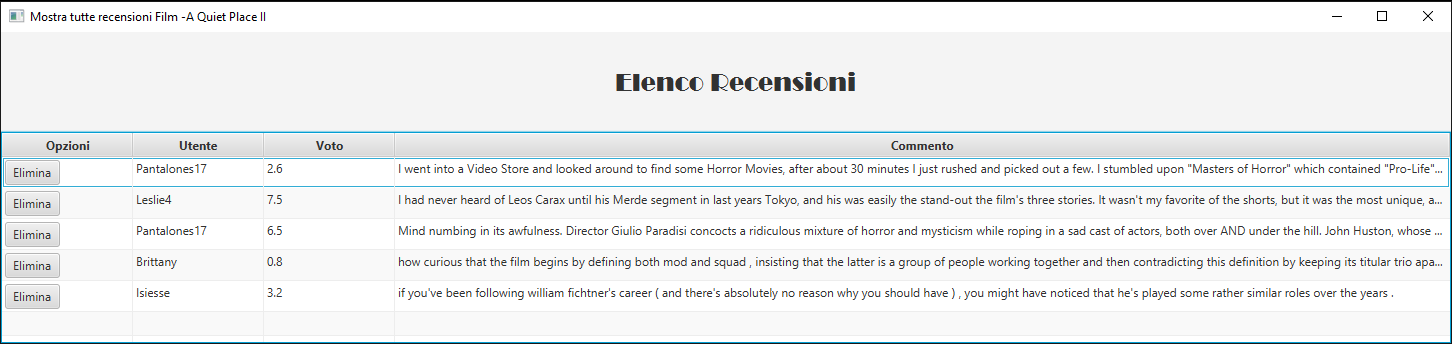
Cliccando sul pulsante *Aggiungi*, l’applicazione mostra un’interfaccia per l’inserimento di un nuovo film.



L’utente inserisce quindi *Titolo*, *Anno, Genere* (obbligatoriamente) e *Paesi produzione*, *Durata*, *Data Uscita* (facoltativamente) quindi premendo il pulsante *Inserisci* l’applicazione registra l’inserimento di un nuovo film lo aggiunge nella lista dei film.

Per ciascun film presente nella tabella è possibile eseguire delle opzioni.

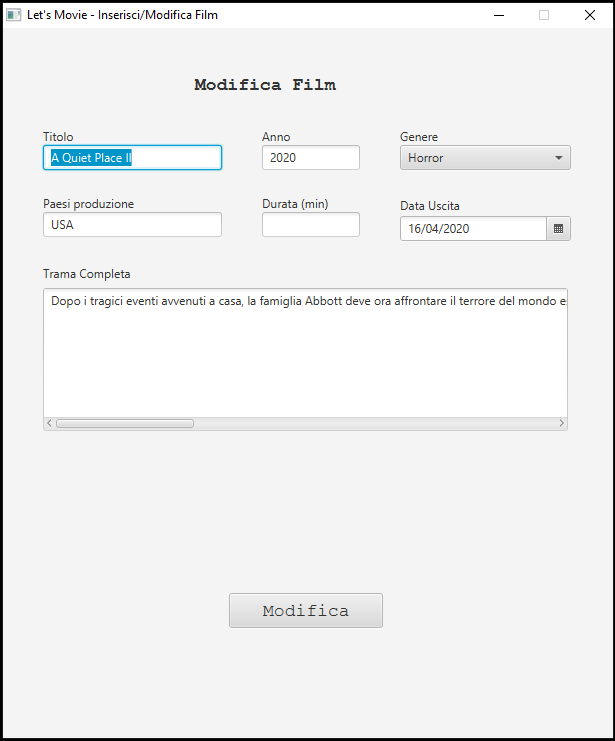
Cliccando su *Mostra*, l’applicazione apre una nuova finestra contenente la lista delle recensioni effettuate da tutti gli utenti relativa a quel film.



Su ciascuna recensione è presente un pulsante *Elimina*.

Cliccando su *Elimina*, l’applicazione rimuove la recensione della lista.

Cliccando su *Modifica*, l’applicazione carica le informazioni relative a quel film e le mostra in una nuova interfaccia, la stessa utilizzata per l’inserimento.

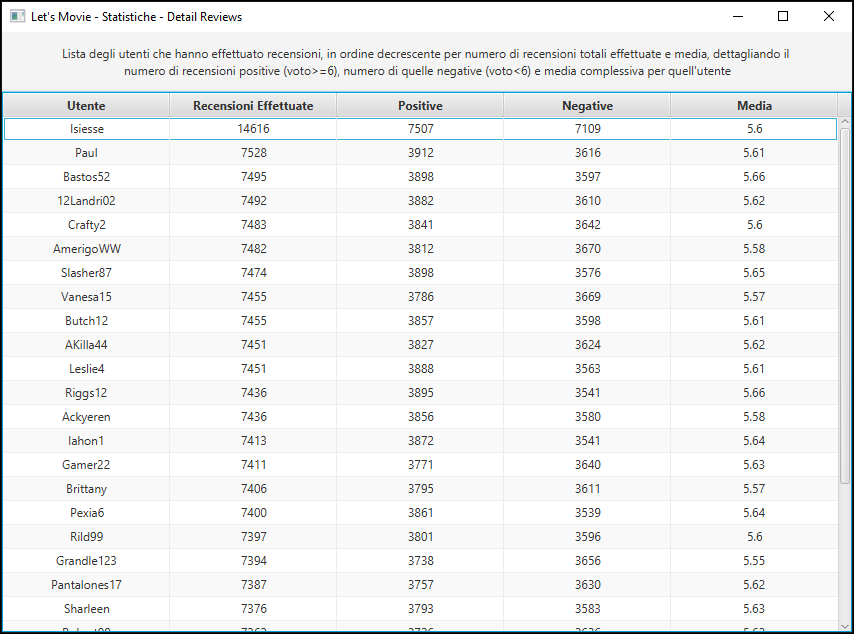


Cliccando sul nuovo pulsante *Modifica*, l’applicazione provvede ad aggiornare le informazioni relative a quello specifico film.

Cliccando su *Elimina*, l’applicazione provvede a eliminare il film in questione dalla lista dei film.

L’utente che è interessato a vedere in che modo gli utenti Standard effettuano recensioni, per capirne l’attitudine di attribuzione di voto e analizzane statistiche, può farlo grazie al pulsante presente nel pannello statistiche.

Cliccando su *Detail Reviews*, l’applicazione apre un’interfaccia dove viene mostrata una lista di utenti recensori, nella quale viene illustrato nel dettaglio il numero totale di recensioni, il numero di recensioni positive, negative e la media voto relativa a ciascun utente. Gli utenti sono visualizzati in ordine decrescente per recensioni effettuate, media voto. L’interfaccia è la seguente:



Cliccando su *Logout*, l’applicazione chiude l’interfaccia corrente e apre nuovamente l’interfaccia di autenticazione.